2014-ENST-0069

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Samuel KALUVURI

14-11-2014

Certification et preuves de sécurité dans les architectures orientées services

Directeur de thèse : **Yves ROUDIER**

Jury

M. Philippe COLLET, Professeur, Université de Nice Sophia Antipolis, France	Président de jury
M. Nora CUPPENS-BOULAHIA, Directeur de Recherche, Télécom Bretagne, France	Rapporteur
M. François CHAROY, Professeur, Université de Lorraine, France	Rapporteur
M. Refik MOLVA, Professeur, EURECOM, Biot, France	Examinateur
M. Michele BEZZI, Research Manager, SAP Labs France, Mougins, France	Invité

TELECOM ParisTech

école de l'Institut Télécom - membre de ParisTech

T H È S E

TELECOM PARISTECH

Security Assurance of Web Services through *Digital Security Certification*

SAMUEL KALUVURI

Academic Supervisor:
Yves ROUDIER
Eurecom Institute

Industrial Supervisor:
Michele BEZZI
SAP SE

April 4, 2016

Dedicated to my parents

Acknowledgements

It is a pleasure to have this opportunity to thank the many people who made this thesis possible.

I would like to thank my PhD supervisor Dr. Yves Roudier for all his help, advice, critical insights and thoughtful comments throughout this journey. I thank him for channeling my enthusiasm in the right direction and for encouraging me to have an ambitious vision towards the thesis right from the day I started the PhD.

I feel very privileged to have Dr.Michele Bezzi as my Industrial Supervisor. He gave me the freedom to select interesting topics while intervening at the right times to ensure that I do not stray away from the big picture. His constant support was instrumental to push my thesis work to various research projects. He was always accessible and made time to discuss any issue I faced during the PhD for which I am very grateful. He encouraged me to collaborate with external partners and facilitated many such collaborations early on in the PhD - again, something from which I benefited immensely. He is one of the most friendly and approachable people I've met and I've learned a great deal from him.

I cannot thank enough my sparring partner, Francesco Di Cerbo, for taking me under his wing and guiding me early on during the PhD. The innumerable brainstorming sessions we had enhanced my understanding of the research topic and scope of my thesis. Working with him on several research papers and setting up collaborations with external partners has been an immense learning opportunity for me. I am very grateful that despite his busy schedule he always made time to review my work when I requested him and gave *pointed feedback.*

A special thanks to my manager, Dr. Jean-Christophe Pazzaglia - he has helped me achieve a lot more than I set out to do, by motivating me, encouraging me and pushing me at various phases during the PhD. I am very grateful to Volkmar Lotz for his insightful comments that, more often than not, helped me look at issues in a totally different light and enhanced my perspective. I would also like to thank Antonino Sabetta, who helped me keep the fine balance between my contributions to the ASSERT4SOA project and the thesis. A huge thanks to Sylvine Eusebi and Gwenaelle Le Stir, who has made my life so much easier by helping me deal the administrative aspects associated with my PhD.

Surely there were frustrating times during the PhD and I feel lucky to have a circle of friends who not only understood the frustration but actually helped me deal with it. Serena Ponta, Giacarlo Pellegrino, Gabriel Serme, Stephane Dupin, Robin Thomas, Alex Egner, Ahmad Assaf just to name a few who helped me get through this PhD with my sanity in check!

I'll be always grateful to my Uncle Satish, for being a huge source of inspiration to me from a very young age, for being a great sounding board, and for always watching out for me. A special thanks to my biggest cheerleader, my sister Serena, for her constant encouragement.

I cannot emphasize how thankful I am to have Tanea for her support, understanding and patience throughout this thesis and especially during the final phases of the thesis.

I would not be in this position if not for the faith and belief my parents had in me. Their support for all my ambitions plans, their encouragement through all my disappointments, and their sacrifices to give me the opportunities to pursue my dreams made this all possible. I cannot thank them enough - I owe everything I am to them. But just for the record - Thank you so very much for everything!!

Abstract

Service Oriented Computing (SOC) has facilitated a paradigm shift in software provisioning models: software is offered as a service – providing enormous benefits to both service providers and consumers. However, a major barrier for a wider adoption of the new service provisioning model in business- and security-critical domains is the lack of security assurance over such service offerings. Security certification, a well established approach in traditional software provisioning models to gain security assurance, can be applied to service environments to provide service consumers with the required assurance.

However, current certification schemes are tailored for traditional software provisioning models where a consumer operates the certified product, static (evaluated at a point in time), and the resulting certificates are represented in natural language. On the other hand, service environments are dynamic with consumers having no control over the service nor its operational environment, and designed to facilitate machine to machine communication. Hence, current security schemes do not scale to service environments, nor can they cater to service specific scenarios such as discovery and composition which rely on automated reasoning.

This thesis proposes the concept of a digital security certificate which is realized by a language to enable security certificate representation in a structured, machine-processable manner. In addition, the thesis presents a framework for the maintenance of the digital security certificates that can cope with the dynamic requirements of service environments. The contributions of this thesis will facilitate the adoption of security certification schemes to service environments.

Contents

List of Figures

List of Tables

Listings

List of Publications

- **Journals**

  ⋆ **S. P. Kaluvuri**, H. Koshutanski, F. Di Cerbo, R. Menicocci, and A. Maña.*A digital security certificate framework for services*. International Journal of Services Computing, 1(1), 2013.

  ⋆ V. Lotz, F. Di Cerbo, M. Bezzi, **S. P. Kaluvuri**, A. Sabetta, and S. Trabelsi. *Security certification for service-based business ecosystems*. The Computer Journal, page 101, 2013

- **Book Chapters**

  ⋆ M. Anisetti, C. A. Ardagna, M. Bezzi, E. Damiani, **S. P. Kaluvuri**, and A. Sabetta. *A certification-aware service-oriented architecture*. In Advanced Web Services, pages 147–170. Springer New York, 2014.

- **International Conferences and Workshops**

  ⋆ **S. P. Kaluvuri**, M. Bezzi, and Y. Roudier. *A quantitative analysis of common criteria certification practice*. In Trust, Privacy, and Security in Digital Business, pages 132–143. Springer International Publishing, 2014.

  ⋆ **S. P. Kaluvuri**, H. Koshutanski, F. D. Cerbo, and A. Maña. *Security assurance of services through digital security certificates*. In Web Services (ICWS), 2013 IEEE 20th International Conference on, pages 539–546. IEEE, 2013.

  ⋆ **S. P. Kaluvuri**, M. Bezzi, and Y. Roudier. *Bringing Common Criteria certification to web services*. In Services (SERVICES), 203 IEEE Ninth World Congress on, pages 98–102. IEEE, 2013.

⋆ **S.P Kaluvuri**, M. Bezzi, A. Sabetta, Y. Roudier, R. Menicocci, V. Bagini, A. Ricardi, M. Orazi. *Applying Common Criteria to Service Oriented Architectures* . International Common Criteria Conference 2012, Paris.

⋆ V. Lotz, **S. P. Kaluvuri**, F. Di Cerbo, and A. Sabetta. *Towards security certification schemas for the internet of services*. In New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, pages 1–5. IEEE, 2012.

⋆ F. Di Cerbo, M. Bezzi, **S. P. Kaluvuri**, A. Sabetta, S. Trabelsi, and V. Lotz.*Towards a trustworthy service marketplace for the future internet*. In The Future Internet, pages 105–116. Springer Berlin Heidelberg, 2012.

⋆ A. Sabetta, M. Bezzi, and **S. P. Kaluvuri**. *Towards a development environment to orchestrate services with certified security properties*. In Proceedings of the 2012 ACM SIGSOFT symposium on Industry Day, pages 1–4. ACM, 2012.

⋆ M. Bezzi, **S. P. Kaluvuri**, and A. Sabetta. *Ensuring trust in service consumption through security certification*. In Proceedings of the International Workshop on Quality Assurance for Service-Based Applications (QASBA), pages 40–43. ACM, 2011

- **Technical Reports**

⋆ H. Koshutanski, A. Maña, R. Harjani, M. Montenegro, **S.P. Kaluvuri**, F. Di Cerbo, E. Damiani, C. Ardagna, M. Anisetti, D. Presenza, S. Gürgens, R. Menicocci, V. Bagini, F. Guida and A. Riccardi. *ASSERT Language V1.2*. ASSERT4SOA Project Deliverable(D1.2), 2012.

⋆ A. Maña, H. Koshutanski,J. González, M. Montenegro, R. Menicocci, A. Riccardi, V. Bagini, F. Di Cerbo and **S.P. Kaluvuri**. *ASSERT Profiles*. ASSERT4SOA Project Deliverable (D1.3), 2012.

⋆ H. Koshutanski, **S.P. Kaluvuri**, A. Maña, M. Montenegro, M. Anisetti, C. Ardagna, E. Damiani, S. Gürgens, D. Presenza. *ASSERT Language V1.4*. ASSERT4SOA Project Deliverable(D1.4), 2014.

⋆ M. Bezzi, S. D’Agostini, **S.P. Kaluvuri**, A. Maña, C. Pandolfo, G. Pujol, A. Sabetta. *Architecture and High-level Design of ASSERT4SOA Framework*. ASSERT4SOA Project Deliverable(D6.1), 2014.

- ⋆ R. Menicocci, V. Bagini, A. Riccardi, M. Bezzi, A. Sabetta, **S.P. Kaluvuri** , G. Spanoudakis, A. Maña. *Framework Requirements for ASSERT4SOA Framework*. ASSERT4SOA Project Deliverable(D7.1), 2011.
- ⋆ F. Guida, V. Bagini, A. Riccardi, **S.P. Kaluvuri**, A. Sabetta, H. Koshutanski. *Report on the identified certificational requirements for ASSERT4SOA*. ASSERT4SOA Project Deliverable(D7.4), 2013.

Chapter 1

Introduction

"Any change, even a change for the better, is always accompanied by drawbacks and discomforts."

— Arnold Bennett

Software provisioning models have undergone a paradigm shift over the past decade – software, now, is offered as a *Service*. Services shield consumers from the complexity of procuring, installing, configuring and maintaining complex software systems on their hardware infrastructure – essentially providing off-premise and on-demand software solutions. Over the last few years, service adoption has grown rapidly, mainly driven by the explosive growth of mobile devices that often rely on services. Service Based Solutions (SBS) such as Gmail [77], DropBox [96], OneDrive [115], and iCloud [108], that have user bases that run into millions, illustrate the popularity of services in today's world.

Though there are many varied definitions of services [4], we consider the broadly accepted notion of services: *Software Services (or simply services) are units of self contained functionality that can be accessed programatically over a network* [137].

Software providers are increasingly offering consumers *Service Based Applications* (SBA), which in effect, are thin clients on the consumer's end that invokes the software, wrapped as a service, running on a remote server and displays the results to the consumer [136]. Basically, SBAs outsource the processing to a remote server and focus on the presentation of the information to consumers. Well known examples such SBAs are web browsers and mobile applications.

There are several reasons for the rapid and exponential rise in the adoption of SBAs. A major reason, though rather an obvious one, is that internet adoption has grown rapidly in the last decade [182]. But a key reason, that gave a huge impetus to the adoption of SBAs, was the introduction of smart phones and tablets and their exponential growth [74]. Smart phones and tablets do not have a lot of computing power when compared to a desktop/ laptop as they run on low power processors and on lower memory chips. They formed a symbiotic relation with services, by outsourcing the processing and focusing on just presenting information to consumers [175]. As consumers started to own multiple devices along with a desktop/ laptop, the need to have data synchronized between these devices is another key reason for consumers moving their data to services or what is more popularly referred as *"the cloud"*.

1.1 Benefits of Service-Orientation

Service-orientation has emerged as an enabler for a range of cloud computing offerings such as Software as a Service (SaaS) [172], Platform as a Service (PaaS) [22] and Infrastructure as a Service (IaaS) [26]. A common characteristic – that is intrinsic to the nature of service-oriented systems, and that is common to all the paradigms mentioned above – is that software deployments involve a number of different organizations, so that no single entity is in control of the whole system.

Service based solutions provide enormous benefits to both consumers and providers. From the consumer point of view, the key advantage is that services are economical as they do not have to procure and deploy the software on their premises leading to huge savings. On top of that, services typically are provided on a subscription basis with the option to scale up or scale down based on demand. Hence consumers can choose to pay for only what they need - once again leading to cost savings. There are also less maintenance issues since consumers do not have to *fix* issues with the software as more often than not service providers ensure that the latest version of the service is available to the consumers that includes bug fixes or patching discovered vulnerabilities.

From the service provider point of view, service provisioning allows them to target a broad range of consumers, for example services such as *GMail* cater to individuals as well as organizations, thereby allowing service providers to enter into new markets. On top of that, they do not have to face issues with various misconfigurations that often happen with ill-informed consumers, since they can ensure that the service is deployed as it should be. This allows service providers to ensure that the service runs as intended. It also reduces the efforts for the maintenance of software since the service providers have control over the operation of the software (in cases of Software as a service).

These are benefits from a monetary perspective, from a technical perspective, services open up unexplored scenarios. Services provide extreme flexibility and applications now can make use of services from different providers to build composite applications and to provide business functionalities to users. Interoperability between organizations, which till recently has been limited to just a few collaborating organizations, is made possible in a very broad manner.

Organizations are embracing this change as well and adopting service provisioning models while moving away from traditional software provisioning models. Organizations such as SAP, Microsoft to name a few have been adopting service provisioning paradigm to deliver their software - both consumption (such as Business Analytics services) and productivity software (e.g., Microsoft Office offered as a service - Office 365 [118]) and in the case of SAP, even ERP solutions [154] are now offered as a service. All this points to the fact, that services are here and here to stay. They no longer are the “next big thing”, they *are* the big thing.

1.2 Security concerns in Service Environments

Technically, service oriented computing is seen as an evolution from modular, component oriented computing, but from a business perspective, it has been quite disruptive, changing the business models, changing the status-quo, lowering the barriers of entry among many other changes. However, this disruption is not quite that pronounced from the consumer's perspective, especially for mobile consumers. This has lead to a situation where consumers have given away too much information to service providers in order to get free/ low-cost services such as *Google*, *Facebook* among others. Such organizations started to use the data collected to generate revenue, by using their information to provide personalized advertisements or localized advertisements. As these services started to gather data about a large percent of the population , Governments around the world started to design and enforce regulations [60, 35, 34] that required these organizations to disclose to the consumers the data being collected. Thus, consumers are becoming increasingly aware of their data being accessed, stored and processed by these organizations. However, with awareness comes concern - on the lack of control [133] over their data, leading to concerns on the security and privacy of their data - whether the service provider is using this data for any commercial or worse - malicious purposes.

The recent scandals be it the National Security Agency (NSA) eavesdropping [14, 64] or hacking of celebrities cloud storage [69] accounts to gain access to their private data have made consumers wary of using cloud services. These incidents, which are becoming more frequent, illustrate the need to secure cloud services.

On the other hand, for end-users, the option to unsubscribe from cloud services is increasingly becoming counter productive, given the tight integration with cloud services that is built-into many popular operating systems. For example, Windows 8 and Windows Phone have a tight integration with Microsoft's cloud storage service OneDrive, while iOS (which runs on the very popular iPhones and iPads) and OS X have a tight integration with Apple's cloud storage service iCloud, and finally, Android (which runs on around 70% of mobile devices) is integrated with GoogleDrive. In an era, where consumers are increasingly owning multiple devices (phones, tablets, laptops/ desktops) the need to have data available and synchronized across all the devices is driving consumers to use these cloud services.

Having said that, service providers *are* trying to ensure the security and privacy of user data is addressed within their services. They are even educating their consumers about the security measures that have been implemented in services [61, 166]. Currently, most service providers have dedicated pages on their websites that discuss their security policies that explain how consumers' information is protected at varying levels of a abstraction. Such measures by the service providers to disclose their security policies, though necessary, are not sufficient to allay the concerns of the consumers. The problem is two fold: *a)* for well known, established service providers - consumers may not doubt the intent of the service provider to secure users' data nor the claims made about the security measures implemented in the service - however, they may have concerns whether these measures are implemented correctly, without any vulnerabilities and that the implemented measures are *sufficient* to provide the required security; *b)* while in the case of new service providers, without an established reputation, consumers not only have concerns whether the service has been implemented without any security flaws, but also they have concerns whether the service provider is non-malicious in the first place - fundamental lack of trust on the intentions of the service provider.

This leads to a undesired scenario where, consumers may trust services offered by well known providers even when they may not be secure, while ignoring the services that maybe secure, but provided by unknown service providers.

1.3 Bridging trust deficit in service consumption

This problem - which, essentially, is a trust deficit on the security of a service - extends to even organizations: when they have to use service based solutions within their organizations, they prefer to use services provided by known providers, or resorting to establish Service Level Agreements (SLA) with certain service providers - to bridge the trust deficit. Service Level Agreements cater to scenarios that are still rooted in traditional software provisioning models, where organizations, that need high level of assurance over the services (or software) they consume, resort to establish SLAs as a means to hold the service provider liable in case of any breach of agreed upon terms and conditions.

However, service oriented computing has created a revolutionary change in business models where services from one provider make use of external services from several other providers leading to a chain of service orchestration. In such scenarios the end user might have to depend on several SLA's to consume a single service.

Let us consider the example of a cloud storage service $\mathcal{S}_1$ used by a client $\mathcal{C}_1$ who establish a service level agreement $\mathcal{SLA}_1$ for the provisioning of the service. However, the service $\mathcal{S}_1$ uses in the back-end two external services $\mathcal{S}_2$ and $\mathcal{S}_3$ to provide *file compression* and *file versioning* functionality respectively and the service provider $\mathcal{SP}_1$ establishes two service level agreements $\mathcal{SLA}_2$ and $\mathcal{SLA}_3$ with the service providers of $\mathcal{S}_2$ and $\mathcal{S}_3$ respectively. When the client $\mathcal{C}_1$ is using the service $\mathcal{S}_1$, she is also making use of the services $\mathcal{S}_2$ and $\mathcal{S}_3$ indirectly. However, the SLA that the client has is only with $\mathcal{SP}_1$ for the service $\mathcal{S}_1$ and so, she cannot be sure that the services $\mathcal{S}_2$ and $\mathcal{S}_3$ do not violate the terms that were agreed with $\mathcal{SP}_1$. The example only goes to show the scalability issues and infeasibility of Service Level Agreements as a primary source of *trust* in service based software provisioning models. Furthermore the assurance gained from SLAs is *a posteriori* and based on the fact that in case of any violation of the agreed terms and conditions, the service provider can be held liable. A critical presumption, in these scenarios, is that any violations to terms and conditions are *detected* and *provable*. However, in service environments, since consumers do not have control over the service or its operational environment, it becomes harder for consumers to *detect* such violations. Hence SLAs, though essential, cannot alone provide the needed assurance in service environments.

Another means for consumers to gain trust on the security of a service is by having a trusted third party verify and validate a service and attest that the service meets certain security properties. This is similar to a security certification process where the trusted third party is, usually, the certification authority. This is a scalable solution for service based ecosystems as it address the security assurance issue at a modular level, there by, allowing composite applications to be built using several "certified " services.

There are several security certification schemes in practice today such as Common Criteria for Information Technology (CC) [45], Commercial Product Assurance (CPA) [17], Certification Sécurité Premier Niveau (CSPN) [2] and so on. Common Criteria (CC) is the most popular, used and recognized scheme among the existing ones. There are more than 1900 products [44] that are certified through the Common Criteria scheme. However, upon careful consideration of the certified products through Common Criteria and the other existing certification schemes, it is noticed that no service based solutions have security certifications. Given the popularity and widespread usage of service based solutions and the security concerns of service based solutions, it is important to understand the reasons behind this apparent lack of security certified service solutions.

We have examined the security certification schemes currently used in order to understand the reasons for the lack of interest from service providers to undergo security certification.

1.4 Problem Statement

Some of the key limitations for adapting security certification schemes to service environments are listed here.

(a) One of the primary reasons for the limitation of existing certification schemes in service environments is the fact that they were all tailor-made for traditional software provisioning models. Service provisioning models bring with them a complexity in providing security assurance to consumers. Services run on an infrastructure that is partly controlled by the service providers and in some cases the service provider does not have *any* control over them. In either case, the consumer has no control over the operation of the service nor any control over the execution environment of the service. This is a key departure from traditional software provisioning models, where the current certification schemes delegate some responsibility to the consumer for the "secure operation" of the certified product.

(b) Another key challenge is that current certification schemes are system wide certifications - where they certify large IT products, these products have a lengthy development cycle and follow the traditional waterfall [24] models during their development and these products are updated in a planned, phased manner. However, with services this scenario doesn't hold water since they are are dynamic, built using agile methodologies, updated frequently on a daily/ weekly basis either to provide more features or to fix any bugs that were found or to patch any vulnerabilities. And such changes to certified products typically invalidate the certification gained which raises concerns on the practical usefulness of the security certification in the first place.

(c) Services are modular and they typically involve multiple parties during provisioning and consumption - and traditional certification schemes do not cater to such scenarios as they assume that the vendor of a certified product is the owner of the whole product that is being evaluated [57].

(d) Lastly, service oriented architectures rely on automated reasoning for service discovery, service binding and service composition (in some cases) and in such scenarios traditional security certifications do not seem to cope well, since they are mostly represented in human readable documents making them unusable in situations where could be essential [25].

1.5 Research Question

The limitations mentioned in the previous section shapes the research goal for the thesis:

How can we apply security certification schemes to service environments?

In order to achieve the main goal of the thesis, there are a couple of key questions that must be answered. As explained in Section 1.4, service provisioning is very different from traditional software provisioning models. In addition, the service development methodologies also differ starkly from traditional software systems development methodology. Hence, an important question that needs to be answered is:

How should the certification processes be modified to bring them into service environments that are dynamic and modular?

The *certification processes*, mentioned in the above question, encapsulate both the production and the maintenance of the security certifications. However, certification maintenance is a key aspect, given that services run remotely and are not under the control of the service consumer.

Another key question that needs to be answered is:

How can security certificates participate in typical service scenarios such as service discovery and service composition?

This question focuses on the last point mentioned in Section 1.4, that is, how should security certificates be formulated and represented that they allow automated reasoning to be performed on them. Security certifications that allow automated reasoning to be performed on them can have a profound impact not only on the service security certification processes, but can change the way services are consumed, similar to how identity certificates have facilitated how we transmit and consume information from remote servers.

1.6 Thesis Contributions

In order to answer the research questions, the thesis focuses on the *production*, *maintenance* and *consumption* of security certifications for services.

In this regard, the key contributions of the thesis can be summarized as follows:

(a) A thorough analysis of the state of the art on security assurance landscape (Chapter 2).

(b) A critical analysis of the current security certification schemes to understand the limitations and drawbacks that prevent them from being applied in service environments (Chapters 3, 4, 5)

(c) A machine processable representation of the security certificate that allows automated reasoning and thus, facilitates the usage of security certificates in service scenarios such as discovery, binding, composition among others (Chapters 6, 8)

(d) A dynamic security certification life-cycle, that primarily focuses on the maintenance of the security certificates in service landscapes that focuses primarily on the providing security assurance over the operational environment through monitoring(Chapter 7)

The thesis does not contribute towards the *criteria* that should be used during evaluation of services. As security certification criteria is not a uniform concept, as each country or each domain has their own certification schemes and hence to propose an overarching scheme is counter- productive as it can be too restrictive. Instead the thesis focuses on generic mechanisms that should be used *within* the certification processes, primarily towards the generation and maintenance of certificates, that will facilitate various schemes to adapt to service environments.

The contributions of the thesis are designed in such a way that they can be incorporated into the various certification schemes. This flexibility allows the contributions to have a greater and wider impact on the security certification landscape.

1.7 Thesis Structure

The reminder of this thesis is structured as follows:

- **Chapter 2** presents the different approaches towards gaining security assurance in traditional software provisioning models, explaining how security certification provides a scalable approach for gaining security assurance. Further, an overview of the various certification schemes is also presented with a brief discussion about each scheme.

- **Chapter 3** presents an detailed analysis of the Common Criteria certification scheme, which is one of the most popular and widely recognized security certification scheme.

- **Chapter 4** presents a quantitative analysis of the Common Criteria (CC) certification *practice* to examine whether the current CC practice stays true to the intentions of the CC scheme and discusses approaches to improve the CC practice.

- **Chapter 5** analyses the various approaches to gain security assurance in Service Environments, and the limitations of the current security certification schemes.

- **Chapter 6** proposes the conceptual model of a Digital Security Certificate for services, and an XML-based language which realizes the conceptual model. It also introduces the concept of a Digital Security Certificate profile, which facilitates an easier adoption by the various certification schemes by imposing constraints on the Digital Security Certificates.

- **Chapter 7** discusses the various trust relationships existing between entities in a service environment and the implications they have on the service's security certificate lifecycle. A dynamic security certificate maintenance framework is proposed that increases consumer's assurance over the service's operational environment.

- **Chapter 8** presents the exploitation of the concepts that have been proposed by the thesis in the industrial context and also discusses the impact of the thesis's contributions on the security certification landscape.

- **Chapter 9** concludes the thesis and provides future directions.

Part I

Security Certification Landscape

Chapter 2

Assurance through Security Certification

"In our reasonings concerning matter of fact, there are all imaginable degrees of assurance, from the highest certainty to the lowest species of moral evidence. A wise man, therefore, proportions his belief to the evidence ."
— David Hume, An Enquiry Concerning Human Understanding

In this chapter, we explain in detail the need for security *assurance* of software and the various means through which security assurance can be gained. We focus on *security certification of software* and explain the advantages of this approach to provide consumers the required assurance in a scalable manner. Finally, we discuss the various security certification schemes currently used in practice.

2.1 Secure software

The terms "secure" or "trusted" are used quite liberally by software vendors while advertising their products. However, in reality, software products are seldom secure even when the product vendors implement the best state of the art security measures to protect it. This is due to the mere fact that threat environment for software is dynamic [39] - there are always people trying to bypass the best security measures put in place to steal information, cause damage to information or the information system itself, tamper the information or the functioning of the information system itself - or just because they enjoy the challenge of bypassing the "secure software" [92]. Hence, it is important that the product is updated with the latest fixes or patches to prevent the product from being exploited by attackers.

More often than not, product vendors do not implement *all* the necessary security measures required to counter the threats. In such cases, the task of an attacker becomes far more simpler, clearly, as there are loopholes through which the overall security measures of the software system can be bypassed. Hence, it is important that the product possesses the required security measures to counter the identified threats.

Another important issue is that security measures implemented by product developers may contain implementation flaws [93] or more popularly referred as "*bugs*", and if they go undetected in the testing phase, these bugs end up in the released software. Security measures that are flawed become very critical threats to the security of the software. As an attacker who is successful in exploiting a vulnerability not only bypasses the security measure, but can in turn gain inside access to alter the functioning of the rest of the security measures.

Finally, threats to software systems are not only due to active or passive attacks by determined people, they can arise due to the software having a poor design leading to the system failing or malfunctioning [143], causing data loss, unintended data modification or data disclosure to unauthorised parties. So it is of vital importance that security measures are free from design and implementation defects.

Security of a software, not only depends on the security mechanisms, design and implementation correctness of the software product but also on the operational environment of the product [103] as shown in Figure 2.1. Operational Environment (OE) or a software product is composed of two aspects: IT related and non-IT related.

IT Operational Environment for a software product is the dependencies that the software has on other software/ firmware/ hardware components. For example, a secure operation of a database product depends on the Operating system that it runs on. Hence, software products must identify such dependencies on other components in the IT infrastructure.

Non IT Operational Environment for a software product deals with the security management processes employed in the Operational Environment. For example, no matter the strength of the security mechanisms that are implemented within a product, if the authorized users of a software system are malicious - the security of the system is compromised. Hence, organizations should follow best practices in assigning access rights to individuals, check the background of employees who have access to certain sensitive information and so on. As shown in Figure 2.1, Non-IT Operational Environment consists of two aspects - *Physical Measures* which deal with securing

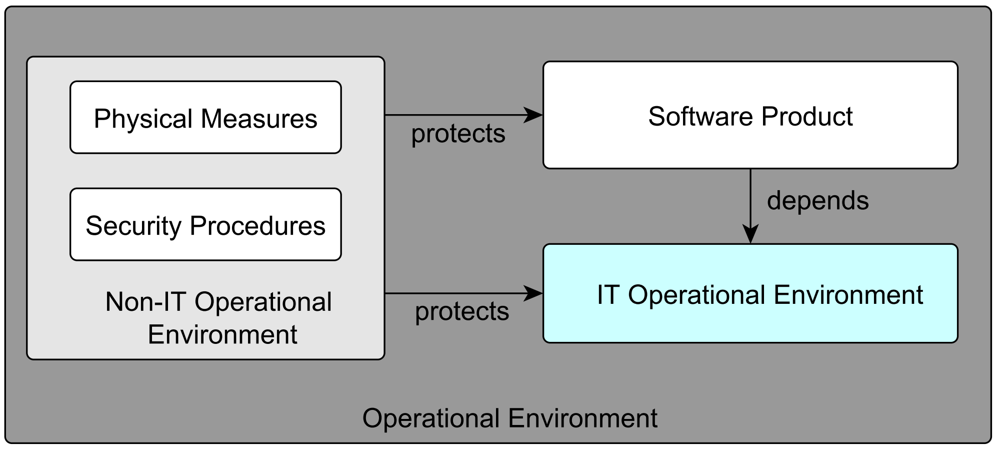


Figure 2.1: Software Product and its Operational Environment

the premises in which the software product operates (perimeter security, background checks of employees etc.); and *Security Procedures* deal with the aspects related to security management of the system, such as proper configurations of components, assigning proper access control policies.

Hence, they key requirements for a product to be secure are:

- It is constantly updated with respect to evolving threats
- It has sufficient security measures implemented to counter the threats identified
- Software is free from design and implementation flaws that can affect the parts of the system, which in turn might affect the security of the overall system.
- Software dependencies for secure operation on other IT components in the Operational Environment must be satisfied
- Security management process, related with non-IT aspects should be enforced

In other words, the software provider is responsible for the *security* of the product, while the software consumer is responsible for the *secure operation* of the product.

However, during procurement of a software, consumers typically focus on the security of the product and even when software providers claim that the software is secure (against certain identified threats), consumers still have concerns using it. This is due to the trust deficit that exists between software vendors and consumers. Consumers need some sort of *assurance* to underpin the claims made by software vendors about the security of the software.

2.2 Assurance of secure software

Typically, trust deficit for consumers arises when they have to procure and use third party software for their IT needs. Large organizations have in-house testers [122] to evaluate a product before it is used within their organizations. Such testing primarily focuses on *black box* approaches, given that they do not have access to the source code of the third party software.

In addition, any new updates to the software have to undergo the testing process. However, as the number of third party software used within an organization increases such in-house testing solutions become infeasible for organizations, since the costs associated of maintaining teams to cope with different software is too high.

Hence, many large organizations enter into *Service Level Agreements* with third party software providers. Service Level Agreements offer a level an acceptable level of trust for software consumers, as they can hold the software provider liable in case of damage caused by the software product.

2.2.1 Service Level Agreements

Service Level Agreements (SLAs) are widely used as a means to establish trust between the software providers and consumers. Typically, through SLAs consumers require the software provider to agree with certain obligations such as providing updates to the software to fix any security flaws that might arise in future or provide guarantees that the software is free from security flaws etc. Typically, SLAs are used as a means to hold the software provider liable for any damages caused by the security issues (among other aspects) of the software. However, they do not imply that the software is secure and are used primarily to hold the software provider liable *after* a security flaw is detected [123] - but, in cases where the consumers need *a priori* assurance over the security of the software, SLAs cannot be applied.

In addition, SLAs, in general, tend to be focused primarily on Quality of Service (QoS) aspects rather than security and tend to be less technical and focus on auditable *QoS* properties. Although certain security measures are guaranteed within the SLAs, they do not explain the security measures implemented within the product nor can they provide assurance that the product is free from implementation flaws, which can affect the security of the software (security bugs). Moreover, SLAs are criticised [168] for being more favourable towards the software providers than consumers, given that they are crafted by the software providers who try to hide themselves as much as possible behind a lot of legalese and loopholes.

In cases where the software consumer is a regular end-user, it is highly unlikely that a single consumer can craft a service level agreement with the software provider and thus, in many cases ends up “agreeing” to the “terms and conditions” of the software vendor. Many times, even when the consumer suffers damages due to breach in security of the product, they cannot hold the software provider liable, because they lack *proofs* to hold the software provider liable for the breach.

In the case where consumers are big organizations, they tend to enter into very complex service level agreements that involve maintenance of the product, timely updates to fix any security flaws in the software. This leads to a unintentional *lock-in* with the software provider [109], that is, the organisation’s IT infrastructure is tailored for that specific software and any changes to the IT infrastructure to use another software has a considerable cost attached to it.

2.2.2 Security Certification

Another means to achieve security assurance is through certification of software [57]. Security certification typically involves a third party, usually the Certification Authority (CA), trusted both by the software consumer and the provider, that evaluates the software based on certain criteria and consequently certifies that it possesses certain security features. The consumer can trust claims made by the CAs and gain the required security assurance. At a very high level security certification schemes can classified into two broad categories: *process based* and *product based* [66].

Process based security certifications typically focus on the Non-IT operational environment of the software. They verify whether the organization has proper security management policies, physical security measures put in place and are operated correctly. They typically provide *a posteriori* assurance, as in, they can verify whether there has been any violations that occurred in the past. Schemes such as *ISO 27001* [91], *SAS 70* [155] fall under this category as well as audit based certifications.

On the other hand, *Product based certifications* allow consumers to gain assurance *a priori*, that is, the consumer can know before-hand whether the certified product meets their security requirements and that the software has been implemented without any flaws. It also relieves the software providers and consumers to establish SLAs with each other, given that they can trust the statements of the vendor that are underpinned by the certification issued by the CAs.

2.2.3 Assurance through Process Security Certification

Security of a product is closely tied to its operation. The *users* of the software system, the *usage* of the software system, the environment in which the software system operates all contribute towards the overall security assurance a consumer gains from the software. Similarly, the development processes of the software also impact the security assurance.

In order to address these aspects, there are several successful security certification standards that are used currently. Though process security certification is *not* the focus of the thesis, in this section, we briefly explain a few process security certification schemes in order to give an overview of the security certification landscape.

ISO 27001:2005

This is an international standard proposed by ISO. It specifies the requirements to establish, implement and maintain an information security management system within an organization. It also contains requirements for assessment of information security risks tailored to the needs of the organization. The requirements that are specified in ISO 27001 standard are rather generic, in order to cater to organizations of all types, sizes and nature [91]. Typically, as part of this certification processes organizations declare *security controls*, which are basically security processes, that are implemented within the organization which mitigate the risks identified. Examples of a few *security control* objectives in the ISO 27001 standard are:

Objective – ***A.8.1 Human resources security prior to employment:*** "*To ensure that employees, contractors and third party users understand their responsibilities, and are suitable for the roles they are considered for, and to reduce the risk of theft, fraud or misuse of facilities*" [91].

Control – ***A.8.1.1 Roles and responsibilities:*** "*Security roles and responsibilities of employees, contractors and third party users shall be defined and documented in accordance with the organization's information security policy* ." [91].

Control – ***A.8.1.2 Screening:*** "*Background verification checks on all candidates for employment, contractors, and third party users shall be carried out in accordance with relevant laws, regulations and ethics, and proportional to* *the business requirements, the classification of the information to be accessed, and the perceived risks.* " [91].

It is evident that the scope for ISO 27001 includes both the *Physical Measures* and *Security Procedures* that are implemented within an organization. This standard has been cancelled and replaced by the ISO 27001:2013 standard [33], although the nature and scope still remain the same.

Statement on Auditing Standards No. 70

Statement on Auditing Standards No.70 [155], more commonly referred as *SAS 70*, is a standard developed by *American Institute of Certified Public Accountants*. SAS 70 provides a set of guidelines that need to be performed by an auditor to issue an opinion about the controls that are implemented within an organization. There is a criticism [73] that SAS does not contribute towards the security assurance, as it is a part of an overall compliance process for financial reporting (Sarbanes-Oxley Act (SOX) [164]) and thus cannot be treated as a security certification process as such. Since 2011, this standard has been replaced by *Statement on Standards for Attestation Engagements No. 16* (SSAE 16) [169].

2.2.4 Assurance through Product Security Certification

Security assurance of software products has always been an important aspect [28], especially in critical domains such as defence, military, aviation, healthcare and finance. Since the software consumers in these domains are not security experts, nor can they rely *only* on the statements and assurances of the software vendor - there was a need for an independent, trusted third party to evaluate the software and certify that it meets certain security features.

Governmental bodies were [124] , and still are (seen collectively), one of the biggest consumers of software products and since data that is processed within those departments can be extremely sensitive (such as in the case of military), governments drafted guidelines for software products that are used within their departments. However, in order to verify that software vendors were meeting those guidelines the products had to be evaluated. Thus, these guidelines evolved into "criteria" against which products were evaluated. Software procurement policies in governmental organizations required the products to have passed the evaluation based on these criteria [173].

However, with an ever increasing number of software vendors and products, governments needed to find a scalable approach. Hence they created centralized “certification” bodies that were dedicated to evaluate products and consequently certify whether they meet the criteria. The certification bodies were responsible for designing and managing the overall certification scheme. The certification bodies authorized several laboratories (accredited evaluation labs) to perform the actual evaluation of products. Software vendors who wanted to provide their products to governmental organizations had to pay these evaluation labs to get their products certified [8]. This model was successful when software products were custom built for specific use.

With the advent of generic software products especially in categories such as Operating systems, Databases, productivity software, this approach was no longer a viable solution for product vendors as: *a)* It was not possible to build products that meet the criteria of every country; *b)* The costs of getting a product certified in each country was impractical for product vendors.

In order to resolve this issue, certification bodies across various countries spent significant efforts to harmonize the certification criteria. One of the first significant results in this direction was the Information Technology Security Evaluation Criteria (ITSEC) [52] which harmonized [144] the security evaluation criteria from United Kingdom, Germany, France and Netherlands.

The United States and Canada still had their own criteria namely the *Trusted Computing Security Evaluation Criteria* (TCSEC also known as “Orange Book”) [171] and *Canadian Trusted Computer Product Evaluation Criteria* (CTCPEC) [107] respectively which were considerably different from ITSEC [30] in terms of specification of security functionalities and the assurance levels provided by these schemes.

Moreover, the application of these schemes was still limited to governmental organizations with limited success [111], however, as software adoption was increasing in non-governmental sector and consumers required to gain security assurance of the software products, the existing certification schemes could not be applied as they proved to be too restrictive. Hence, there was a need for a generic software certification scheme.

In order to provide a unified criteria for security evaluation, the ISO brought the various certification bodies together to combine and harmonize the evaluation criteria which resulted in the “Common Criteria for Information Technology Security Evaluation” (referred as CC henceforth) [45]. CC is an international standard (ISO/IEC 15408) and is recognized in 26 countries [50]. It is the most popular, used and recognized IT security certification in the world.

Though CC is widely applicable and recognized, it is also expensive and time consuming, hence there were several specialized domain-specific, lightweight certification schemes that have been developed over the years. In addition, there have been country specific variations of the CC scheme. But CC scheme can be seen as the most broadest of all product based security certification schemes. The various certification schemes and approaches are discussed in detail in Section 2.3.

Vetting Processes in Software Marketplaces

Recently, we have seen software marketplaces emerging, primarily driven by the explosive growth in mobile devices, as a facilitator between software consumers and providers. Given that most marketplaces are operated by organizations that also provide the operating system on which the software is executed, marketplace operators have a stake in the security of the software admitted to their marketplaces.

In this regard, marketplace operators can adopt different approaches to deal with security while delivering applications to end users: in particular, Barrera and Van Oorschot [19] propose three categories, “Walled garden”, “Guardian” and “User control”; they range from a rigorous assessment of any applications on the market, to a completely open model, where security checks are upon user’s responsibility. They also propose a classification of *vetting tests*, which can also be seen as “Lightweight certification”, for applications to be advertised on a (mobile software) marketplace. The seven categories are: “smoke tests”, “hidden-API checks”, “functionality checks”, “intellectual property, liability and terms-of-service checks”, “UI checks”, “bandwidth checks”, and “security checks”.

We present in Table 2.1 a number of relevant marketplaces, together with their publicly disclosed security assessment criteria.

Market name	Code reviews	Architectural review	Hands-on assessment	Periodic security review	Application Removal
Salesforce AppExchange	No	Yes	Yes	Yes	Yes
Google AppsMarket	No	No	No	No	Maybe
Windows Azure Marketplace	No*	No*	No*	No*	Maybe
Apple App Store	No*	No*	No*	No*	Yes
Android Market (Play)	No	No	No	No	Yes
Windows Store	No	Yes	No	Yes	Yes
Nokia Store	No	No	No	No	Yes
BlackBerry App World	No*	No*	No*	No*	Yes

Table 2.1: Security Features Of Existing Software Markets. Information marked with ‘*’ are not completely publicly disclosed by providers

Salesforce releases a customer relationship management (CRM) system on the cloud that has a number of companion tools. It permits third-parties to publish and advertise their applications (or extensions to existing

Salesforce applications) that can operate on customers' data and information, on a specific marketplace with defined security review policies [148]. Google Apps Market is a store where third-parties can advertise complementary services for Google Apps services. Google explicitly informs its customers that no security checks are conducted on advertised applications [75]. Windows Azure Market is the official marketplace for Windows Azure (Platform-as-a-Service). Third parties can advertise their services, that apparently are not verified by Microsoft [113]. Existing marketplaces adopt the previously-described "User Control" approach.

Apple's App Store, instead, can be seen as an example of the "Walled Garden" approach, meaning that anything that runs on the mobile devices(iPhone & iPads) must be explicitly approved by Apple. The app review process is not publicly disclosed; in a response to a FCC request in 2009, Apple disclosed some information[12], that are contained in Table 2.1. Microsoft offers Windows Store [114] to users of its Windows Phone OS. Application publishing and review process is documented in MSDN [112], the reference guide for any development effort with Microsoft technologies. Also Nokia has a specific certification process for publishing applications on its market [125], the Nokia Store [126]; nevertheless, newer Nokia's Windows mobile phones should follow Microsoft guidelines. RIM's App World is the reference software market for BlackBerry devices. Almost no public information on security assessment could be found, except those contained in [145].

In summary, where applicable, none of the above marketplaces discloses:

- the details of its security assessments, or
- the results of the vetting process for each applications.

This means that users have to cope with a "one-size-fits-all" definition of security, like in the majority of cases, having no option but to trust blindly marketplaces' procedures; or they have to face the absence of security assessments, having no option but to trust third-parties. Moreover, marketplace operators, who assume the role of Certification Authorities, control the execution environment of the services or mobile applications, which may not be the case for generic security certification schemes. However, the vetting processes adopted by the various marketplace operators can be seen as a lightweight, scalable, custom certification schemes.

2.3 Product Security Certification Schemes

There are several product security certification schemes that are currently in use. At a high-level, these schemes can be classified along the following dimensions:

- Applicable product types
- Certification scope
- Evaluation Criteria against which products are certified
- Assurance Levels based on the evaluation results

Product types identify the range of products that can be certified under a certification scheme. IT products can range from hardware, firmware to software and typically the evaluation methodologies and requirements specification can vary for each category of products. Hence, certification schemes explicitly identify the IT product types that can be evaluated under their scheme.

Certification scope specifies the aspects of the software that will be under the scope of evaluation. For example, the CC scheme considers threat modelling, that is, it does not to be outside the certification scope. It also specifies that it does not certify the operation of the product, but focuses solely on security of the product from its design, development to delivery of the product.

Evaluation criteria specify the methodologies to evaluate a security functionality of a product. Defining the evaluation methodologies is important given that in many cases, there is no consensus between security experts on which methodology is better. Hence, after careful consideration the certification bodies select specific methodologies for evaluation of a product. These methodologies contribute towards the criteria that lie at the heart of each certification scheme. In addition to the evaluation methodologies, the evaluation criteria specifies how various parts of the system must be identified, represented and consumed.

Security assurance is a multi-dimensional property. It is based on

a) the trust that the consumer has on the certification bodies (including the evaluation labs)

b) the number of security features implemented within a certified product;

c) the rigour of evaluation performed by the evaluation labs.

Current certification schemes assume that the consumers trust the certification bodies and focus on the other two aspects: security features and evaluation rigour.

Security features implemented contribute towards the assurance gained by consumers. While basic security features are necessary, a high number and level of security features goes a long way to establish high level of trust on the software product. Hence some certification schemes allow products to be certified for various levels.

Finally, security features in a product may be evaluated using different methodologies such as formal analysis or security testing. Formal analysis of security features of software products yield higher assurance when compared to products that have been security tested. Even within a certain methodology, there are several metrics such as coverage in the case of security testing, that will impact the assurance gained by consumers.

Since these are rather detailed aspects resulting from the certification process, understanding such details is not trivial for regular consumers. Hence, certification schemes prescribe ***Assurance Levels*** using one or more dimensions presented before. This is in order to distinguish products based on the strength of security functionalities or based on the rigour of evaluation to allow consumers to compare different certified products.

Such layered approach by certification authorities allows products to be certified at varying levels of assurance. However, consumers need to have a proper understanding of the certification scheme (scope, criteria, levels) to understand whether the certified product's claims meet their assurance requirements and the *Assurance Levels* should be interpreted within the proper context specified by the certification scheme.

2.3.1 Current Security Certification Schemes

As part of our analysis, we have examined the following security certification schemes:

- Common Criteria [45]
- Certification Sécurité Premier Niveau (CSPN) [2]
- Commercial Product Assurance [17]
- FIPS-140-2 [120]
- FIPS-201 [121]
- McAfee Secure [110]

Certification Scheme	Software	Firmware	Hardware
Common Criteria	Yes	Yes	Yes
CSPN	Yes	Yes	Yes
CPA	Yes	Yes	Yes
FIPS-140	Yes	Yes	Yes
FIPS 201	Yes	Yes	Yes
McAfee Secure	Yes	No	No
Vetting Processes	Yes	No	No

Table 2.2: Type of products that can be certified under various schemes

- Vetting processes employed by software marketplaces [12, 114, 126]

The selection of these schemes was not an arbitrary choice, rather we choose each scheme based on its visibility, reputation and recognition. Even though some of these schemes such as *vetting processes* and *McAfee Secure* are not, strictly speaking, certification schemes, we have included them in our analysis to have a broader picture of the software certification landscape.

Based on the different aspects we identified before, we tried to classify these schemes in order to understand the differences among them.

Product Types

Figure 2.2 shows the product types that can be certified through the various schemes. It is clear that Common Criteria and schemes that are based on Common Criteria, such as CSPN, CCP-Mark are applicable to all the different IT product types. The FIPS standards are also applicable to all the product types. However, such a high level classification can be misleading given that in many cases, some of the certification schemes have specialized security domains.

Hence, we further examined the schemes more closely to understand the security domains in which these schemes are applicable. Figure 2.3.

Crypto Security	Mobile Security	Anti-Malware	Content Security	Network Security	IC/ SC Security	Auditing	Identity Management
Common Criteria (CC)							
Commercial Product Assurance (CPA)							
Certification Sécurité Premier Niveau (CSPN)							
FIPS 140	-	-	-	-	-	-	-
-	-	-	-	-	-	-	FIPS 201
-	-	McAfee Secure	-	-	-	-	-
-	Vetting Processes			-	-	-	-

Table 2.3: Applicable Security domains for the various certification schemes

We can find out that schemes based on FIPS standards, though have the broadest product type range, are applicable in just one specific application domain. While the CC scheme and the schemes based on CC have the broadest security domain range.

Certification Scope

Since all the certification schemes we have examined are product based certification schemes and hence, all process related aspects are outside the scope of these certifications. Specifically, for Common Criteria and its offshoots, threat specification is considered to be outside the scope as well as security policy specification. This implies that the certification authorities do not evaluate whether the threats, against which are a product is certified, are realistic or pragmatic.

In other words, a vendor can choose to be certified against minor threats and pass the certification. This can lead to a scenario where the certification stamp is used for marketing purposes, while the product may not be necessarily providing any realistic security at all.

Evaluation Criteria

CC and CC-based schemes have standardized evaluation criteria that specify the methodologies that must be followed during a product's evaluation. FIPS standards, specify the functional criteria that must be met by the products but do not specify any standardized evaluation criteria. Since we do not have access to the evaluation processes of *McAfee Secure* and the custom vetting processes implemented in the software marketplaces, we cannot ascertain on the evaluation criteria of these schemes.

Assurance Levels

CC allows products to be certified at varying levels of assurance. These levels are based on the rigour of evaluation of the product. These levels are referred as *Evaluation Assurance Levels* (EALs), that range from EAL1–EAL7, with EAL7 being the highest. CPA, though based on Common Criteria, has only one level of assurance - *Foundation Grade* [17]. CSPN also has a single level of assurance.

FIPS-140 defines four levels of security - Level 1 to Level 4, however these levels are based on the strength of the security functionalities that are implemented within the product - this is in contrast with the CC schemes, where the assurance levels are based on evaluation rigour.

McAfee Secure and marketplace vetting processes typically have a single level of assurance.

Conclusion

In this chapter, we have identified the requirements for "secure software" and explained the need for assurance over secure software. We have examined the various means through which assurance can be gained, and how security certification is a scalable means to gain the required security assurance. We present the various certification schemes, focusing on product based security certification. We examine the different product security certification schemes and deduce that Common Criteria security certification scheme is the most widely applicable, recognized and used certification scheme. Hence, we present the CC scheme in much more detail in the next chapter.

Publications

The contents of this chapter have been discussed in the following publications:

- ⋆ V. Lotz, **S. P. Kaluvuri**, F. Di Cerbo, and A. Sabetta. *Towards security certification schemas for the internet of services*. In New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, pages 1–5. IEEE, 2012.

- ⋆ F. Di Cerbo, M. Bezzi, **S. P. Kaluvuri**, A. Sabetta, S. Trabelsi, and V. Lotz.*Towards a trustworthy service marketplace for the future internet*. In The Future Internet, pages 105–116. Springer Berlin Heidelberg, 2012.
- ⋆ A. Sabetta, M. Bezzi, and **S. P. Kaluvuri**. *Towards a development environment to orchestrate services with certified security properties*. In Proceedings of the 2012 ACM SIGSOFT symposium on Industry Day, pages 1–4. ACM, 2012.

Chapter 3

Common Criteria Certification Scheme

"All for one and one for all"

— Alexandre Dumas, The Three Musketeers

Among the current security certification schemes for software, Common Criteria is the most widely used [44], recognized [50] and is applicable to a broad range of product categories(Section 2.3.1). In this chapter, we present the evolution of the CC scheme from different national schemes and explain in detail the certification process as well as the various documents (certification documents) that result from the CC certification process.

3.1 Evolution of CC Scheme

One of the most important aspects of the CC scheme is the international recognition that it enjoys. It is recognized in 26 countries, thus providing immense value to product vendors by getting a product certified once and having it recognized it around the world. But the CC scheme is the result of years of work by various governmental bodies around the world to harmonize the criteria to evaluate IT products. Here we present the different schemes that are part of the evolution of the CC scheme.

3.1.1 Trusted Computer System Evaluation Criteria (TCSEC)

The Trusted Computer System Evaluation Criteria (TCSEC) or more commonly referred as the "Orange Book" was designed by the US Department of Defence for software products that are used within the US governmental and military organizations. It can certify products that range from software, firmware to hardware. It aims to provide a standard for product vendors to gain an insight into the security features that must be implemented within their products in order to provide products that satisfy trust requirements [171].

The criteria for the TCSEC requires a security policy to be defined that is explicit and defined in a structured manner. The security policy should be enforced by the mechanisms implemented within the software. The various aspects of the system that are evaluated should be clearly marked and uniquely identified. It requires that the product comes with supporting documentation regarding the design, specification, test documents and user guides. It also requires the audit data should be recorded and protected against manipulation in order to ensure accountability.

It has four main keys of assurance ranging from *A* to *D*, with *A* being the highest assurance that can be provided.

- **D: Minimal Protection** is for systems that have been evaluated but fail to pass the evaluation and have no specific security characteristics that meet the criteria.

- **C1:Discretionary Protection** offers minimal assurance and requires the products to have implemented Discretionary Access Control mechanisms. There should be a separation between users and information, and identification and authentication mechanisms should be present in the product.

- **C2:Controlled Access Protection** it should meet the requirements for **C1**. It further requires the products to have auditing capabilities for tracking the actions (such as access requests) of the product users. It also requires the product to undergo stringent testing.

- **B1: Labelled Access Protection** it should meet the requirements for **C2** and in addition should implement Mandatory Access Controls (MAC) for specific sensitive resources. Each sensitive resources should have a classification label and each subject must have a clearance label. The security policy for this level requires an informal statement and the design specifications are reviewed and verified.

- **B2: Structured Protection** it should meet all the requirements for *B1*. In addition, it requires the security policy of the the product to be clearly defined and documented. The system design and implementation are subject to a more rigorous evaluation process. It requires more stringent authentication mechanisms to be implemented with the product. The system is also verified to ensure that there are no covert channels.

- **B3: Security Domain** requires the product to meet all the requirements of *B3*. In addition, the reference monitor that is implemented within the product should be tamper-proof and is tested thoroughly. It checks whether the system can be recovered after a failure without compromising the security.

- **A1: Verified Design** requires the product to meet all the requirements of *B3*. And in this class of assurance, formal techniques are used to prove the equivalence of the security mechanisms that are implemented in the system to the security policy model for the system. It also places stringent controls over the change management for the development of the system. This offers the highest assurance in the TCSEC scheme.

A key limitation of this scheme is that it was designed for operating systems though some of the criteria has been adapted and interpreted for other product categories. It has a heavy focus on Bell-LaPadula model, however for most commercial products they do not need the Bell-Lapdula model based access control. It also focuses primarily on data integrity and availability, which are very important attributes for organizations. It combines security functionalities and assurance requirements in a single scale.

3.1.2 Information Technology Security Evaluation Criteria (ITSEC)

This was developed by the European Union to address issues related with security evaluation of products. It decouples the security functionalities such as access control, auditing etc., from the security assurance attributes such as the methodologies used to evaluate the security functionalities such as formal proofs, security testing etc.

The main goal for ITSEC [52] was to harmonize [144] the different evaluation criteria that were designed in United Kingdom, Germany, France and Netherlands. However, it goes beyond harmonizing the criteria in these countries and in fact extended the criteria of these different schemes. It provides a compatible basis for security certification of IT products by national authorities in these countries which are recognized by the other cooperating countries.

The ITSEC scheme allows products vendors to choose aspects of the system that will be evaluated, this is referred as *Target of Evaluation (TOE)*. The security features implemented within the TOE are specified in a document called *Security Target(ST*. The ITSEC scheme mandates that the security target contains the following aspects:

1. **System Security Policy (SSP) or Product Rationale:** When the TOE is a system, SSP specifies the set of rules, laws and practices that regulate the management, protection and distribution of sensitive information. When the TOE is a product, the product rationale provides information to a prospective purchaser of the product whether the product meets the security objectives of their IT environment.

2. **Security Objectives:** They specify, at a very abstract level, the security functionality that is desired from the TOE.

3. **Security Enforcing Functions:** Specify the exact functionalities that are implemented in the TOE.

4. **Security Mechanisms (optional):** Specify how the security functionality is provided.

5. **Minimum Strength of Mechanisms:** The security target should contain the minimum strength of the security mechanisms implemented within the TOE against direct attacks, as claimed by the product vendors.

6. **Target Evaluation Level:** Security targets should also contain the target evaluation level for the evaluation of the TOE.

The ITSEC scheme proposes ten security functionality classes that are implemented in TOEs. However, the usage of these classes is not obligatory, such as in cases where the predefined classes are not sufficient to express the security functionalities. However, a security functionality class can only be used within the security target only if all the aspects of the class are implemented in the TOE. The security functional classes are mentioned below:

- **Class F-C1** It is derived from the security functional requirements of the US TCSEC class C1 and thus focuses on discretionary (need-to-know) access control mechanisms.

- **Class F-C2** It is derived from the security functional requirements of US TCSEC class C2 and thus provides fine grained discretionary access control as compared to class F-C1. It requires the users to be held individually accountable for their actions through identification procedures, auditing of security relevant events and resource isolation.
- **Class F-B1** It is derived from the security functional requirements of US TCSEC class B1 and requires the product to have implemented functions to maintain sensitivity labels used to enforce a set of mandatory access control rules.
- **Class F-B2** It is derived from the security functional requirements of US TCSEC class B2. It extends mandatory access control to all subjects and objects and requires more stringent authentication requirements to be put in place compared to F-B1.
- **Class F-B3** It is derived from the security functional requirements of US TCSEC classes B3 and A1. It requires the functions of B2 to be implemented and in addition it provides functions to support distinct roles for security administrators and audit is expanded to security relevant events.
- **Class F-IN** It requires that products implement security functionalities that protect the integrity of the data.
- **Class F-AV** It requires that products implement security functionalities that protect the availability of the data.
- **Class F-DI** It requires that products implement security functionalities that safeguard the integrity of data during transmission.
- **Class F-DC** It requires that products implement security functionalities that protect the confidentiality of data during transmission.
- **Class F-DX** It requires that products implement security functionalities that protect the confidentiality and integrity of information that is exchanged on a network.

The ratings that ITSEC provides are based on effectiveness and correctness of the security measures implemented within the TOE. The assessment of the effectiveness of the security functionalities of the TOE is based on the vulnerability analysis and the ease of use of the security measures to ensure that the product security measures do not impede the functionality of the product. The correctness on the other hand focuses on the design and implementation of the TOE. It evaluates the architectural design and the mapping of security functionalities against the security policy. There are seven levels that are defined for the assurance over the correctness of the security functionalities that are implemented within a product *E0* designates the lowest assurance level that can be gained, while *E6* the highest.

- **Level E0** it represents inadequate assurance (product failing the correctness evaluation).
- **Level E1** The security target and a description of the architecture of the TOE are needed to be provided at this level. Optionally, documentation regarding functional testing of the security mechanisms should indicate that the TOE satisfies the security target and the library of the test programs and tools used to test the TOE could also be provided.
- **Level E2** In addition to the Security Target and an information description of the architecture of the TOE, an informal description of the detailed design of the TOE is necessary. The test documentation and the library of test programs and tools used for testing the TOE are required.
- **Level E3** The Security Target, Information description of the architecture of the TOE, informal description of the detailed design of the TOE, the test documentation, library of test cases and tools used are evaluated at this level. In addition, the source code or hardware drawings for all security enforcing and security relevant components is required. An informal description needs to be provided that provides the correspondence of the source code or hardware drawings to the detailed design specified.
- **Level E4** The following documentation is evaluated at this level. *a)* The security target for the TOE; *b)* A formal model of security should be defined or a reference to such model should be provided; *c)* Informal interpretation of the security model in terms of security target should be provided. *d)* Semi-formal description of the architecture of the TOE must be provided; *e)* Semi-formal description of the detailed design;; *f)* Test documentation and library of test programs and tools used for testing the TOE; *g)* Source code or hardware drawings for all security enforcing and security relevant components and an informal description that provides their correspondence with the detailed design.

- **Level E5** At this level all the documentation that is provided for Level E4 is required and is evaluated. In addition, the source code of all runtime libraries that are used within the product or system should be provided and evaluated.
- **Level E6** At this level all the documentation that is provided for Level E5 is needed and evaluated. The TOE should be described formally as opposed to level E5 where it is described semi-formally. The formal specification of security enforcing functions should be also be provided. In addition tools that are used to detect inconsistencies between source code and executable code should be provided. This level provides the highest assurance regarding the evaluation of the products or systems.

The main advantage of ITSEC over TCSEC is that while TCSEC bundles functionality and assurance into one rating, ITSEC evaluates these two attributes separately thereby providing more flexibility to certify products for the mainstream commercial market as opposed to the single minded focus of TCSEC for governmental bodies. ITSEC also provides classes to address integrity, availability properties for products in addition to confidentiality, where as TCSEC only supports confidentiality related properties. ITSEC also supports evaluation of networked systems, whereas TCSEC supports only standalone systems.

3.2 CC Certification Scheme

The *Common Criteria* certification scheme originated from *Trusted Computer System Evaluation Criteria* (TCSEC) [171], the *Canadian Trusted Computer Product Evaluation Criteria* (CTCPEC) [107] and *Information Technology Security Evaluation Criteria*(ITSEC) [52]. It has unified its predecessors with a standard set of criteria for security evaluation. It decouples the specification of security functional and assurance requirements, this is in direct contrast with the approach of its predecessor *TCSEC* where the security functional and assurance requirements are coupled [171] together to provide a “balanced” assurance regarding a system. This decoupling is needed as CC targets a commercial security market, while the *TCSEC* was limited to products designed for US Governmental organizations, and needs to be generic in order to certify different kinds of products that range from software, firmware to hardware.

The CC scheme allows product vendors to describe the Security Functional Requirements (SFRs) for the product and to prove that the set of

Table 3.1: Security Functional Requirements Classes

SFR Class	
Security Audit (FAU)	Communication (FCO)
Cryptographic Support (FCS)	User Data Protection (FDP)
Identification and Authentication (FIA)	Protection of TOE Security Functionality (FPT)
Privacy (FPR)	Security Management (FMT)
Resource Utilization (FRU)	TOE Access (FTA)
Trusted Path/Channels (FTP)	

SFRs are able to counter the threats identified for a Target of Evaluation (TOE), which identifies the specific aspects of the product that will be evaluated. In addition, the CC scheme allows product vendors to choose particular configurations of the product that will be evaluated and these "golden" configurations are also part of the TOE. This information is captured in a document called "Security Target" (CC-ST) which can be seen as the *descriptive* part of the CC certification [21]. The product vendor then defines the set of Security Assurance Requirements (SARs) - that specify actions to be performed by the evaluating laboratories. Based on the *SARs* selected for the product the certification authorities determine the *Evaluation Assurance Level*.

The drawback of this approach is that the EAL can only specify how thoroughly the evaluation has been performed, but it does not answer the question of "Is the software secure?". The answer to this question can be provided by the SFRs that are implemented in the product. The CC scheme classifies the SFRs into 11 high level classes as shown here:

An example of an SFR in the *Security Audit* class and an SAR in the *Security Vulnerability* class can be seen below:

"***SFR: FAU_GEN.2.1*** *For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event* ." [46]

"***SAR: AVA_VAN.1.3E*** *The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.*" [47]

The CC scheme is generic and does not impose specific requirements for different types of IT products. Hence product vendors can implement certain specific security functionalities (SFRs) and get specific parts of their system evaluated in a certain way (SARs) and consequently certified, which may not address the requirements of consumers. To address this issue, CC allows consumers to use Protection Profiles (CC-PP) that contain a combination of SFRs and SARs for a particular type of application, such as Operating System or Databases. When products conform to a specific protection profile, it is easier for the consumer to select and compare the best fit for their needs. But conformance to CC-PP by products is not mandatory, and there is a criticism that product vendors exploit this flexibility of the CC scheme, and choose not to conform to any protection profiles that could be applied for their products [161, 41].

3.2.1 CC Security Certificate Documents

In this section, we present the various documents that result from the CC certification scheme. The CC certification prescribes a common structure for security certificates to facilitate comparison between certified products. Common Criteria is, in-fact, a multi-document certification scheme - i.e., the CC certification results in several certification documents. The CC certification results in the following documents:

- Protection Profile
- Security Target
- Evaluation Report
- Certification Report

Among these documents, the Protection Profile is required only when the product claims conformance to a protection profile. All the documents, except the evaluation report are publicly available documents. These documents are intended for human consumption and are represented in natural language and stored as *PDF* documents.

Security Target

The Security Target (ST) is the most important and informative among the certification documents. The structure of the ST is defined by the CC scheme and its overview is shown in Figure 3.1.

1. The Security Target Identification describes the Target of Evaluation (TOE) at three levels of abstraction:
    - Security Target Reference identifies the security target, it consists of the following elements: ST Title, ST version and ST date.

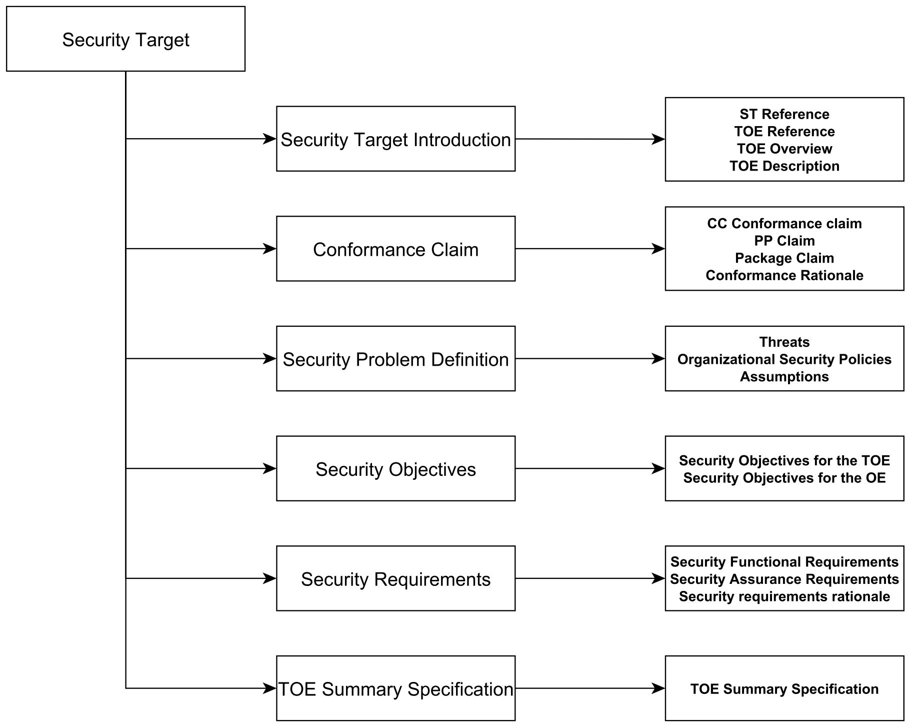

Security Target
Security Target Introduction
ST Reference
TOE Reference
TOE Overview
TOE Description
Conformance Claim
CC Conformance claim
PP Claim
Package Claim
Conformance Rationale
Security Problem Definition
Threats
Organizational Security Policies
Assumptions
Security Objectives
Security Objectives for the TOE
Security Objectives for the OE
Security Requirements
Security Functional Requirements
Security Assurance Requirements
Security requirements rationale
TOE Summary Specification
TOE Summary Specification

Figure 3.1: Security Target Structure Overview

- TOE Reference provides identification material for the TOE and it consists of the TOE title, TOE version, TOE Build number, TOE developer, Evaluation Sponser.
- TOE Overview describes in natural language the TOE, its architecture, the physical and logical boundaries for the TOE.

2. An ST has the following conformance claims:

   - CC conformance specifies the CC version based on which the ST is written, this is essential since each CC version has a certain set of criteria and products that have been certified based on different CC versions cannot be compared in a straightforward manner. It further describes the version of CC standard that the ST conforms to - which implies that the SFRs and SARs in the ST are based only upon the components described in the CC version. While CC extended implies that at least one SFR or SAR in the ST is not based upon the components described in the CC version.

   - PP Conformance: If the ST conforms to any protection profiles, the PPs should be identified clearly.
   - In case the ST conforms to additional CC packages, these must be stated as well. In addition, the conformance rationale for all the different elements must be provided.

3. The Security Problem Definition specifies the security problem that is addressed by the TOE. The actual process of defining the security problem is outside the scope of the CC certification scheme.

   - A security problem can be due to some threats that were identified for the TOE or the Operational Environment.
   - The security problem can also arise due to organizational security policies that require certain assets to be secured.
   - Any assumptions that are made for the OE are also specified in this section. These assumptions are considered to be true during evaluation and not tested in any way.

4. Security Objectives are basically a concise and abstract statement of the solution that is intended to be implemented in the TOE that will counter the problem identified by a security problem definition. The high level security objectives are divided into two part wise solutions:

   - Security Objectives for the TOE: The TOE provides certain security functionality that solves a certain part of a security problem definition.
   - Security Objectives for the OE: the OE of the TOE provides technical and procedural measures to assist the TOE in correctly providing its security functionality (defined by the Security objectives for the TOE).
   - The ST contains a tracing between the security objectives and the security problem definitions that they solve (completely or partially) and in addition, there should be a set of justifications provided that show how each security objective addresses a (part of) security problem definition such as threats, OSPs and assumptions.

5. Security Requirements basically consist of two kinds of requirements:

   - Security Functional Requirements that translate the security objectives that are described on an ad-hoc basis to a standardized language. They are more detailed than the security objectives but refrain from being implementation specific. A rationale and tracing must be provided between each SFRs and the Security objectives that they map to.

- Security Assurance Requirements describe how the TOE must be evaluated using the standardized language that is specified in the CC scheme. It facilitates comparison between two STs. In addition, the ST should also contain the rational for choosing the appropriate SARs.

6. The TOE Summary specification informs how the SFRs are satisfied by the TOE implementation. It should provide general technical mechanisms that the TOE uses to provide the SFR.

Protection Profile

Protection profiles are typically written by users or user communities, regulatory bodies, or a group of developers that define a common set of security needs for a certain kind of products. The protection profiles structure is similar to the security target structure as shown in Figure 3.2, the key difference is that while a security target is specific to an implementation of the product, the protection profile is described in an implementation independent manner in order to cater a range of similar products. In addition, the PP clearly states how security targets should conform to it:

- Strict conformance requires that security targets must meet all the requirements that are mentioned in the PP, such as threats, security objectives, SFRs, SARs among others.
- Demonstrable conformance requires that STs must offer a solution to the generic security problem described in the PP, but it can be done in a manner that is equivalent or more restrictive than what is described in the PP.

3.2.2 CC Certification Process

An overview of the CC certification process is shown in Figure 3.3. The CC certification process is a collaborative process between the following entities:

- Product Users

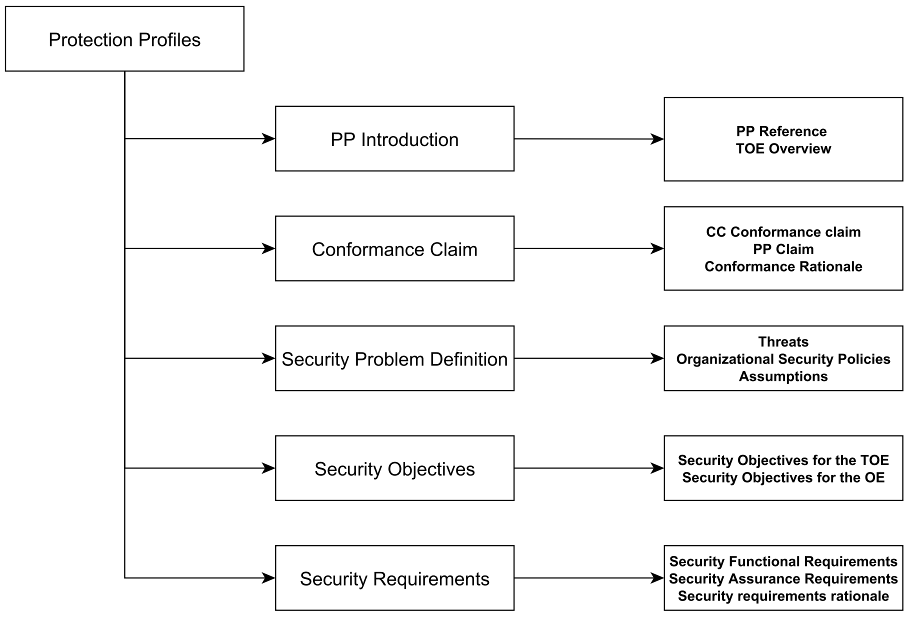

Protection Profiles
PP Introduction
PP Reference
TOE Overview
Conformance Claim
CC Conformance claim
PP Claim
Conformance Rationale
Security Problem Definition
Threats
Organizational Security Policies
Assumptions
Security Objectives
Security Objectives for the TOE
Security Objectives for the OE
Security Requirements
Security Functional Requirements
Security Assurance Requirements
Security requirements rationale

Figure 3.2: Protection Profile Structure Overview

- Product Vendors
- Certification Authorities
- Evaluation Labs

Product Users

Product users or user communities can create a Protection Profile that meets their Security Functional Requirements and Security Assurance Requirements. This also identify the threats that the product must counter, the expected Operational Environment for the product among other details. The SFRs and SARs that are used within the PP are from the Package catalogue of the CC scheme. The PP, once approved by the Certification Authorities is admitted to the PP catalogue from where it can be used by product vendors (or manufacturers).

Product Vendors

When product vendors are trying to build a product that meets the security requirements, they have the option to comply with a Protection Profile created by the target users, in case it is available. In any case, product vendors develop the Security Target using the SFRs and SARs from the CC package catalogue. Based on the SFRs mentioned in the ST and the requirements for the product functionalities vendors develop the product.

Evaluation Labs

Evaluation labs that participate in the certification process must be accredited by the Certification Authorities. Product vendors submit the security target that contains the SFRs and SARs along with a lot of information regarding the product, as well as the product for evaluation. If the product passes the evaluation, which is performed based on the SARs selected by the vendor, the Evaluation Labs submit a Evaluation Report to the Certification Authorities. In case the product fails the evaluation process, the evaluation labs ask the vendors to fix the issues identified in the evaluation phase.

Certification Authorities

CC certification authorities are typically national regulatory bodies, who grant certification to products based on the Evaluation report submitted by the evaluation labs. They issue a certification report, which is publicly available.

Conclusion

In this chapter, we have presented the CC scheme in detail analysing its evolution from the various national schemes. We explain the various concepts of the CC scheme, the various certification documents and their contents, and the CC certification process. From this chapter, we gain an insight on how the scheme was intended to be used *in theory*.

In the next chapter, we analyse the CC scheme *in practice* to see the deviations, if any, from how the CC scheme was intended to be used and how it is actually used.

Publications

The contents of this chapter have been discussed in the following publications:

- ⋆ V. Lotz, **S. P. Kaluvuri**, F. Di Cerbo, and A. Sabetta. *Towards security certification schemas for the internet of services*. In New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, pages 1–5. IEEE, 2012.

- ⋆ F. Di Cerbo, M. Bezzi, **S. P. Kaluvuri**, A. Sabetta, S. Trabelsi, and V. Lotz.*Towards a trustworthy service marketplace for the future internet*. In The Future Internet, pages 105–116. Springer Berlin Heidelberg, 2012.

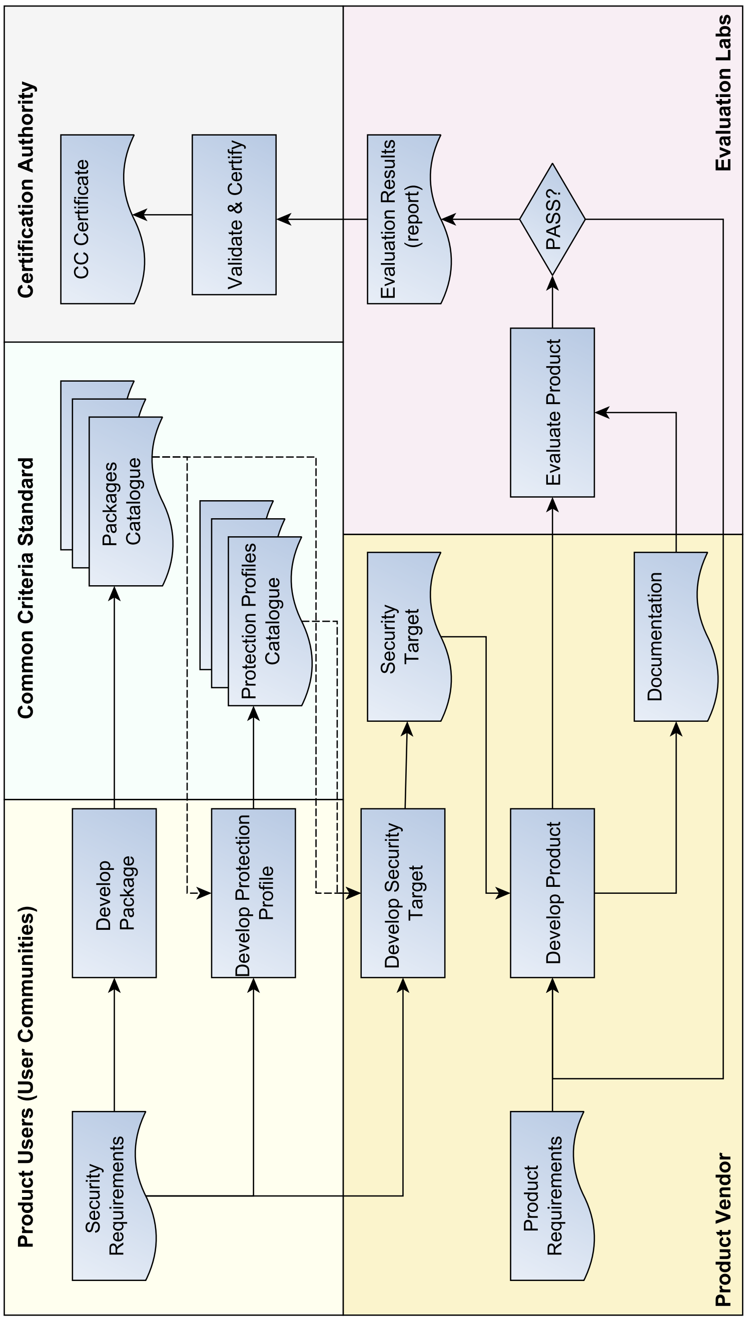


Figure 3.3: Common Criteria Certification Process Overview

Chapter 4

Common Criteria Security certification Practice

"But who will guard the guardians?"

— Juvenal, Satire VI

Common Criteria for Information Technology Security Evaluation (CC) (ISO/IEC 15408) [50] is the most popular security certification standard. It is a globally recognized set of guidelines that provides a common framework for specification and evaluation of security features and capabilities of IT products. At the heart of the CC scheme lies a "common" set of security functional and security assurance requirements. These common requirements enable potential consumers to compare and contrast the the certified products based on their security functional and assurance requirements and to determine whether a product fits into their needs. The CC scheme allows the evaluation of the products at varying levels of evaluation rigour, called *Evaluation Assurance Levels* (EAL), in a range of 1 to 7 (where 7 is the highest assurance level).

Despite the wide use and economic success of Common Criteria scheme [163, 80] (mostly driven by government regulation and government purchase) its current practice has been receiving significant criticisms.

1. *Comparability.* One of the main objectives of CC is to allow consumers to compare certified products on the market in a objective way from a security point of view. However, certification documents are filled with legalese and technical jargon. Hence, comparison is not straightforward nor easy.

2. *"Point in time" certification.* CC certifies a particular version of the product in certain configurations. Any changes to the configuration or

any updates to the product that affect the *Target of Evaluation (TOE)*, which is the part of the product that is evaluated, invalidate the certification. This is not a desirable situation, given that products evolve and are updated at a frantic pace and the certification must not be "frozen" to a specific version of the product.

3. *Long and expensive.* CC evaluation life cycle is lengthy and expensive [132, 181, 180]. In fact, due to the complexity of the process and the high cost, vendors have to spend a large effort on preparation for the evaluation, which adds to the cost and time of the evaluation itself. High assurance level (as EAL4) certification can take $1 - 2$ years, and, often, by the time the process is completed a new version of product is already delivered.

4. *Concerns for Mutual Recognition.* Though the CC scheme is a widely recognized international standard, there are several concerns regarding the consistency of the assessments by the evaluating laboratories located in different countries, since the *Common Criteria Recognition Arrangement* (CCRA) does not prescribe any monitoring and auditing capability. In addition, the relevance of CC certification for governmental institutions, specific national interests can impact the impartiality of the assessment [94, 41].

Although, most of these shortcomings of the application of the CC scheme have, to the authors knowledge there is no quantitative study of the CC certificates, which provides the evidence that these criticisms are applicable to a broad category of CC certified products or are limited to just a few cases.

This chapter provides an exhaustive analysis of CC scheme by systematically analysing the certificates (in Section 4.3) to quantitatively assess the relevance of the points 1 and 2 above. It proves that these issues are well grounded and affect a large part of existing certificates. The points 3 and 4 are out of scope of our analysis, because: an analysis on cost and duration of CC certifications have been discussed in [132, 180] (addressing Point 3), and the mutual recognition issue (point 4) cannot be analysed looking at certificates.

4.1 Analysis Objectives

The fundamental aim of our analysis is to verify whether the CC *practice* fulfils to the intentions of the CC scheme. The main goals of the CC scheme are: *a)* Enabling the comparison of the security characteristics among (certified) "similar" products; *b)* Providing meaningful and useful security assurance to the consumer.

Let us see how these objectives can be reached in practice, and defined the possible checks to assess if these objectives are reached by the current practice (Checks are indicated in **bold** in the following).

For comparing products of the same category, for example databases, from the security assurance point of view, we need to evaluate them against a common set of security requirements (SFRs). To support that, CC proposed the Protection Profiles, that allow for describing a predefined set of requirement for a class of products. Accordingly, to assess if this objective is reached in the actual practice, we need to check:

- **C1**: Do similar products (same category) address the same set of SFRs? **(Differences in the number of SFRs for a given category)**

- **C2**: Does the usage of a specific Protection Profile results in a actual common set of SFRs? **(Differences in the number of SFRs for a given class in PP conforming products)**

- **C3**: Are Protection Profiles available for the different categories ? **(Protection Profile Availability in each category)**

- **C4**: Are Protection Profiles actually used? **(Protection Profile conformance by products per category)**

To provide meaningful assurance to the consumer, the certification issued should be valid along the product lifecyle. Considering the need to perform changes in the software (e.g., security patches to address new vulnerabilities) or in the landscape (e.g., configuration), CC scheme proposes a *product certification maintenance* under the Common Criteria Maintenance Agreement (CCMA). Under this scheme, a specific version of the product can be initially certified and any changes made to it in future will be localized to the aspects that have been changed instead of the whole product being re-evaluated. So, our next objectives are to evaluate:

- **C6**: Is the CCMA actually used in practice? **(How many products are maintained under the CCMA?)**

- **C7**: Are CCMA certified products secure? **(How many CCMA certified products have disclosed vulnerabilities?)**

4.2 Analysis Methodology

The analysis is performed using information collected from the Common Criteria website [48], that contains products that are certified, and the National Vulnerability Database (NVD) [1], that contains the list of disclosed vulnerabilities in products. In particular we considered the following data sources:

(a) CC Certified Product List [51]

(b) CC Certified Protection Profile List [49]

(c) Security Targets of Certified Products [51]

(d) CC Part 2: Security Functional Requirements Document [50]

(e) NVD database [1]

The data from these sources could not be processed automatically, given that the data is contained in human readable documents in most cases. Hence, we performed additional processing to allow advanced reasoning to be performed :

1. The Certified Products and the Protection Profile CSV files were converted to SQL and stored in a database
2. We downloaded the Security Target files from the Certified Products list which contains URLs of the Security Targets of the respective products
3. We examined the Common Criteria Part 2 document that contains the standardized Security Functional Requirements and stored them in database
4. We analysed the Security Targets for the presence of these standardized SFRs and stored the results in the database
5. Cross-reference certified products against the National Vulnerability Database (NVD) for disclosed vulnerabilities.

The whole analysis was automated using Python scripts except for steps *3* and *5*. The certified product list contains data of products that fall under three categories:

(a) Certified products

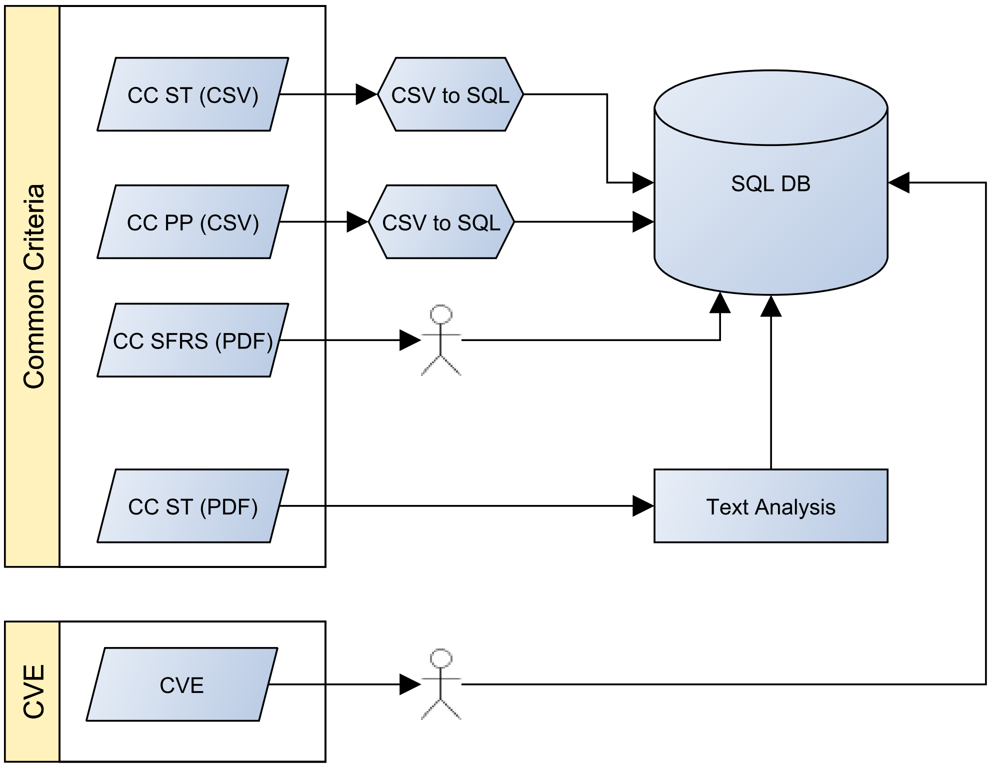


Figure 4.1: CC Certificate Analysis Approach Overview

(b) Certified products under *maintenance agreement*

(c) Archived certified products.

We consider only products with valid certificates (1971 certified products) and ignored the archived certificates for our analysis. Due to technical reasons, such as malformed URL or a digitally signed PDF document that could not be parsed into text, we could not process 95 certificates. Hence the data set that we considered in our analysis was 1532 security targets of certified products and 344 security targets of products under the maintenance agreement.

4.3 Analysis Results

In this section, we present the results of our analysis. Although we have performed the analysis on a broad range of categories, in this section we present products that have been certified at EAL4+ since most of the products are certified at this level.

4.3.1 Comparability of certified products

We compared products based on the number of *SFRs* that are addressed by each product in a certain category to understand the differences in certified products based on their security functionalities. Figures 4.2 and Figure 4.3 show the SFRs addressed in products for *Database* and *Operating System* categories certified at *EAL4* (and *EAL4+*) and conform to CC version *3.1*. Each shade of the bar in the figures 4.2 and 4.3 represents products that conform to a specific protection profile and the white bars represent products that do not conform to any protection profiles.

It can be observed from Figures 4.2 and 4.3 that even among products claim conformance to a protection profile, there is a considerable difference between the SFRs addressed by the products. And products that tend to show little or no difference are either different versions of the same product or products from the same vendor. Among the products that do not conform to any protection profile there is a huge difference in the number of SFRs addressed.

Since products that conform to protection profiles have a smaller variation between security functionalities when compared to products that do not conform, we analysed the protection profile conformance across all categories. The availability of protection profiles across various product categories is shown in Figure 4.4 compared against certified products in the

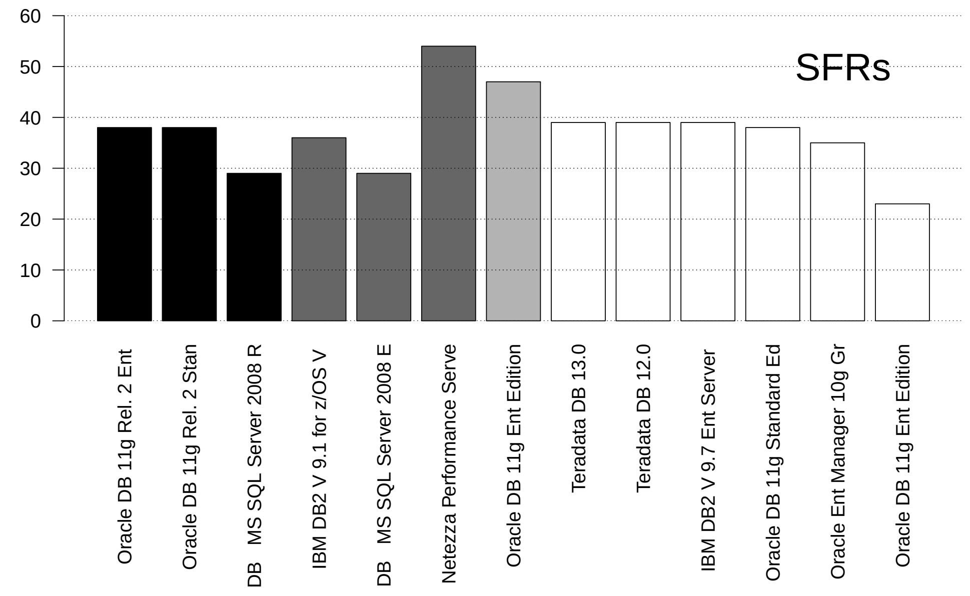


Figure 4.2: SFR variation in Database Category for EAL4+ (Each shade of the bar represents products that claim conformance to a particular Protection Profile)

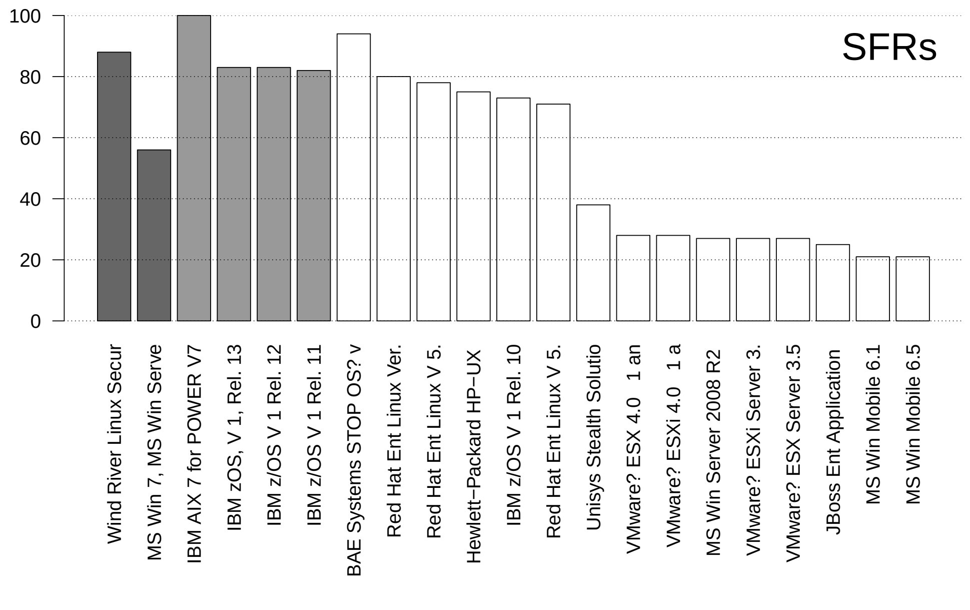


Figure 4.3: SFR variation in OS category for EAL4+ (Each shade of the bar represents products that claim conformance to a particular Protection Profile)

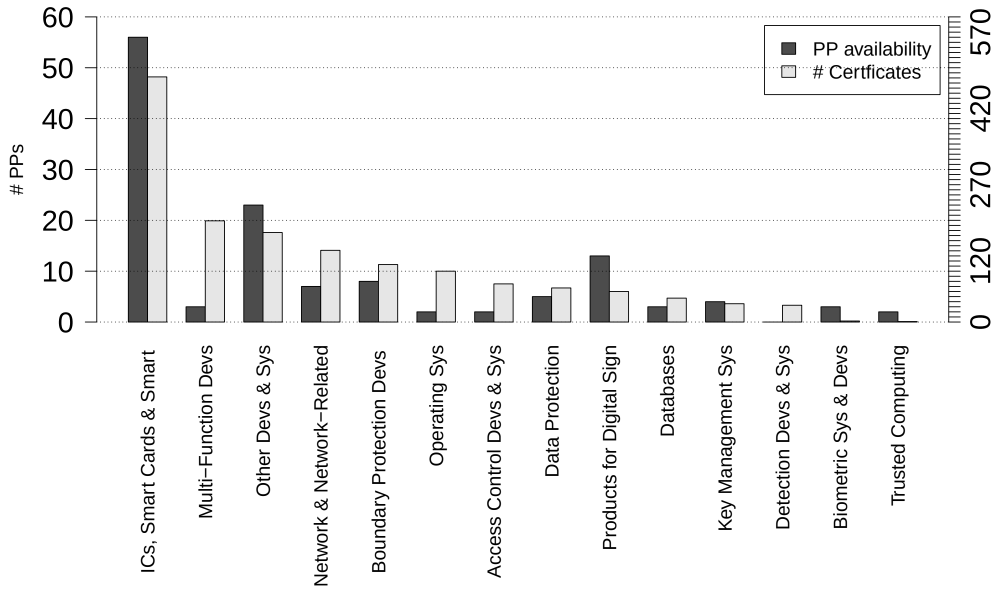


Figure 4.4: Protection Profile Availability and Certified Products across categories

same category. It can be noted that the availability of protection profiles is rather low across all categories of products except the *IC's and Smart Card* category.

Figure 4.5 shows the percentage of certified products conforming to at least one protection profile. The average PP conformance rate among certified products of our data set is 14%, with standard deviation around 10 % (see Fig 4.5, rightmost column). This indicates that a relatively low number of certified product use PPs with relevant differences among categories. Indeed, a closer inspection reveals that the products broadly related to *hardware or firmware* show higher conformance than products that fall under the software-only category. This low conformance could also be due to vendors finding it difficult to build products that conform to a particular protection profile, while the products themselves are targeted for the general commercial market. And so, to conform to a particular protection profile, which is produced by specific consumer or a consumer group, does not provide any competitive advantage in the general market.

On the other hand, the low PP conformance makes it difficult to compare and contrast the security requirements addressed by the certified products. In fact, the non-conformance to a standardized PP allow vendors to customize the scope of certification to features that are very different from other certified products. As an example, a product in a certain category

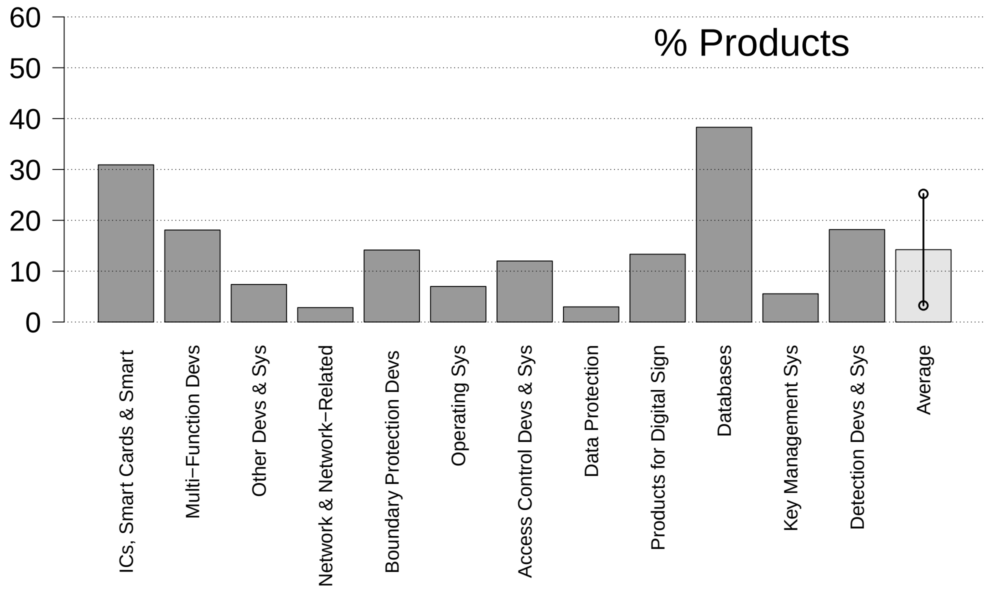


Figure 4.5: Protection Profile Conformance among certified products (Right-most bar shows the average conformance to PP for our dataset, 14%, and corresponding standard deviation, 11%)Note that categories with less than 10 products are not shown

could make claims that it addresses more SFRs related to data protection, while another certified product in the same category may have claims addressing SFRs in the access control class. Furthermore, they can make different assumptions about the threat agents and their capabilities, thereby affecting the nature of the security assurance that can be gained by the consumers. Hence, comparison of certified products in such cases can become rather labour intensive and a very time consuming process.

In order to better understand the variation of Security Functionalities addressed in each certified product, it is not simply enough to look at the number of SFRs addressed. Hence, we looked more closely at the usage of each SFR in various products. The variation in the usage of SFRs for products in the database category, certified at EAL4+ is shown in Figure 4.6. It can be clearly noted, that there are not a lot of SFRs that are commonly used by the various products, even though they are in the same category of products. This further illustrates that comparison of certified products is not a trivial aspect.

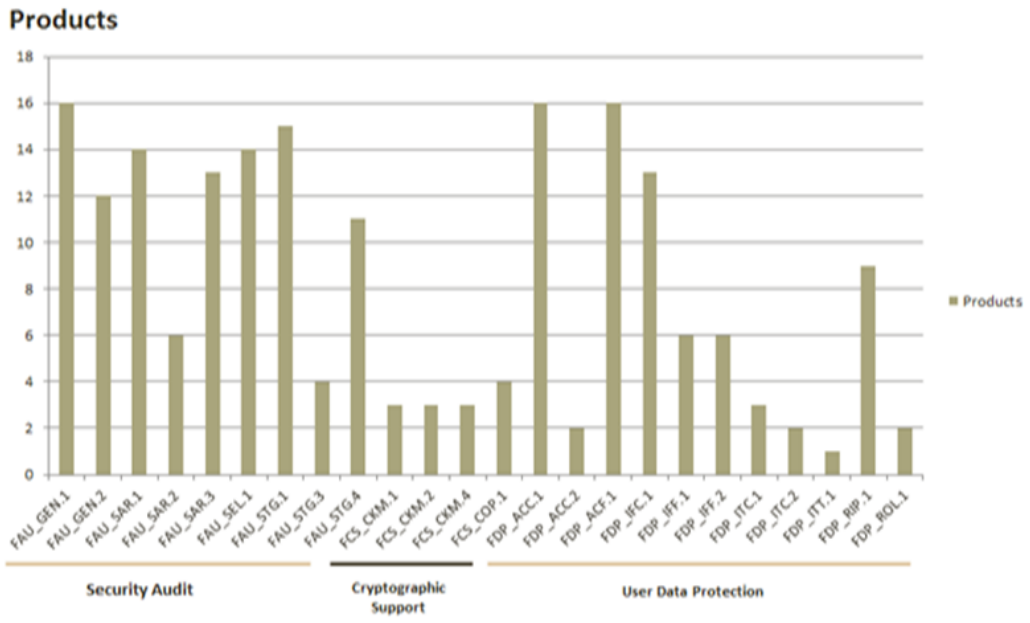


Figure 4.6: SFRs usage among different products in Database category (EAL4+)

4.3.2 Point in time Certification

The CC scheme certifies products at a point in time, that is, certification applies to a particular version of the product and in a certain set of configurations. But products do need to evolve - either to provide new functionalities or to fix problems or both. And in such cases, the CC certification does not apply to the new version and the whole product has to undergo the certification all over again which is once again a very time consuming and expensive process, especially when the changes made to the product are very minor. In order to avoid such situations, the CC scheme allows products to be under the CC Maintenance Agreement (CCMA) where only the changes made to the product are evaluated and certified. This aspect of the CC scheme would allow the products to be certified over a *period of time* instead of a *point in time*.

We verified this aspect in practice and Figure 4.7 shows the certified products that are under the maintenance agreement across the various product categories. It can be observed that the number of products under the CCMA scheme is high among IC's and Smart Card category when compared to the other categories. And indeed the total percentage of products that are under the maintenance agreement is just 22% of all the certified products. And in fact, excluding the IC's and Smart Cards category, the

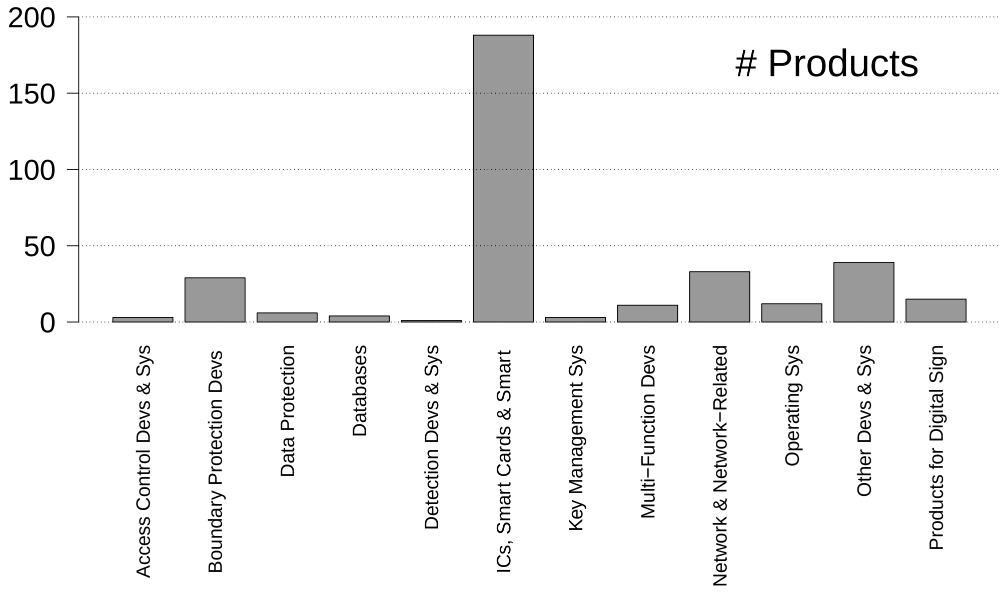


Figure 4.7: Products under CCMA

percentage of products that are under the CCMA scheme comes down to approximately 15%.

Such low numbers of products under maintenance raise an important question on the product's lifecycle, especially when vulnerabilities are found in the product which need to be fixed - can a product vendor issue a fix and technically *lose* the certification or keep selling the vulnerable version of the product to claim the certification?

In order to better understand this question, we have used the National Vulnerability Database (NVD), to cross-reference the certified products with products that have known vulnerabilities. Since we could not automate this step, we limited our analysis to the Database and Operating System categories certified at assurance level EAL4+. In the Operating System category (shown in Figure 4.8), we found 22% of the products under the maintenance agreement have disclosed vulnerabilities. And in the database category we found only 25% products under the maintenance agreements are shown to have a known vulnerability. To contrast this, we cross reference products (in the database category at EAL4+) that are not under the maintenance agreement and 85% of the products have shown to have a known vulnerability (Figure 4.8) while 72% of products not under CCMA in the Operating systems have reported vulnerabilities.

Though we do not claim that the vulnerability is in the certified "golden" configuration, these figures show that in practical usage of the products the

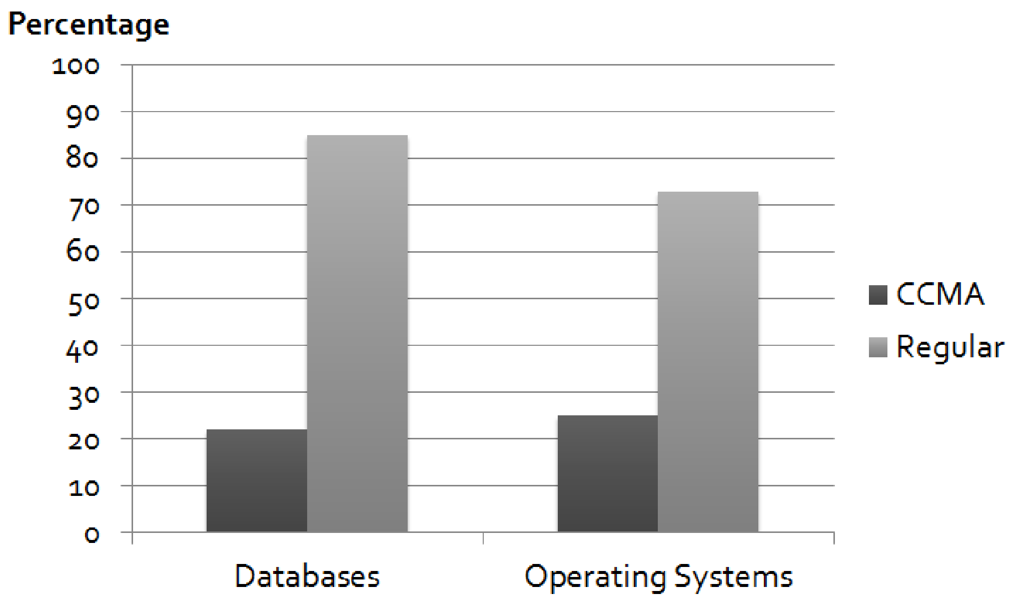


Figure 4.8: Vulnerabilities in products under CCMA compared to regular CC certified products

issue of addressing new vulnerabilities must be discussed. And clearly, a point in time certification does not cope well with the dynamic landscape of a product's lifecycle.

4.4 Findings about the certification practice

In this section, we present our findings based on the results from the analysis of the CC certificates.

Comparability of certificates

Our results illustrate some reasons behind the lack of comparability of of certificates. In particular, the security assurances sought in the certificates produced for the same class of products often exhibit large differences.

When products conform to Protection Profiles, the variation between SFRs addressed by the products is not so large. However, the protection profile conformance rate is rather low, particularly in software products. We believe that making products conform to standard Protection Profiles in each product class could provide better comparability between certified products.

On a more fundamental level, we found out that without tool support it is not a trivial task to perform comparison between products based on not only their EALs but also their SFRs. In this regard, the Security Targets should be represented in a machine processable manner which would facilitate automated reasoning to be performed on them.

One point in time certification

The low numbers of products under maintenance raise an important question on the product's lifecycle, especially when vulnerabilities are found in the product which need to be fixed - can a product vendor issue a fix and technically *loose* the certification or keep selling the vulnerable version of the product to claim the certification? It is rather obvious, that the product vendor will choose to fix issues and risk losing the certification.

Our results show that, despite the finding of new vulnerabilities, which are sometimes unknown at the time of initial certification, and the provisions made by the Common Criteria scheme to support incremental certification (CCMA), the certified products are overwhelmingly certified once and for all. While this is perfectly valid in itself, it shows that two certificates should be compared with respect to their time of issuance, as well as with respect to information from publicly available vulnerability databases (such as NVD). It also illustrates that the description of the TOE and SFRs should make it clear whether new vulnerabilities should be considered, and would in particular benefit from being machine readable to automate this identification.

4.5 Outlook for Common Criteria scheme

Contributions have been proposed in order to extend the comparability of the certificates produced by the Common Criteria evaluation process and to ease its application. Those approaches rely either on an extension of the CC certification scheme, or on tools to support a more homogeneous generation of certificates.

Common Criteria Framework Extensions

Countries that are members of the Common Criteria Recognition Agreement (CCRA) have recently agreed to develop internationally accepted Protection Profiles (known as Collaborative Protection Profiles - CPPS) for each class of products. Each product has to conform to the CPP that is applicable in its class.

Computer Aided Certification

These approaches most notably aim at providing some guidance for evaluators in their writing of certificates, and in making sure that their description is consistent. Those different approaches might therefore be extended in order to provide the necessary support to implement the recommendations we suggest above, in particular that of rendering certificates machine readable, with comparable SFRs and TOEs.

Certification Toolboxes have for instance long been designed. The Common Criteria Design Toolbox created by Tore Nygaard [127] aims at supporting the writing of Common Criteria certificates. The toolbox aims at supporting the uniform definition of protection profiles, and at certifying those profiles themselves.

Other proposals have extended such toolboxes with security ontologies. Ekelhart et al. [65] and Chang et al. [38] proposed to use an ontology as the core tool to manipulate Common Criteria certificates. Ekelhart et al. first define a security certification ontology that describes the core concepts of the Common Criteria framework. The authors have also applied a similar approach to another certification framework, namely ISO/IEC 27001 in [68]. They then define a tool to annotate an existing certificate and to link different parts of the certificate document through relationships between concepts. Chang et al. created another tool based on Eckelhart et al.'s ontology and in particular detail the implementation of their tool. The main improvement of this approach over plain toolboxes is that the definition of an ontology makes the relationships between the different concepts apparent. For instance, those relationships materialize consistency checks between the different sections of a certificate or of a protection profile.

Conclusion

We have presented the results from a thorough analysis of the certificates of Common Criteria certified products to concretely understand the drawbacks of the CC practice. We presented evidence on the variation of security functionalities in products and that EAL should not be considered to be the only metric to analyse the security of a product. The low rate of conformance to protection profiles makes the comparison of certified products more complex. In addition, we also discovered that very few products are under the maintenance agreement.

We propose that conformance to a standard (or basic) protection profile for each product category could help in allowing easier comparison between products. Based on the complexity we encountered in performing the analysis, we believe that machine processable representation of Security Targets of Common Criteria can ease the comparison process to a large extent.

Publications

The contents of this chapter have been discussed in the following publication:

- ⋆ **S. P. Kaluvuri**, M. Bezzi, and Y. Roudier. *A quantitative analysis of common criteria certification practice*. In Trust, Privacy, and Security in Digital Business, pages 132–143. Springer International Publishing, 2014.

Chapter 5

Security assurance in Service Environments

"Whenever we proceed from the known into the unknown we may hope to understand, but we may have to learn at the same time a new meaning of the word 'understanding'."

— Werner Heisenberg, Physics and Philosophy: The Revolution in Modern Science

Traditionally, organizations had to procure large scale IT infrastructures to power the software solutions as shown as Figure 5.1. This requires considerable amount of human, financial and technical effort to maintain such large IT infrastructures. However, over the last decade, there has been a profound change in the manner software is provided and consumed. Software solutions have been moving from *on-premise* to *off- premise* and are consumed as *Services*.

5.1 Services - Benefits & Security Concerns

The change to cloud based solutions is mainly driven by the adoption of Service Oriented Architectures (SOA) by many software vendors. SOA enables software vendors to enable usage of their software as a "*Service*". A service, in this context, can be defined as software running on a remote server, consumed by "*clients*" through an Application Programming Interface (API). Typically, service consumers use *thin clients* to access the service such as Web Browsers, Mobile Applications and so on as shown in Figure 5.2. The important aspect of this software provisioning paradigm is that the consumer does not procure or maintain the IT infrastructure needed for

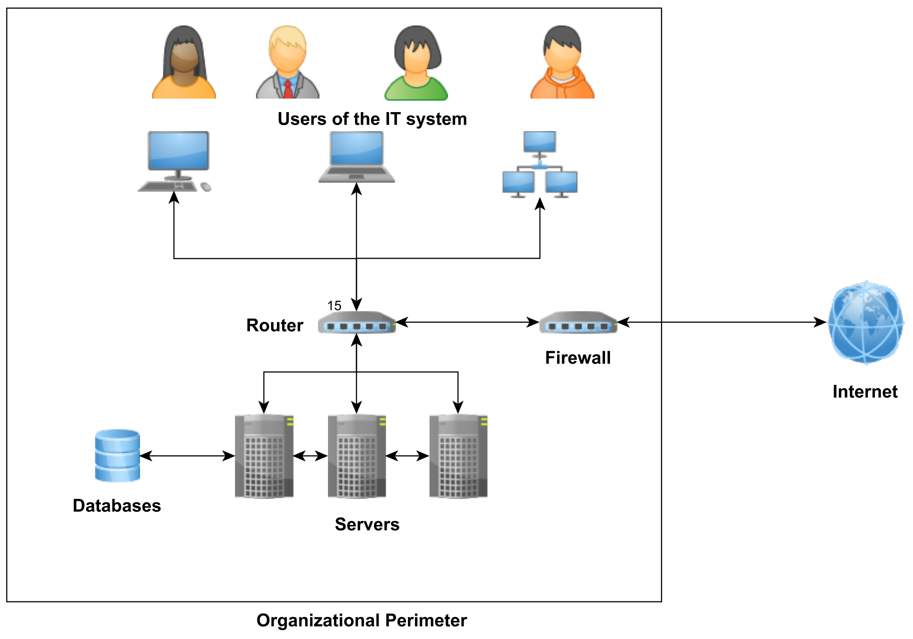


Figure 5.1: Traditional IT environment in Organizations

the software and in many cases is oblivious to the actual IT infrastructure on which the software is executed.

Cloud Services, as defined by the National Institute of Standards and Terminology (NIST) [79], have the following characteristics:

- **On-demand self-service** - implies that a consumer can unilaterally provision the service without any prior human interaction with the service provider.
- **Broad network access** - implies that all the capabilities offered by a service should be accessible over the network
- **Resource pooling** implies that the service provider's resources (computing, network, data) are pooled to serve multiple consumers using a multi- tenant model, with different physical and virtual resources dynamically assigned or reassigned based on consumer demand.
- **Rapid elasticity** implies that the service capabilities should be elastically provisioned and released dynamically in order to scale up or down based on consumer's needs.
- **Measured service** implies that the service provider should be able to

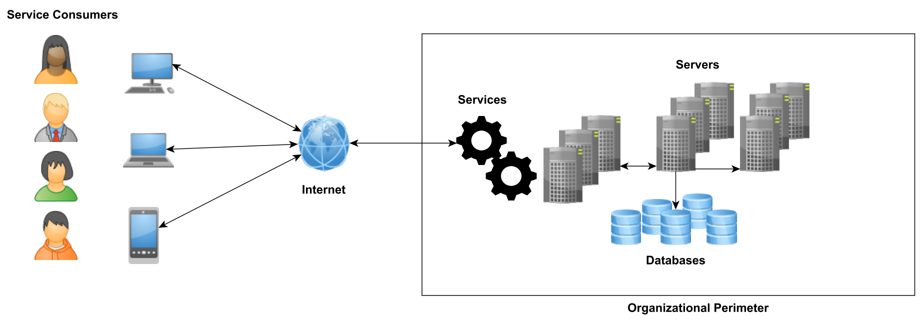


Figure 5.2: Service solutions in Organizations

monitor, control and report usage of the capabilities of the service in a transparent manner to the consumer and provider.

SOA has enabled various software provisioning models where in different computing capabilities can be consumed by users based on their need. A brief description of the different service models is provided here, along with a few examples:

- **Infrastructure as a Service (IaaS):** The hardware infrastructure is offered as a service. This model had a profound impact on organisations in particular, where they could make use of IT infrastructure that is *off- premise* and *on-demand* which provides enormous flexibility and economic benefits. The continued success of Amazon Web Services [7] and other similar IaaS offerings underpins the effectiveness of such solutions.

- **Platform as a Service (PaaS):** A platform on which consumers can deploy applications created by them without having the need to manage the underlying infrastructure. Platforms require consumers to use their Software Development Kits (SDKs) to use the functionalities provided by the platform. Popular examples of PaaS solutions are Windows Azure [113], Google Marketplace [75] among others.

- **Software as a Service (SaaS):** allows a software running on a remote server to be accessed through a web interface or from thin clients. The consumer of such SaaS offerings does not need to maintain, configure or manage the SaaS offering with the exception of user specific configurations allowed within the SaaS offering. Popular SaaS solutions are Gmail [88], iCloud [87] among others.

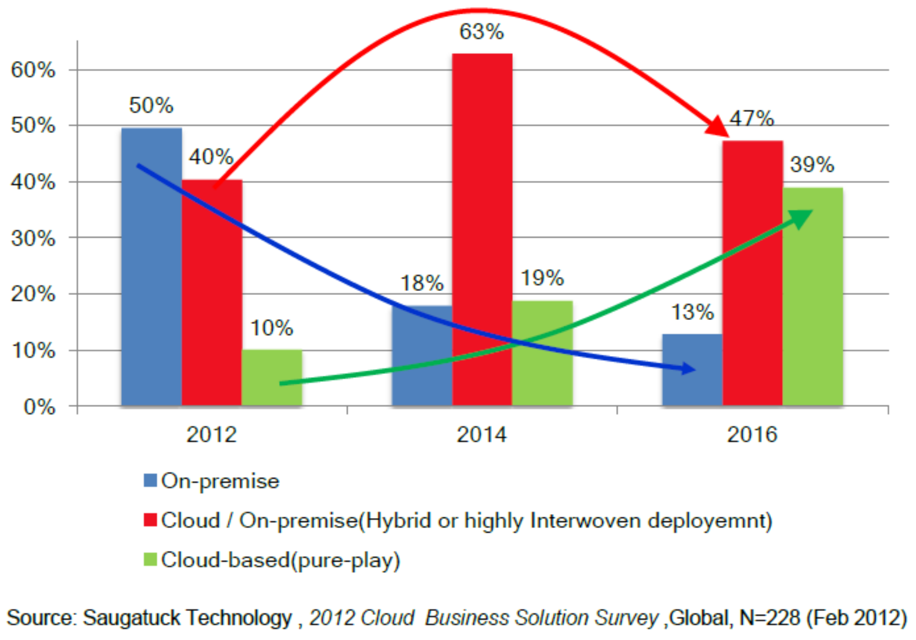


Figure 5.3: Growth of Cloud and Cloud based Applications

5.1.1 Benefits of cloud services

A global survey [32] conducted on the adoption (or planned adoption) of cloud services in enterprises reveals an increased push towards cloud based solutions. In fact, the acceleration in the growth (current and projected) of cloud based services is striking (Figure 5.3). In order to understand better the reasons behind this switch, we explain the benefits that cloud services offer to enterprises.

Economic Benefits

Cloud services relieve consumers from having to procure large scale IT infrastructure, thus directly reducing the capital for small and large enterprises, and thus also reduces the operating costs required for maintenance of such large IT infrastructures [13].

Time to Market

Cloud services have significantly lower *time-to-market* when compared with traditional software. The primary reason is that cloud services are deployed and executed in a certain configuration [177], thus facilitating the developers to tailor the service implementation for the platform on which it is executing. This reduces the efforts to test the product on multiple configurations (hardware/ software) to check for compatibility and so on. On the other hand, cloud services development is typically based on Agile methodology [102], and thus reduces the *time-to-market* even further.

Interoperability

Cloud services facilitate inter-organizational, interoperability on an precedented scale. Since services communicate using standardized protocols and exchange messages, it facilitates organizations to collaborate together, without impacting their internal software infrastructure.

Broad market reach

Cloud services provider can target consumers that range from enterprise to regular end-users with the same implementation of the service. For example, OneDrive [89], a cloud storage and synchronization service provided by Microsoft, not only caters to business users offering up to 1 TB of storage, multiple accounts and among others, but also has plans for regular end-users. Such broad market reach is not always feasible through traditional software models.

Scale up/ Scale Down

A major reason behind the success of cloud services is that they allow consumers to scale up or scale down based on their needs. This allows organizations to convert their fixed IT costs, that is the costs of maintaining a large scale IT infrastructure are always fixed irrespective of the performance of the company, into variable costs - where in, consumers can decide to scale down in phases where the demand for their products/ services is not high thus reducing the overall expenditure of a company.

Service Composition

Another key benefit of cloud services is the ease with which services can be composed to form complex applications. The composition can happen *vertically*, that is, for example, a Software as a Service Provider(SaaSP) can use a Platform as a Service (PaaS) as an execution environment for their service, while the PaaS provider could in turn make use of Infrastructure as a Service (IaaS). A *horizontal* composition typically can be seen as *Service Orchestration* where several *SaaS*s are composed to form complex business applications.

These are just a few, but key, benefits that cloud services offer to consumers. However, there are still issues that prevent cloud services to achieve their full potential. A major obstacle for the adoption of cloud services is the concerns on the security of such solutions. We discuss these issues in detail in the next section.

5.1.2 Security concerns in Services

The very benefits offered by cloud services raises security concerns. A study conducted [90] in the same year as the previous survey on cloud adoption [32], reveals that an overwhelming majority (78%) of the organizations were concerned about the security of cloud service offerings. In fact, they cite security concerns as a key reason for them not adopting cloud services.

The reasons for such security concerns stem from the following major aspects:

- **Lack of Transparency:** [40, 98, 99, 27] Service consumers typically invoke a service through its web interfaces (web APIs) and they cannot know the internal architecture of the service beyond the description of the interfaces, unless the cloud service provider discloses this to consumers. In such scenarios, consumers - in particular, enterprise consumers - have concerns on *where* their data might be stored or processed. Especially, given that enterprises have to often comply with data protection laws that prevent enterprises from storing their consumer's information outside certain jurisdictions.

- **Lack of Control:** [40, 142, 170, 139] Services shield consumers from the complexity of procuring IT infrastructure by allowing them to use the computing resources as services, however, there consumers fear losing control over *how* their data is handled and used by cloud service providers.

- **Lack of Accountability:** [78, 105, 18] Typically, in cloud environments, no single entity has complete control over the software, its execution environment, and the physical IT infrastructure. This raises several concerns regarding security, liability and accountability for consumers of such service based solutions. Moreover, auditing processes which are quite well established in traditional software environments, are still not yet adapted to the needs of cloud environments.

In addition, there are several other concerns, stemming from the reasons explained before, such as concerns on data ownership, privacy, third party usage of data among others. Privacy, in particular, has been a primary concern for end -users as they store more personal and private information on the cloud services. The advent of Social Networking Sites (SNS) has only made the issues of privacy even more complex, as consumers have a legitimate need to share information using SNS but at the same time, retain control over their privacy.

Hence, security assurance of services is needed in order to facilitate the adoption of cloud services in critical domains such as healthcare, finance, defence among others. In the next chapter, we discuss the various means through which cloud providers aim to provide security assurance to consumers.

5.2 Security Assurance of Services

In this section, we examine whether the various means to gain security assurance in traditional software provisioning models (presented in Chapter 2) can be applied in Service Environments.

In addition, we examine cloud-centric, lightweight certification schemes, both security specific or covering security aspects, currently used in cloud environments.

The security concerns in service environments (presented in Section 5.1.2), i.e., lack of transparency, control and accountability should be allayed by one of the approaches to gain security assurance.

5.2.1 Service Level Agreements

Service level agreements have been the focus on a lot of work to gain security assurance in service consumption [59]. By establishing SLAs with a service provider, consumers can hold the provider accountable for any violations of the agreement that might occur. In addition, providers can agree to implement certain security measures thus partially allaying consumer's concern on the lack of control. However, they do not still address the issue of lack of transparency.

Another issue of using SLAs in cloud environments is the expression of objectives that the provider has to meet in cloud environments. When the objectives are expressed in an abstract manner, it becomes harder to monitor them to detect violations, on the other hand, when the objectives are expressed in fine- grained manner, it consumes a lot of resources of the cloud provider to constantly monitor these constraints [3] and hence is not a scalable approach in cloud environments where services have user bases that run into millions. Moreover, most of the work is focused on Quality of Service (QoS) aspects than security aspects. QoS objectives are more easier to monitor from a consumer's end as opposed to security objectives.

5.2.2 Process based Security Certification

Process based security certifications, such as ISO 27001:2005 [91] and SAS 70 [155], are used extensively in cloud environments especially because consumers do not have any control over the operational environment of the service. However, the assurance gained from such schemes partially address the concerns of lack of transparency, control and accountability. This is because, process based security certification schemes only focus on the *security measures* and *physical measures* but not on the service implementation, service architecture nor its design.

5.2.3 Product based Security Certification

Product based security certifications, such as CC and CPA, are able to provide security assurance that addresses the concerns on lack of transparency. Descriptive certification schemes such as CC [176], that contain a lot of information on the certified product and its architecture helps to mitigate concerns on lack of transparency. In addition descriptive certification schemes also provide the security measures that are implemented within the product and thus contributes to mitigating concerns on the lack of control over the service to a large extent. However, since they do not consider the process based aspects (security procedures and physical measures), they do not completely mitigate the concerns on lack of control. However, product based security certifications identify clearly identify the provider and their obligations (to secure the service against the identified threats) and hence when any violations occurs the provider can be held accountable, theoretically - as there is no precedent for a certified service (or software) provider to be held liable for damages.

However, CC and CC based schemes, in their present form, does not cope well with service environments and hence cannot be used to gain the necessary security assurance to consumers. There are three main reasons:

(i) **Processing of CC Certificates:** The certificates resulting from the CC certification process are represented in natural language and are filled with legalese and above all, are rather unstructured. Such certificates do not scale well to service scenarios such as Service discovery, service selection and service composition.

(ii) **Assurance of Deployed Services:** During the CC evaluation process, only the chosen “deployable” configurations are evaluated and certified while making assumptions on the operational environment for the secure operation of the product. From CC evaluation perspective, these assumptions are always considered fulfilled, and the consumer is delegated the responsibility of operating the product in an environment that satisfies these assumptions. In fact, such strong assumptions on the operational environment are not viable in a service landscape as, very often, neither the service provider nor the consumer have any control over the operational environment. Moreover, certifying “deployable” configurations of a service does not provide much value to consumers, as they need assurance on the “deployed” service.

(iii) **Assurance of composed services:** CC scheme is a system wide certification and is not devised to certify modular systems. Though the CC scheme proposes the usage of Composed Assurance Packages (CAP) to gain assurance over composed systems it does not provide high levels of assurance [106], and, as a matter of fact, these packages are not used.

In the following subsections, we discuss in detail the issues that arise from these key aspects.

Processing of CC Certificates

In service environments, consumers need to be able to discover services in an automated fashion. In case of certified products, consumers need to be able to specify their security requirements and find certified services that match their requirements.

CC certification results in several documents out of which the Security Target document contains a lot of information regarding the certified product. It contains, at varying levels of abstraction, the architecture of the certified product, the threats countered by the product, the security functionalities that are implemented within the product among many other details. However, this information is represented in natural language and in an unstructured manner. This prevents CC certificates to be used within service scenarios such as service discovery, service selection among others.

Assurance of Deployed Services

In Common Criteria certification the Target of Evaluation (*TOE*), which describes the parts of the product that are subject to evaluation, delegates responsibility to the Operational Environment (*OE*), which is the environment in which the *TOE* operates, for its secure operation. Typical examples can be Application Server or Database products that require the Operating System, which is part of the OE, to provide user role management. In traditional software provisioning models, these objectives for the *OE* serve as a guidance for the consumers to configure their *OE* to ensure the *TOE* operates securely. This is possible only when consumers have control over the *OE* which is the case in the traditional software provisioning models.

However, in service environments consumers do not have neither any control over the service (*TOE*) and its execution environment (*OE*) nor transparency regarding the service architecture. In such scenarios, certifying "deployable" configurations does not provide any meaningful assurance to service consumers. Only certifying the "deployed" services can provide the required assurance to consumers. Certification of a deployed service can be done similarly to the certification of a (set of) "deployable" configuration(s) of the service. But the service landscape is dynamic and neither the service consumer nor the certification authorities have control over the service and its operational environment. Thus, even when a service is certified, consumers cannot be certain that the service that is being consumed is operating in the certified configuration. The reason for this is due to the static certification lifecycle of the CC scheme, where once a product passes the evaluation and is certified, the role of the CC authorities end and the onus is on the consumer to ensure that the product he procures is the same as the one that is certified. The CC scheme's " Assurance Continuity" [43] allows incremental versions of a product to be certified by evaluating only the changes made to the products. However, this does not reflect any proactive role of the certification authorities, it merely provides software providers to reduce the time and expense to get newer versions of their certified products evaluated. Clearly, this *static* certification lifecycle does not scale to *dynamic* service environments.

Assurance of composed services

A key feature of SOA is the ease through which services can be composed to form complex, composed applications. Indeed, interoperability of services is a major factor for the widespread adoption of service based solutions. Composition of services can happen at design time or at runtime in both the scenarios, if we assume the participating services are CC certified, it is not trivial, (in the case of runtime compositions practically impossible) to gain the assurance of the overall composition. One reason being that composition was never an inherently addressed issue within the CC scheme, as it was designed to be a system wide certification. Another reason being the natural language representation of security certificates from the CC certification process.The CC scheme requires product vendors to disclose certain information regarding the system being certified, the security features implemented in the product, the assets being protected, the threats and so on, in a document called as the *Security Target (CC-ST)*. This document is seen as the descriptive part of the CC certification. The CC scheme only prescribes the content that must be captured in the CC-ST document, but does not prescribe any rules or structure for the representation of this content. The only standardized elements in the CC-ST document are the Security Functional Requirements (SFR) that the vendor claims the product meets and Security Assurance Requirements (SARS) that describes the rigour of evaluation of the product. These SFRs and SARs are prescribed in the CC standard, but in the context of the CC-ST they cannot provide complete information regarding the security of a product. This is a major limitation in performing any sort of automated reasoning on the security certificates, which is an essential step in facilitating certification composition to assess the overall assurance of a composed service.

5.2.4 CSA STAR

Cloud Security Alliance [54], has proposed a Security Certification Scheme for cloud services called STAR (Security, Trust, Assurance Registry). The STAR certification scheme is one of the most comprehensive security certification schemes that are proposed for Cloud services and in fact are tailored for cloud services. It allows cloud services to be certified at three levels of assurance 5.4, based on based on the evaluation models selected by the service providers. CSA prescribed the Cloud Controls Matrix (CCM) cite that is used as a criteria against which cloud services are evaluated in the CSA STAR certification scheme.

Level 1: Self Assessment:

At this level, the security controls that are implemented by a cloud provider are documented and made available to consumers. This provides transparency for consumers into the security practices implemented by the cloud provider and allows consumers to compare different cloud providers.

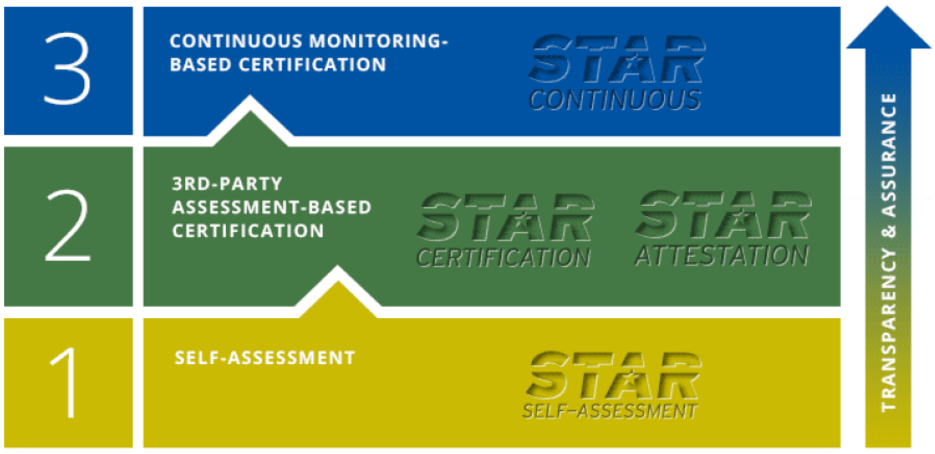


Figure 5.4: Assurance Levels in CSA STAR certification scheme [55]

Level 2: Attestation & Certification

CSA STAR attestation provides guidelines for Certified Public Accountants (CPAs) to conduct SOC 2 evaluation using the criteria from AICPA and the CCM. CSA STAR certification involves rigorous assessments by independent third party laboratories of the security of a cloud service provider. This leverages the requirements of the ISO 27001:2005 management system together with the CSA CCM.

Level 3: Continuous Monitoring

This provides the highest level of assurance by constantly monitoring the cloud provider's security measures in an automated manner based on CSA formatting and specifications that are scheduled to be released in 2015. This level is still being developed at the writing of this thesis.

The CSA STAR certification address all the concerns identified in Section 5.1.2. Furthermore, since it is tailored for cloud environments, it does not suffer from the limitation faced by the CC scheme, that is, *Assurance of Deployed Software*. Even though CSA STAR scheme accommodates certification of *Composed Services*, it does not cannot cater to scenarios of run-time service composition. Moreover, the CSA STAR suffers from the same limitation of CC certificates, that is, they are the resulting certification documents, though structured, are represented in documents that are tailored for human consumption, and hence do not facilitate any automated reasoning to be performed.

5.2.5 Trusted Computing Approaches

In order to provide assurance to consumers on the confidentiality and integrity of their data on cloud infrastructures, there have been several approaches that have been proposed in the last years based on *Trusted Computing* approaches. Trusted Computing [138] ensures that the hardware and software behaves consistently as expected through a unique encryption key loaded on the hardware. This unique key is not accessible to the rest of system - hardware or software. Typically, trusted computing uses a hardware module called Trusted Platform Module (TPM) [84, 37], which contains the unique key. The TPM is used to provide the provide remote attestations by creating a unique hash key summary of the hardware and software configurations. By asking for remote attestations from the TPM, third parties can verify that the software configurations are not modified in an undesirable manner.

These approaches have been adopted to cloud computing [162] where the user can verify the software and hardware configurations on the remote nodes in the cloud provider infrastructure where the service is executed through remote attestations. There have approaches proposed [141], that involve using virtual TPMs that help in can be applied efficiently in cloud infrastructures.

However, in cloud environments, TPMs can provide assurance regarding computation and storage, but they cannot provide assurance that the cloud service itself is secure.

5.2.6 Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) [83, 119] is a policy-based technology infrastructure. It binds a public key issued to a particular user identity by a Certification Authority. The binding of the key to a user identity is done through a registration process which, based on different assurance levels, is completely automated or requires human supervision.

PKI is used to provide cryptographic services to users and it consists of the following components:

- Public-Key Certificates
- Public-Key Certificate Authority
- Registration Authority
- Certificate Practice Statement

- Certificate Revocation List

The public key certificate, also known as a Digital Certificate (DA) is a machine processable document used as a proof to establish the binding of a public key with a particular user. Public-key certification authorities are entities that are authorized to issue a DA after verifying the identity of the user. The CA uses its private key to sign the certificate that contains the public key of the user, hence the trust that consumers of a certificate depends on the trust they have on the CA. The registration authorities bind the individual to the certificate in order to ensure non-repudiation. Certificate practice statement is the process that is followed to issue a DA to a user by the certification authorities. The certificate revocation list maintains a list of certificates that have been revoked.

PKI, applied to cloud services, can provide assurance regarding the identity of the service provider, but they are not capable of offering assurance regarding the security of the service itself.

5.2.7 Various Trust Seals

TRUSTe [23] is a privacy certification scheme for web sites (and by extension to services). It provides assurance to consumers over the privacy of their data that is collected by the certified web sites [85]. TRUSTe validates the privacy policies of a web site (or service) and if it meets the criteria that is specified by TRUSTe, it proceeds to audit the web site (or service) implementation to verify whether the privacy policies are enforced. Web sites (or services), if they pass the audit, will be allowed to display the TRUSTe seal of approval on their websites and in cases of web services can claim to have TRUSTe certification.

However, there have been several criticisms for the TRUSTe scheme, in particular about the repeated privacy violations that have occurred in TRUSTe certified web sites [70]. One the other hand, the fact that TRUSTe certification body has limited power to force certified web sites that have violated the privacy guidelines is also a major drawback to this approach [117].

In addition to TRUSTe, there are similar privacy certifications that are currently used such as WebTrust [53], BBBOnline [36] among others.

In addition, the US-EU Safe Harbor [174] certification provides assurance to consumers in European Union to use United States based services that collect personal information of consumers. US-EU Safe Harbour certification evaluates whether a US based service provider complies with the EU data protection laws [67].

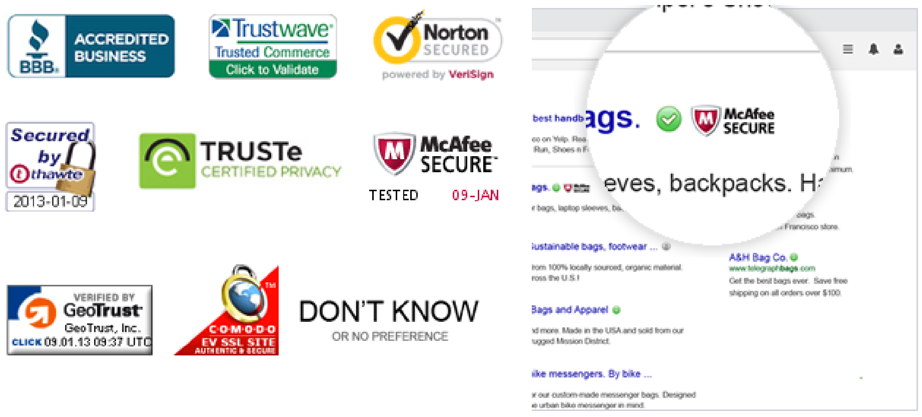

ACCREDITED BUSINESS
BBB
Trustwave Trusted Commerce Click to Validate
Norton SECURED powered by VeriSign
Secured by thawte 2013-01-09
TRUSTe CERTIFIED PRIVACY
McAfee SECURE TESTED 09-JAN
VERIFIED BY GeoTrust GeoTrust, Inc. CLICK 09.01.13 09:37 UTC
COMODO EV SSL SITE AUTHENTIC & SECURE
DON'T KNOW OR NO PREFERENCE

Figure 5.5: Few Security and Privacy Seals [110, 23]

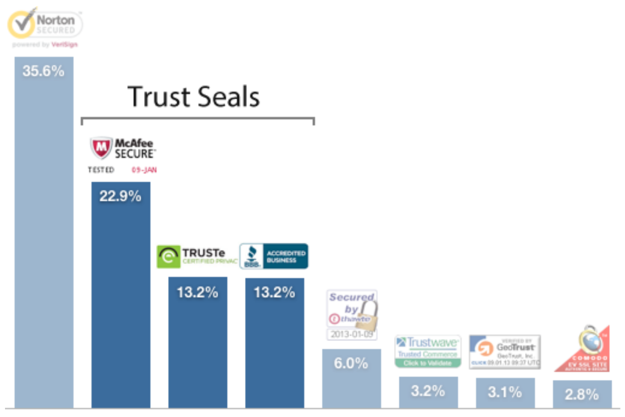


Figure 5.6: Survey of consumers trust in various online seals [20]

McAfee SECURE certification scheme is tailored for websites to verify whether a website is safe for visitors. McAfee uses proprietary scanning tools to scan a website for viruses, malware, phishing attacks, spam generation or other malicious activities. Once a website passes the scanning process, it is awarded the McAfee SECURE seal. This is displayed on the website, and is even integrated with McAfee SiteAdvisor software so that when users search for a website using popular search engines the McAfee SECURE seal appears next to the search results as shown in Figure 5.5. This provides assurance to consumers that the website is safe to browse.

However, McAfee SECURE does not provide any assurance to consumers on the data that is website. It cannot provide assurance that the data collected, processed and stored is done in a secure manner. Hence, the assurance gained from such this scheme is rather very limited.

In addition, there are other several very popular *SSL Seals* such as Norton Secured, Trustwave, Comodo among others that are used extensively by websites. In Figure 5.6, the results of a survey [20] conducted on consumer's trust on various online seals is shown. The survey was a comparison of different trust seals, and hence the results should be interpreted in that context.

5.3 Certification Schemes applicable to Services

In order to understand whether the different approaches presented in Section 5.2 can provide the required assurance to service consumers we have cross-checked the concerns in services against the assurance provided by different schemes. The security concerns in services presented in section 5.1.2 have been refined further to have a more fine-grained representation of concerns.

- **Lack of Transparency:** This can be further classified into:
  - ⋆ T_1: Lack of transparency regarding service architecture
  - ⋆ T_2: Lack of transparency regarding service internals
  - ⋆ T_3: Lack of transparency regarding security measures in services
- **Lack of Control:** This can be further classified into:
  - ⋆ C_1: Lack of control over the non-IT operational environment
  - ⋆ C_2: Lack of control over the IT environment
  - ⋆ C_3: Lack of control over the service implementation
- **Lack of Accountability:** This can be further classified into:
  - ⋆ A_1: Lack of accountability by service provider
  - ⋆ A_2: Lack of accountability by infrastructure providers

In addition, we have considered an usability issue regarding the certifications produced. In service environments, services are discovered and selected in an automated manner. Hence the certification documents should be able to participate in such process and enable consumers to make use of information contained in the certification documents during service discovery and service selection processes.

- **Usability:** This can be further classified into:

Scheme	T_1	T_2	T_3	C_1	C_2	C_3	A_1	A_2	U_1	U_2
SLAs	X	X	X	X	X	X	✓	X	X	X
ISO 27001	X	X	X	✓	X	X	✓	✓	X	X
CC	✓	✓	✓	X	X	✓	✓	X	X	X
CSA STAR	X	X	✓	X	*	✓	✓	X	✓	X
McAfee SECURE	X	X	X	X	X	X	X	X	X	✓
PKI	X	X	X	X	X	X	✓	X	✓	✓

*Planned, but not yet incorporated

Table 5.1: Security Assurances Approaches addressing concerns in SOA

- ⋆ U_1: Certificates should be structured
- ⋆ U_2: Certificates should be machine processable

We choose the most popular certification schemes and match the assurance provided by each scheme against the concerns listed above in Table 5.1.

As can be seen from the table, the assurance gained from SLAs mainly relates to accountability on the service provider in case of violations of the SLA. ISO 27001 scheme provides assurance on the non-IT operational environment as well as contribute towards accountability of the service provider. Typically, in cases, when the operational environment is not under the control of the service provider, they typically delegate the responsibility to the infrastructure provider who has control over the Operational environment and this is stated by the service provider.

It can be noted that CC is able to address most of the concerns, however, as discussed before, assurance over Operational Environment (IT and Non-IT) is not possible nor can assurance be gained over the infrastructure provider. Apart from the CC scheme, the CSA STAR scheme provides assurance that addresses most of the concerns in services. However, CSA STAR certification does not address the requirements for transparency regarding the architecture nor the service internals.

Interestingly we notice that no major security certification scheme produces documents that are machine processable apart from McAfee SECURE and PKI schemes which do not provide necessary security assurance needed by consumers.

Conclusion

In this chapter, we have discussed the explosive growth of cloud based services in the recent years. We explained the benefits as well as security concerns of using cloud based services. We discussed the various means through which security assurance can be gained and concluded that CC seems the most viable option for providing the maximum assurance in cloud environments. We discuss the fact that no no major security certification scheme produces machine processable certificates and how it can hinder an wider adoption of security certifications in cloud. In addition, a major drawback of CC and CPA, the most suitable schemes for service security certification, is that the certificates, currently, are a point-in-time, that is even if a service violates the certified properties, the certification still holds as the authorities do not have any means to detect such violations.

In the next chapters, we present the core contributions of this thesis to the security certification landscape:

1. Digital *Security* Certificate – which is a generic, machine processable, *descriptive*, security certificate that can be adopted by the different security certification schemes presented in this chapter.

2. Dynamic Security Certificate Lifecycle – a framework that provides assurance over the IT aspects of the operational environment.

Publications

The contents of this chapter have been discussed in the following publications:

⋆ **S. P. Kaluvuri**, M. Bezzi, and Y. Roudier. *Bringing Common Criteria certification to web services*. In Services (SERVICES), 203 IEEE Ninth World Congress on, pages 98–102. IEEE, 2013.

⋆ **S.P Kaluvuri**, M. Bezzi, A. Sabetta, Y. Roudier, R. Menicocci, V. Bagini, A. Ricardi, M. Orazi. *Applying Common Criteria to Service Oriented Architectures* . International Common Criteria Conference 2012, Paris.

Part II

Applying Security certification to SOA

Chapter 6

Digital Security Certificate for Services

"Without tradition, art is a flock of sheep without a shepherd. Without innovation, it is a corpse."

— Winston Churchill

In this chapter, we present a scenario of a cloud storage service which we use as a running example throughout the rest of this chapter. We explain the requirements for a digital security certificate in service environments. We propose a conceptual model for a *Digital Security Certificate* and present a language that is based on the conceptual model to represent the security certificates of services. We discuss the capabilities and limitations of our model. In order to overcome some of the limitations of the Digital Security Certificate, we introduce the concept of a *Digital Security Certificate Profile*, that helps various certification schemes to adopt our model to represent the security certificates resulting from their certification processes.

6.1 Requirements for Service Security Certificates

In this section, we analyse and identify the requirements for a security certificate in service environments. In a service environment, consumers should be able to compare the (certified) security features of a service with their security requirements and in addition to compare the (certified) security features of service offerings from different service providers. But security certificates that are represented in natural language, filled with legalese and are rather unstructured, prevent any sort of automated reasoning to be performed on them.

In order to bring security certification to the service environments in a manner that they can play a constructive role in the service selection and service consumption process, several modifications to the current state of the art are necessary [104]. Among them, we focus on the aspects related to the security certificate generation and representation. In this regard, we have identified the following key requirements from a service consumer perspective, that need to be addressed in order to facilitate security certificate adoption in service landscape.

Requirement 1. *The security certificates must be machine processable in order to allow automated reasoning to be performed on them.*

Requirement 2. *Security certificates should contain enough information about the certified entity so that they can cater to consumers with varying levels of security knowledge, such as regular users with limited security understanding to security experts of organizations. In other words, it is necessary that the certificates are descriptive [176], meaning, that they describe with sufficient details the security features of their services, together with supporting evidences.*

Requirement 3. *Mechanisms must exist in order to bind a service and its security certificate, given that a service implementation can change, while maintaining the same external interface or API. Consumers would need to have trustworthy and dynamic means to verify whether a service implementation they are using is the certified one.*

There are many different security certification schemes that *are* being applied in cloud environments such as CSA STAR, ISO 27001 and those that *could* be applied such as CC, CPA and so on. The security certificates should be able to represent information coming from these diverse certification processes.

It is essential to have a standardized representation for a security certificate as it is a first step towards facilitating reasoning between different security certification schemes. Moreover, having a standardized representation of security certificates of different schemes is a huge advantage for the proliferation of security certificates in services as it makes it easier for developers to integrate reasoning on security certificates while using service solutions in a standard manner. This is another key requirement to be addressed.

Requirement 4. *A security certificate should be certification scheme independent, so that different certification schemes could generate security certificates that are represented in a standard manner.*

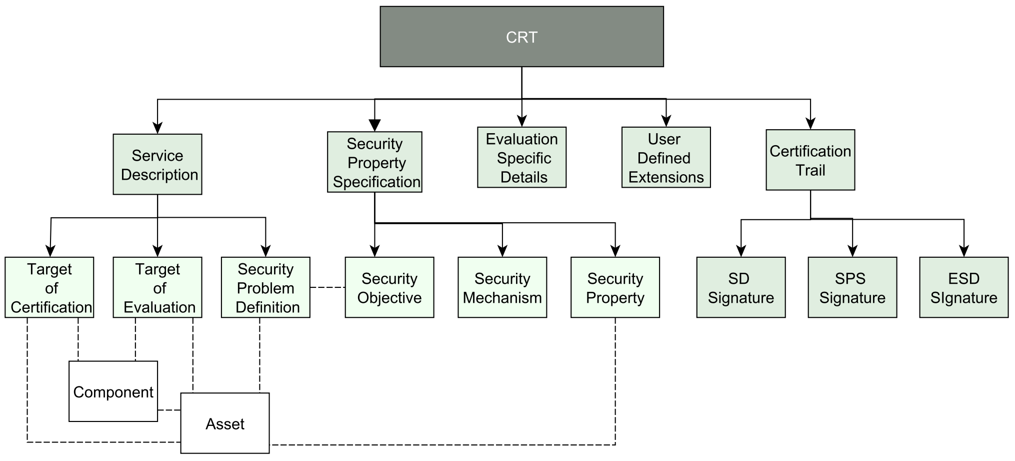


Figure 6.1: Conceptual Model for a Digital Security Certificate

6.2 Conceptual Model

We present our proposal for a digital security certificate concept (from now on referred as $\mathcal{CRT}$), that in our view addresses the mentioned security certification issues in SOC. In fact, $\mathcal{CRT}$s are designed to be machine readable (and thus addressing Requirement 1). They are designed to cope with existing web service interaction models and standards, thus to ease their adoption and integration in SOC. Moreover, $\mathcal{CRT}$s are designed to be completely descriptive and thus allowing automated reasoning upon them (to cope with Requirement 1). In particular, $\mathcal{CRT}$s can provide evidence of the presence and implementation of a service's security features, that are collected through a service evaluation phase; this permits further customer analysis, with respect to specific security requirements (Requirement 2). The binding between a service implementation and its security certificate (Requirement 3), is discussed in the next chapter (Chapter 7, while in the current form of $\mathcal{CRT}$ we provide a specific element to contain information that helps in ensuring the binding. Finally, the security certificate representation is designed to be broad enough to capture information coming from different certification schemes and we introduce the concept of a *Digital Security Certificate Profile* that allows certification authorities and/ or service consumers to be able to specify requirements on a $\mathcal{CRT}$ - both semantically and syntactically (Requirement 4).

The conceptual model for the $\mathcal{CRT}$, is designed to capture information emanating from security certification processes. In particular, we have considered the CC scheme, as it is the most broadest scheme currently. The

CC-ST, which is the descriptive part of the CC scheme, serves as a foundation for our $\mathcal{CRT}$.

However, we have extended this significantly, in order to make it machine processable (**R1**) and suitable for service-specific needs. In contrast to CC, and other existing certification schemes, the digital security certificate is designed to be completely descriptive [176] (**R2**), and hence it contains the description of the certified entity, the security properties of the certified entity, the evaluation details that underpin the certified properties.

Definition 1. *The digital security certificate is a tuple* $\mathcal{CRT} = \langle \mathcal{SD}, \mathcal{SPS}, \mathcal{ESD}, \mathcal{UDE}, \mathcal{CT} \rangle$ *where,* $\mathcal{SD}$ *is the service description,* $\mathcal{SPS}$ *is the security property specification,* $\mathcal{ESD}$ *is the evaluation specific description,* $\mathcal{UDE}$ *is User Defined Extensions and* $\mathcal{CT}$ *is Certification Trail.*

The $\mathcal{SD}$ provides details about the service and its underlying architecture, thereby mitigating the concerns of the consumers on the lack of transparency of services since services just expose an interface and the internal dynamics of the service and its architecture are hidden from the consumer. The $\mathcal{SPS}$ provides details about the security properties of the service at varying levels of abstraction, thereby catering to consumers with varying levels of security knowledge and requirements. The $\mathcal{ESD}$ provides details regarding the evaluation process and its results, such as the test suites that were executed or the formal models and proofs used to verify and validate the security properties of the service. The *User Defined Extensions* ($\mathcal{UDE}$) can be either used by the service providers to disclose any additional information and/ or by the certification authorities to state any further criteria. The *Certification Trail* ($\mathcal{CT}$) contains the *digital signatures* of entities that participate in the certification process and especially the entities that *approve* different elements in the certificate such as approving the $\mathcal{SD}$ by a security expert in the evaluation lab. Certification Trail brings accountability in the certification process, where if a service was found to be vulnerable after the certification process and it was found to be an error during the testing of the service, it can be traced back to the evaluation lab or the expert who has approved an poorly tested service. These five elements serve different purposes and together contribute in providing assurance on the security of the service.

6.2.1 Service Description

In (CC-ST), the assets are described in natural language and no identifiers are provided for them; therefore, an explicit link cannot be made between the security properties and the assets that they secure. In order to overcome this we adopt a asset-centric approach with explicit references between the assets and the different elements in the certificate.

Definition 2. *An Asset, a, is an entity that is of some value to the consumer or the provider. Assets can be data, applications, the IT equipment on which the service operates or even users of the Information System.*

The CC-ST contains the Target of Evaluation (TOE) that describes the system that is being certified and the the boundaries of the evaluation are indicated, albeit in an ad-hoc manner. However for a machine readable certificate there should be a clear distinction between the system that is being certified and the aspects of the system that are subject to evaluation. It is of utmost importance in service based systems, due to the fact that services can be easily composed of external services and this information should be a part of the service description but clearly marked as outside the scope of evaluation.

The TOE in a CC-ST also contains the system architecture, the different components that compose the system among other information such as configuration in which the system is evaluated, the underlying IT architecture etc., and this is represented in natural language accompanied by architecture diagrams. This poses another issue in representing the TOE in a machine processable manner. So in order to address these two issues, we introduced an element called Target of Certification ($\mathcal{TOC}$) that describes the service being certified, in addition the TOE which describes the part of the Target of Certification that is evaluated.

Definition 3. *A Target of Certification is a tuple $\mathcal{TOC} = \langle \mathcal{ACI}, \mathcal{DM}, \mathcal{TT}, \mathcal{TH} \rangle$, where $\mathcal{ACI}$ is Asset-Component Identification, $\mathcal{DM}$ is the Deployment Model, $\mathcal{TT}$ is the TOC Type and $\mathcal{TH}$ is the TOC Cryptographic Hash.*

Definition 4. *An Asset-Component Identification is a tuple $\mathcal{ACI} = \langle \mathcal{A}, \mathcal{C}, \alpha \rangle$, where $\mathcal{A}$ is a set of all the assets identified for the TOC, $\mathcal{C}$ is a set of all the components in the TOC and $\alpha \subseteq \mathcal{A} \times 2^{\mathcal{C}}$ maps each Asset with a set of Components.*

Definition 5. *A Deployment Model ($\mathcal{DM}$) is a set of software configurations ($sc \in \mathcal{DM}$) and the software configuration is a tuple $sc = \langle \mathcal{S}_s, \mathcal{SC}_c \rangle$, $\mathcal{S}_s$ identifies the software component in the deployment environment and $\mathcal{SC}_c$ is a set of all allowed configurations for the software component.*

Definition 6. *A software component configuration $sc_c \in \mathcal{SC}_c$ is a key-value pair $sc_c = \{k, v\}$ that identifies the configuration key and configuration value respectively.*

Definition 7. *The $\mathcal{TOE}$ is a subset of the Asset-Component Identification.* $\mathcal{TOE} \subseteq \mathcal{ACI}$

The TOC *Components* are an integral part of the $\mathcal{TOC}$ as they allow the $\mathcal{TOC}$ to be expressed in a modular and structured manner. It comprises an abstract model of the Component, the *Component Model*: it can be as simple as just containing the interfaces of the component, or a more detailed specification of the internal dynamics of the component as deemed sufficient by the *Certification Authorities*. It must also contain technical specifications of the Component, again at the level of abstraction as deemed sufficient.

Definition 8. *A Component is a tuple $\mathcal{C} = \langle \mathcal{C}_{id}, \mathcal{CM} \rangle$, where $\mathcal{CM}$ is the component model and $\mathcal{C}_{id}$ is the component identifier.*

In the current security certification schemes the target of evaluation, which is basically the system being certified, is described in natural language and in an ad-hoc manner. There is no explicit description of the system internals nor its internal dynamics in a concrete manner. At most the system description is given at a very abstract and high level point of view. This creates a situation where the system being certified along with the security mechanisms that are implemented in the system are described in a manner that is completely detached from the actual system.

In order to overcome this, a novel means to represent the system that is being certified that ties to the actual implementation of the system is needed. The thesis contributes in this area, by proposing a flexible and extendible model to describe the service components.

The aim of this model is to capture the system design, interactions and information flow between the components.

Definition 9. *The component model is a tuple $\mathcal{CM} = \langle \mathcal{AT}, \mathcal{OP}, \mathcal{INT} \rangle$, where $\mathcal{AT}$ is the set of Attributes, $\mathcal{OP}$ is the set of Operations, and $\mathcal{INT}$ is the Interaction Model.*

This facilitates encapsulating different facets of a component in one model, thereby having an integrated structure to capture the structure, and the interactions.

Definition 10. *An Attribute is a tuple $\mathcal{AT} = \langle A_n, A_t, A_d \rangle$, where A_n is the Attribute, A_t is the Attribute type, while the A_d is the Attribute data type.*

Definition 11. *An operation is a tuple $\mathcal{OP} = \langle OP_n, \widehat{\mathcal{AT}}, OP_d \rangle$, where OP_n is the Operation, $\widehat{\mathcal{AT}} \subseteq \mathcal{AT}$ is the Input attributes for the Operation, while the OP_d is the Operation return data type.*

The interaction model captures the invocations made by each method in the Component to other methods. Each interaction contains the package, the class in which the invoking method is in as well as the package, the class of the invoked function.

Definition 12. *An interaction model ($\mathcal{INT}$) is a set of invocations where each invocation is a tuple, defined as $\mathcal{INV} = \langle P_f, C_f, M_f P_i, C_i, M_i \rangle$, where P_f represents the package of the invoking function, C_f represents the class of the invoking function, M_f represents the invoking method, P_i represents the package of the invoked function, C_i represents the class of the invoked function, and M_i represents the invoked function.*

Here, we assume that the services are developed in Object Oriented Programming languages and thus the invocation model is designed to capture such interactions. However, conforming to this particular model is not a constraint that is forced on the $\mathcal{CRT}$ artefact.

A *Security Problem Definition* (spd) is essential in a security certificate as it provides the rationale for securing the assets. The rationale for securing the assets can stem from the threats that are identified for the assets by the service provider or from the service provider's security policy (which in turn could be due to compliance to regulations etc.).

Definition 13. *The security problem definition is a tuple $spd = \langle \hat{\mathcal{A}}, spr \rangle$, where $\hat{\mathcal{A}} \subseteq \mathcal{A}$ is a set of assets that need to be secured and spr is a security problem rationale for securing the assets.*

Definition 14. *The security problem rationale (spr) is a union of threats $\mathcal{T}$ and service provider's security policy $\mathcal{SSP}$. $spr = \mathcal{T} \cup \mathcal{SSP}$*

The service description must contain the description of the certified system, the part of the system that is evaluated and the rationale for protecting the assets that are identified.

Definition 15. *The Service Description is a tuple $\mathcal{SD} = \langle \mathcal{TOC}, \mathcal{TOE}, \mathcal{SPD} \rangle$ where, $\mathcal{SPD}$ is the set of security problem definitions(spd).*

6.2.2 Security Property Specification

The CC-ST contains a vast amount of information but is often presented in heavy-jargon; this rarely allows a consumer (a non security expert) to get a high level perspective of the security features provided by the software/ service. Hence we introduced a new element in the $\mathcal{CRT}$ model called as "security property specification" which enables a fine grained description of the security property that originates from a multi-layered model. It comprises of different elements, from abstract security properties to concrete security mechanisms.

Definition 16. *An* Abstract Security Property $\hat{p}$ *is an atomic security attribute for an asset.*

For example, abstract security properties can be confidentiality, integrity, availability, authenticity, non-repudiation, utility, privacy and so on.

Since abstract security properties by themselves do not convey any information on how the property is applied, there is a need for contextual information. Hence we define *Contextual Security Property*.

Definition 17. *A Contextual Security Property is an abstract security property realized in a certain context.* $\hat{p_c} = \langle \hat{p}, c \rangle$ *where* c *is a context.*

Contexts depend on the abstract security property. Abstract security properties that are data centric such as the *CIA triad* can have contexts such as transit, rest and usage. Such as *Confidentiality in rest*, *Integrity in transit* and so on. However, these properties still lack the subject, i.e., no indication of "what" is being secured. This is addressed by the certified security property.

Definition 18. *A certified security property,* p*, is a contextual security property (* $\hat{p_c}$*) applied on a set of assets (*$\hat{A}$*).* $p = \hat{p_c} \times \hat{\mathcal{A}}$

The (certified) security property provides a high level overview of how an asset is secured. But this does not provide any information on how the $\mathcal{SPD}$ are addressed. This is addressed by using the concept of "Security Objectives" similar to the CC scheme. A security objective, so, counters, mitigates or detects a spd that is identified for the $\mathcal{TOE}$ and contributes to the realization of a security property p for the TOE.

Definition 19. *A security objective is a tuple* $so = \langle \mathcal{O}, \mathcal{OT}, \widehat{\mathcal{SPD}} \rangle$*, where* $\mathcal{O}$ *is the objective,* $\mathcal{OT}$ *is the objective type,* $\widehat{\mathcal{SPD}} \subseteq \mathcal{SPD}$ *is a set of security problem definitions.*

The $\mathcal{OT}$ is an enumerated type and contain one of the three values: *1)* O_{toe}, which indicates that the SO is targeted for the TOE component; *2)* O_{it} indicates that the SO is targeted for the IT Operational Environment on which the service is running; *3)* O_{nit} indicates the SO is targeted for the Non IT Operational environment.

All the security objectives are necessary and sufficient conditions to realize the security property. In other words, a TOE can have a security property p if and only if all the security objectives for the TOE are satisfied. Security Objectives are realized by security mechanisms that should be implemented in the TOE.

A Security Mechanism, sm, is an action, device, procedure, or a technique that meets or opposes (counters) a threat or an attack by eliminating or preventing it, by minimizing the harm it can cause or by discovering and reporting it so that corrective action can be taken. Security mechanisms refer to the security objectives that they satisfy, and they can be mapped to specific functional criteria of the certification scheme that is issuing the certification scheme.

Definition 20. *A security mechanism is a tuple $sm = \langle \mathcal{M}, \mathcal{SFC}, \widehat{\mathcal{SO}} \rangle$ where, $\mathcal{M}$ is the mechanism that is implemented, $\mathcal{SFC}$ is the security functional criteria of a certification scheme and $\widehat{\mathcal{SO}} \subseteq \mathcal{SO}$ is a set of security objectives that the mechanism realizes.*

Definition 21. *A Security Property Specification is a tuple $SPS = \langle \mathcal{P}, \mathcal{SO}, \mathcal{SM}, \gamma, \eta \rangle$ where $\mathcal{P}$ is a set of certified security properties, SO is a set of Security Objectives, SM is a set of Security Mechanisms , $\gamma \subseteq \mathcal{P} \times 2^{\mathcal{SO}}$ maps each security property to a set of security objectives and $\eta \subseteq \mathcal{SO} \times 2^{\mathcal{SM}}$ maps each security objective to a set of security mechanisms.*

This fine grained representation has two major advantages: allows consumers with varying security understanding to gain understanding of the security features provided by the service (security properties to security mechanisms); allows the certified security property to be machine processable that enables consumers to easily search for services that match their security requirements.

6.2.3 Evaluation Specific Details

The $\mathcal{ESD}$ defines the representation of the details and results of the service evaluation process needed to support the certified security property. We identified two different categories for evaluation of services: Evaluation through testing [9, 10], and Evaluation through formal analysis [71]. Given that these different types of evaluation approaches in very heterogeneous models and results, we refrain from modelling them at a conceptual level. We rather allow many different evaluation models and results to be plugged into the $\mathcal{ESD}$ element in the broad categories mentioned before.

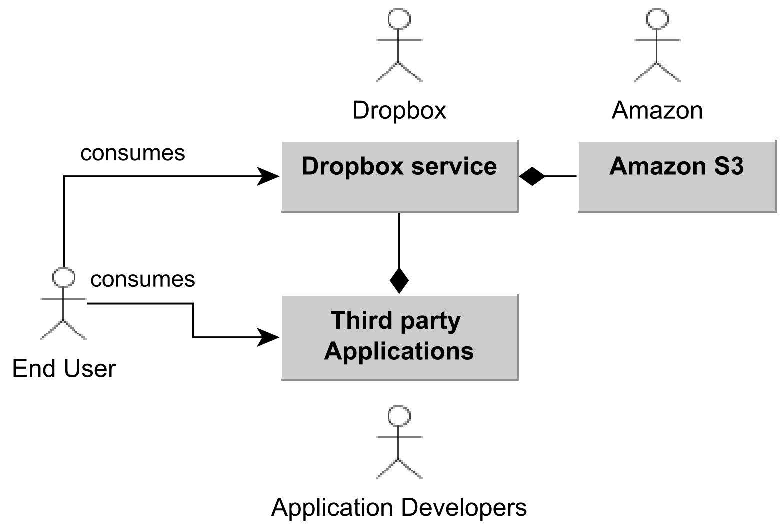


Figure 6.2: Dropbox Overview

6.3 Scenario

Consider the example of the cloud storage service *Dropbox* [61]. It is one of the most widely used cloud storage service solutions currently. There were several security and privacy concerns over how *Dropbox* handles its clients' [31, 157] data and vulnerabilities [42] that were found in *Dropbox* service, some of which have been actually acknowledged by *Dropbox* itself [63]. In order to allay those concerns and provide updated information on the security practices followed by the service, *Dropbox* clearly states its security policy highlighting the measures taken to secure the clients' data as well as the providing information on how the data is handled and stored.

The security policy explicitly discloses that *Dropbox* makes use of another widely used cloud storage service in the backend, i.e., *Amazon Simple Storage Service* (Amazon S3) [159], to store all its clients' data. However, *Dropbox* clearly states that it encrypts its consumers' data before sending the data to *Amazon S3*. Encryption mechanisms allow the confidentiality of *Dropbox*'s consumers' data to be preserved even when the data itself is stored externally from *Dropbox*'s servers.

Dropbox depends on certain security features to *Amazon S3* such as perimeter security - which secures the premises of the data centres where the data of *Amazon S3* is stored. Dropbox gains assurance regarding the security features that it depends on *Amazon S3* through the various secu-

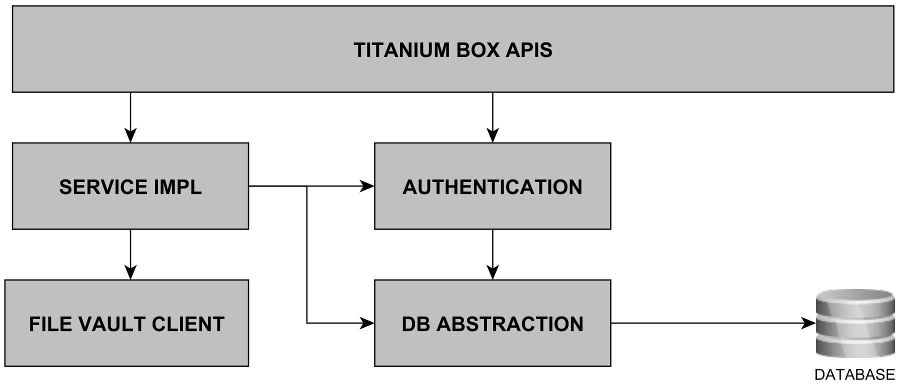


Figure 6.3: Titanium Box - Architecture Overview

rity certifications of *Amazon S3* such as ISO 27001 certification, SSAE 16, US Health Insurance Portability and Accountability Act (HIPAA) among others [6].

Dropbox also exposes an API to third party applications that, with due prior authorization from the consumers, permits access to the third party applications to the consumer's data stored in their *Dropbox* account. An example of such service is an image processing service that accesses the consumer's photos stored in their *Dropbox* account and process those images (such as resizing, applying filters, and so on) and store them again as shown in Figure 6.2. Such applications can be extensively found in the mobile domain (Google Play, Apple AppStore, Windows Phone Store). Assuming that the *Dropbox* service undergoes security certification using one of the available schemes (such as CC), we examine whether the information emanating from the certification process can be captured in a machine processable format.

However, since the source code of *Dropbox* is not available publicly, we have developed a service, which we refer as "TitaniumBox", that is similar to *Dropbox* both in terms of functionality and architecture. Though it shares several similarities with *Dropbox*, few assumptions have been made where information was not available regarding *Dropbox* implementation and these are stated clearly.

6.3.1 TitaniumBox - A cloud storage service

The major difference of *TitaniumBox* with *Dropbox* is that it has been implemented in *JAVA*, running on an Apache Tomcat Server [11] using Jersey Framework [135] to provide RESTFul web service, where as *Dropbox*

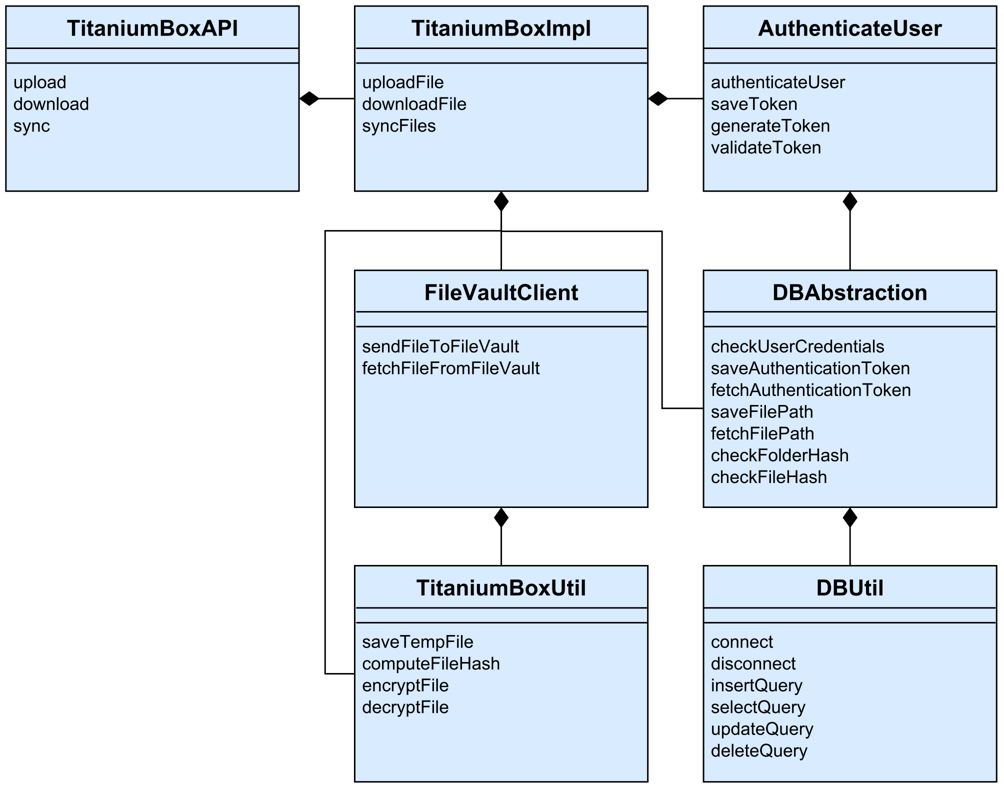


Figure 6.4: Titanium Box - Class Diagram

is developed using Pyston [62] which is JIT-based Python implementation.

A service similar to *Amazon S3* has been developed, called “File Vault”, that acts as a storage service without the synchronizing and versioning features that are offered by TitaniumBox. The *FileVault* service is treated as a black box, similar to how *Dropbox* treats *Amazon S3*, and is assumed to have all the security controls that are put in place by *Amazon S3* such as perimeter security, data redundancy etc. In the use case, only the interfaces for the *File Vault* service are shown and used within the TitaniumBox Service to store and retrieve files.

A high level overview of the TitaniumBox service is shown in Figure 6.3. The *Titanium Box APIS* component that exposes RESTFul Interfaces to consumers. It makes use of the *Service IMPL* component that contains the application logic. The *Service IMPL* component invokes the *AUTHNENTICATION* component to perform request authentication and authorization operations. The *DB ABSTRACTION* component connects to a database to perform any operations that require data persistence. A more detailed class diagram can be found in Figure 6.4 and the APIs exposed by TitaniumBox are shown in Listing 6.1 - 6.3, while the general implementation overview is shown in Listing 6.4 and Listing 6.5

Listing 6.1: **TitaniumBox Upload Interface**

```java
        @POST
        @Path("/upload/{authKey}")
        @Consumes(MediaType.MULTIPART_FORM_DATA)
        @Produces("application/json")
        public Response upload(
                        @PathParam("authKey") String token,
                        @FormDataParam("request") InputStream
                            requestStream,
                @FormDataParam("request") FormDataContentDisposition
                    requestDetail){
                TitaniumBoxImpl impl=new TitaniumBoxImpl();
                return
                    impl.uploadFile(requestStream,requestDetail,token);
        }
```

Listing 6.2: **TitaniumBox Download Interface**

```java
        @GET
        @Path("/download/{fileID}/{authkey}")
        @Produces("text/plain")
        public Response download(
                        @PathParam("fileID") String fileID,
                        @PathParam("authkey") String token){
                TitaniumBoxImpl impl=new TitaniumBoxImpl();
                return impl.downloadFile(fileID,token);
        }
```

Listing 6.3: **TitaniumBox Sync Interface**

```java
        @GET
        @Path("/sync/{hashValue}/{authkey}")
        @Produces("text/plain")
        public Response sync(
                        @PathParam("hashValue") String hashValue,
                        @PathParam("authkey") String token){
                TitaniumBoxImpl impl=new TitaniumBoxImpl();
                return impl.syncFiles(hashValue,token);
        }
```

Listing 6.4: **TitaniumBox Upload Implementation**

```
        public Response uploadFile(InputStream requestStream,
                       FormDataContentDisposition requestDetail,
                           String token) {
                AuthenticateUser auth=new AuthenticateUser();
                int userID=auth.validateToken(token);
                if(userID<1)
                        return
                           Response.status(401).entity("Unauthorized
                           Request").build();
                TitaniumBoxUtil tUtil=new TitaniumBoxUtil();
                String tmpFile=tUtil.saveTempFile(requestStream,
                    requestDetail.getFileName());
                FileVaultClient fclient=new FileVaultClient();
                String response=fclient.sendFileToFileVault(tmpFile);
                return Response.status(201).entity(response).build();
        }
```

Listing 6.5: **TitaniumBox Download Implementation**

```
        public Response downloadFile(String fileID, String token) {
                AuthenticateUser auth=new AuthenticateUser();
                int userID=auth.validateToken(token);
                if(userID<1)
                        return
                           Response.status(401).entity("Unauthorized
                           Request").build();
                String filePath=DBAbstraction.getFilePath(
                        Integer.parseInt(fileID),userID);
                FileVaultClient fclient=new FileVaultClient();
                File file=fclient.fetchFileFromFileVault(filePath);
                ResponseBuilder response=Response.ok((Object) file);
                response.header("Content-Disposition",
                        "attachment;filename="+file.getName());
                return response.build();
        }
```

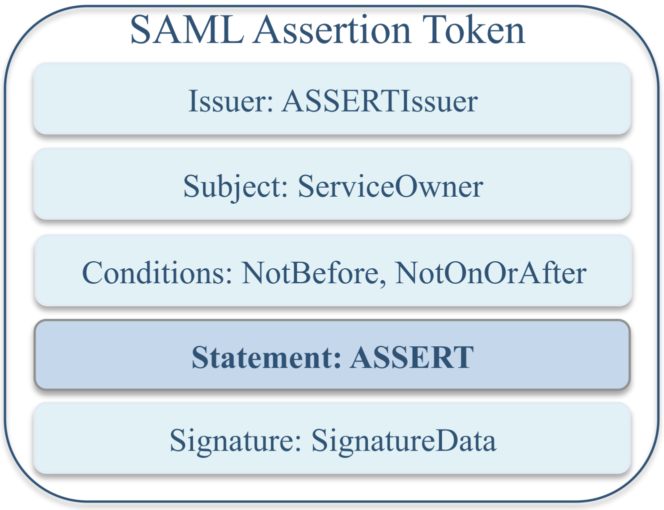


Figure 6.5: SAML Assertion Token as Container of `ASSERT` Data

6.4 Realization of the conceptual model

In order to realize the conceptual model of the digital certificate $\mathcal{CRT}$, we have developed an XML-based language that enables the representation of the certificate in a machine processable form, which from henceforth we refer to as an `ASSERT`. A detailed version of the schema can be found in [100] and in this section we will explain its most relevant elements using the example introduced in Section 6.3.1.

6.4.1 Container of Digital Security Certificate

The management and exchange of the `ASSERTS` is an important consideration for a successful implementation of a certification ecosystem life-cycle, i.e., production, maintenance, consumption of certificates. In this context, the container of the `ASSERTS` assumes significant importance as it is needed to encapsulate the certificate data into an interoperable format that can be used with existing web service standards and technologies.

In order to facilitate an easier and faster adoption of $\mathcal{CRT}$s in the *SOC*, we choose to use the existing standards as *Containers* for the $\mathcal{CRT}$. In this regard, we have considered the *digital certificate* standards, that are primarily used for identity and authorization management, as possible candidates given their widespread usage and acceptance. Standards such as X.509 [178], SAML [149], SPKI/SDSI [167], and Kerberos [97] are well known and among those, X.509 and SAML are the most widely used in practice. Both standards support public-key (identity) certificates and attribute certificates for purposes of user authentication and authorization. The *attribute* certificates of X.509 and SAML standards support extensibility of the attribute part of the certificate to accommodate domain-specific data. This aspect makes both standards suitable to provide a PKI-compliant container for encapsulating the content of $\mathcal{CRT}$s.

However, we have chosen the SAML standard [149] as a container because it is widely used in decentralized systems for its support for request and exchange of "SAML Assertions", be that for authentication or authorization of entities, or any attributes of an entity. The SAML standard has support for several standard profiles for usage of SAML tokens in specifications such as WS-Security [128], WS-SecurityPolicy [129], WS-Trust [130], etc. These aspects make SAML a good choice to be a container for exchanging `ASSERTS` in service environment. We use the SAML Assertion tokens to encapsulate `ASSERT`-specific data. We use the SAML Assertion tokens to encapsulate `ASSERT`-specific data.

Figure 6.5 shows the main elements of the SAML assertion token structure where the `<Statement>` element defines an abstract statement of an assertion. We extended this element (similar to how SAML authentication and authorization decision statements extend the abstract <Statement> element) to provide a statement about a service's description, its security property along with the corresponding evidence. The standard field *Issuer* in the SAML token is used as a means to capture the `ASSERT` Issuer's identity (the certification authority issuing the `ASSERT`). The *Subject* field represents the identify of the certificate requester, which in most cases will be the service provider. And the validity conditions and the signature data are inherent to all security tokens.

6.4.2 ASSERT Structure

Listing 6.6 shows main elements of the `ASSERT` structure. It has three major elements: *ASSERTCore*, *ASSERTTypeSpecific* and *UserDefinedExtensions*. The *ASSERTCore* part contains elements that are independent of the evaluation of a service, i.e. the $\mathcal{SD}$ and the $\mathcal{SPS}$ elements in the conceptual model. The evaluation information in the conceptual model, i.e. $\mathcal{ESD}$, is contained in the *ASSERTTypeSpecific* element, while the $\mathcal{UDE}$ is captured in its namesake element *UserDefinedExtensions*.

Listing 6.6: **ASSERTS captured as SAML Assertions**

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:Assertion ID="ID_6c5d94ff-f4f5-4321-ba6e-dbab10ae5f0f"
    IssueInstant="2014-10-12T13:54:09.567+02:00" Version="2.0"
    xmlns:ns2="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ns3="http://www.w3.org/2001/04/xmlenc#"
    xmlns:ns4="http://www.w3.org/2000/09/xmldsig#" xmlns:ns5="urn:assert4soa:assert:2.0">
    <ns4:Signature/>
    <ns2:Subject/>
    <ns2:Conditions NotBefore="2014-08-07T11:24:47.152+02:00" NotOnOrAfter="2014-08-07T11:25:12.061+02:00"/>
    <ns2:Statement Version="3.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns5:ASSERTSAMLAssertionStatementType">
        <ASSERTCore>
            <TargetOfCertification/>
            <SecurityProblemDefinition/>
            <CertificationCriteria/>
            <PerformedBy/>
            <EvaluationData/>
            <SecurityProperty/>
            <ServiceBinding/>
            <ASSERT4Humans/>
            <ASSERTSigner/>
        </ASSERTCore>
        <ASSERTTypeSpecific>
            <Property/>
            <ServiceModel/>
            <Results/>
        </ASSERTTypeSpecific>
        <UserDefiniedExtensions/>
        <CertificationTrial/>
    </ns2:Statement>
</ns2:Assertion>
```

ASSERT Core

The *ASSERTCore* element contains, in addition to the $\mathcal{SD}$ and $\mathcal{SPS}$, elements such as *ServiceBinding* that provides a robust link between the service and its ASSERT, *CertificationProcess* that provides information related to the certification process of a given service, and a textual description of the certificate in the *Assert4Humans* element where the certified service and the certified property are explained in natural language for end-user comprehension. The *AssertSigner* element identifies the entity that signs the ASSERT, while the *PerformedBy* element in the *CertificationProcess* identifies the entity who performed the service evaluation.

Since multiple entities can be involved in a certification process, for example the ASSERT issuing process and service evaluation process may be undertaken by different entities, we provide this feasibility so as to increase the accountability during the production of certificates. In order to better illustrate the ASSERT language we provide code excerpts from the ASSERT of the example we provided in Section 6.3.1.

Service Description in ASSERT Core: The $\mathcal{SD}$ in the conceptual model is mapped to the *CertificationProcess* element in the ASSERT language. It contains the elements such as *TargetOfCertification* and *SecurityProblemDefinition* which map to the $\mathcal{TOC}$ and $\mathcal{SPD}$ respectively in the conceptual model. In addition, we have incorporated an element called *CertificationCriteria* used to represent any specific criteria followed during the service certification process (e.g., compliance to regulations).

Listing 6.7: **Target of Certification**

```xml
<TargetOfCertification Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#Software-as-a-service">
    <Assets>
        <Asset ID="A_REQUEST" Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InputParameter">
            <Name Format="FormDataContentDisposition">requestDetail</Name>
            <APIBinding>WIDL/upload/requestDetail</APIBinding>
            <TOCComponents>
                <TOCComponent TOCComponentRef="C_TITANIUMBOXAPI"/>
            </TOCComponents>
        </Asset>
    </Assets>
    <TOCComponents>
        <TOCComponent ID="C_TITANIUMBOXAPI" InTargetOfEvaluation="true"/>
        <TOCComponentModel/>
    </TOCComponents>
    <DeploymentModel/>
</TargetOfCertification>
```

The *TargetOfCertification* element is depicted in Listing 6.7. The elements in the $\mathcal{ACI}$ are represented directly in the *TargetOfCertification* element i.e., the *Assets*, *TOCComponents*. It also contains the *Type*, *DeploymentAndImplementaionModel* and *Description* providing textual description of the *TargetOfCertification* for end-user comprehension. We enforce the explicit identification of both the *Assets* and *TOCComponents* by making the use of the *ID* element mandatory. The set that maps assets with components, α, in the $\mathcal{ACI}$ is realized within the asset definition by mapping each asset to specific components using the *TOCComponentRef* (which is of type *IDRef*) to provide a binding between the assets and components.

The TOE is not represented as an explicit part of the service description in the *ASSERTCore*, but we use the flag *InTargetOfEvaluation* in the *TOCComponent* element that indicates whether the component is a part of the TOE, and avoids a duplicate representation of the components in both the TOE and TOC to have an optimized ASSERT.

The *SecurityProblemDefinition* element in the *ASSERTCore* contains a list of *ProblemDefinition* as shown in Listing 6.8. Each *ProblemDefinition* is mapped to the spd in the conceptual model.

Listing 6.8: **Security Problem Definition**

```xml
<SecurityProblemDefinition>
    <ProblemDefinition ID="SPD_DISCLOSURE">
        <Assets>
            <Asset AssetRef="A_REQUEST"/>
        </Assets>
        <Rationale>
            <Threats>
                <Threat ID="T_INFDISCLOSURE">
                    <Name>INFORMATION_DISCLOSURE</Name>
                    <Value/>
                </Threat>
            </Threats>
        </Rationale>
    </ProblemDefinition>
</SecurityProblemDefinition>
```

Security Property Specification in ASSERT Core: The *SecurityProperty* element maps to the p element in the conceptual model.

However, on the representation (language) level we have defined a single property certified in ASSERT. Such “separation” of certified properties allows us to have practical implications on management of ASSERTs throughout their life-cycle, such as issuance, consumption (reasoning), and revocation of ASSERTs. For example, if an ASSERT certifies two properties, say “ confidentiality in transit” and “confidentiality in storage”, and during the ASSERT lifetime the given service does not any more comply/ provide “confidentiality in transit” due to some technical reasons, the certification authority has to revoke the ASSERT although the second property may still hold.

Listing 6.9: **Security Property Specification**

```
<SecurityProperty
    PropertyAbstractCategory="http://www.assert4soa.eu/ontology/security/security#Confidentiality"
    PropertyContext="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InTransit" Version="V1">
    <NameID>Confidentialy of Data in Transit</NameID>
    <Assets>
        <Asset AssetRef="A_REQUEST"/>
    </Assets>
    <SecurityObjectives>
        <SecurityObjective ID="SO_1" Type="TOE">
            <Name>SECURE_CHANNEL</Name>
            <SecurityProblemDefinition SecurityProblemDefinitionRef="SPD_DISCLOSURE"/>
        </SecurityObjective>
    </SecurityObjectives>
    <SecurityMechanisms>
        <SecurityMechanism ID="SM_1" Type="http://www.assert4soa.eu/ontology/usdl-sec#CertificateExchange">
            <SecurityObjective SecurityObjectiveRef="SO_1"/>
            <SecurityFunctionalCriteria Scheme="CC_V3">CONF_1.1.1</SecurityFunctionalCriteria>
        </SecurityMechanism>
    </SecurityMechanisms>
</SecurityProperty>
```

Listing 6.9 shows the *SecurityProperty* element structure consisting of an abstract security property realized in a context and on a set of assets.

The *SecurityProperty* contains a *NameID* that defines a name identifier of the described property. The *NameID* allows reference to external ontologies to describe the certified security property. The *PropertyAbstractCategory* defines the abstract category of the security property.

The *PropertyContext* element defines a context in which the abstract security property is realized. The *Assets* defines a set of *Asset* elements on which the security property applies. Each *Asset* element is a reference to an *Asset* definition in the *TargetOfCertification* section.

The *SecurityObjectives* defines a set of *SecurityObjective* elements of the security property. The main elements of the *SecurityObjective* are: *a)* an identifier of the described security objective; *b)* a set of *SecurityProblemDefinitionRef* each referring to a *ProblemDefinition* defined in the *SecurityProblemDefinition* section; *c) Name* that contains the name of the security objective; *d) Description* which describes the security objective. It is an implicit assumption that all *SecurityObjectives* together contribute to the realization of the *SecurityProperty*. A *SecurityObjective* can refer to one or more *Prob-*

lemDefinitions.

The *SecurityMechanisms* defines a set of *SecurityMechanism* elements. Each element consists of an *ID* that identifies the security mechanism, the *Type* of the security mechanism (or the family of the security mechanism), a set of *SecurityObjectiveRef* elements each referring to a security objective that the security mechanism corresponds to. A *SecurityMechanism* can refer to one or more *SecurityObjective*s.

Evaluation Specific Details in an ASSERT

The two different categories of evaluation are referred as *ASSERT-E* - for test based evaluation, and *ASSERT-M* - for formal analysis. We identified three abstract elements that are common to the different types of evaluation: TypeSpecific-Property specification, ServiceModel specification, and Results of evaluation. These elements facilitate advanced reasoning to be performed on the certificates, by comparing and contrasting services based on the evaluation details such as the cardinality of the test suites, the number of tests executed and so on [9]. These elements depend on the processes and the results of each evaluation type and require different syntactic structures.

However, we consider such details to be outside the scope of this paper as it would involve discussing the current evaluation methodologies and practices. The ASSERT language, at this point, supports a choice between the three evaluation types, thus restricting an ASSERT to have one type of evidence. This is needed as the evaluation processes and the results from the three different categories are heterogeneous in nature and having multiple types of evidences in a single ASSERT would complicate the processing of the ASSERT especially in certificate comparison.

6.5 Vocabulary Integration

In the conceptual model we have presented the elements that need to be captured in a digital security certificate and we have presented a language through which we realize this conceptual model in a machine processable manner using SAML Assertions. However, the `ASSERT` language provides a data structure to represent the certificates but it does not provide nor prescribe the data that should be contained in the data structures. This is an intentional choice in order to have a clear separation between the conceptual model, the realization and the actual content of the certificates, that for instance would ease the adoption of $\mathcal{CRT}$ with different certification schemes and evaluations.

Therefore, the ASSERT language elements *should* make use of vocabularies, that must be defined by different certification authorities for their respective schemes based on the certification/ evaluation processes and the types of products that are certified.

Vocabularies can make of the existing security ontologies to describe different elements in the ASSERT language, thus permitting reasoning on them, also taking benefit from the *Linked Data* [29] paradigm, with respect to establish a link to other ontologies. An example of this flexibility is represented by the use of an ontology, called USDL-SEC C, for expressing the security mechanisms in the ASSERT language, while specific vocabularies are also foreseen for the expression of other ASSERT elements, like for security properties.

In this section, we present the vocabularies for a few identified language elements for illustration purposes:

TargetOfCertification Type

The Target of Certification Type identifies the service model being certified. The vocabulary allows the following models to be specified. The *Service Component* specifies that a component within a service is being certified, while the others identify the different service models (such as Software as a Service, Platform as a Service, Infrastructure as a Service). This allows service providers to search for specific service model types.

Listing 6.10: **TargetOfCertification.Type**

```
http://assert4soa.eu/ontology/language#ServiceComponent
http://assert4soa.eu/ontology/language#Software-as-a-service
http://assert4soa.eu/ontology/language#Platform-as-a-service
http://assert4soa.eu/ontology/language#Infrastructure-as-a-service
```

Asset Type

The assets are one of the key elements in the Digital Security Certificates. Clear identification of the asset types is necessary in order to provide precise information to consumers on what data is being secured. In the vocabulary shown below, we specify the asset types from a service consumer's perspective. We identify the asset types as *input parameters, output parameters* of the service. In addition, we define parameters that are internal to the service, for example, a cryptographic keys of a service provider used to encrypt user's data are internal to the service but need to be secured.

The last asset type are the parameters that are used by the service which come from external third party services. The $\mathcal{CRT}$ model nor its realization (ASSERT) impose any constraints on the asset type, but this is the role of the certification authorities to define the asset types based on their certification schemes and evaluation methodologies used.

Listing 6.11: **Asset.Type**

```
http://assert4soa.eu/ontology/language#InputParameter
http://assert4soa.eu/ontology/language#OutputParameter
http://assert4soa.eu/ontology/language#InternalParameter
http://assert4soa.eu/ontology/language#ExternalParameter
```

SecurityProperty.AbstractSecurityProperty

In our model, we follow a data-centric approach and hence we identify the following abstract security properties. These attributes extend the *CIA* triad, with properties related to privacy, authenticity, robustness and utility.

Listing 6.12: **AbstractSecurityProperty**

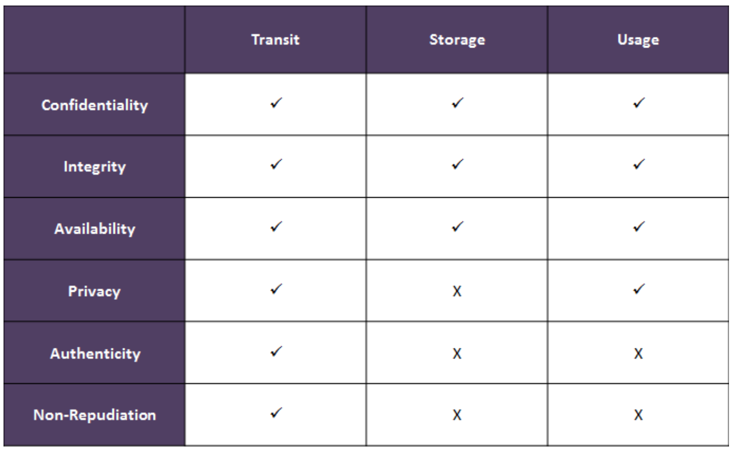
```
http://assert4soa.eu/ontology/security#Confidentiality
http://assert4soa.eu/ontology/security#Authenticity
http://assert4soa.eu/ontology/security#Privacy
http://assert4soa.eu/ontology/security#Integrity
http://assert4soa.eu/ontology/security#Non-Repudiation
http://assert4soa.eu/ontology/security#Availability
http://assert4soa.eu/ontology/security#Utility
```

Security Property Contexts

We identify the following contexts to the abstract security properties. Typically, the property contexts depend on the asset types and the abstract security properties chosen. For example, for the CIA triad, which are data centric, we identity the following contexts for the abstract security properties.

Listing 6.13: **SecurityProperty.PropertyContexts.PropertyContext**

```
http://www.assert4soa.eu/ontology/language#Transit
http://www.assert4soa.eu/ontology/language#Storage
http://www.assert4soa.eu/ontology/language#Usage
```



	Transit	Storage	Usage
Confidentiality	✓	✓	✓
Integrity	✓	✓	✓
Availability	✓	✓	✓
Privacy	✓	X	✓
Authenticity	✓	X	X
Non-Repudiation	✓	X	X

Figure 6.6: Mapping of contexts to Abstract Security Properties

Transit refers to the state where the assets are communicated between two different nodes on a network. The network can be either on the internet or the internal network of a platform provider. Security of assets in this state must be protected by the service providers.

Storage refers to the state where the assets are stored either on the file storage, database or any persistent or temporary storage location. Examples include storing the asset in a log file or a copy of the asset on a different location for redundancy etc. However, this does not refer to asset stored temporarily in cache, memory and the like.

Usage refers to the state, where the asset is being processed or used. This involves having the asset stored in memory, or cache and so on.

In Figure 6.6, we show the mapping of some of the abstract security properties with the contexts shown in Listing 6.13.

SecurityProperty.SecurityMechanisms.SecurityMechanism.Type

We use a closely related ontology vocabulary for expressing types of security mechanisms, called USDL-SEC specification (presented in Appendix C). Some of the USDL-SEC types of security mechanisms are presented here, though this is not an exhaustive list.

Listing 6.14: **SecurityProperty.SecurityMechanisms.SecurityMechanism.Type**

```
http://assert4soa.eu/ontology/usdl-sec#AccessControl
http://assert4soa.eu/ontology/usdl-sec#Certificate
http://assert4soa.eu/ontology/usdl-sec#CertificateExchange
http://assert4soa.eu/ontology/usdl-sec#Challenge-Response
http://assert4soa.eu/ontology/usdl-sec#Checksum
```

```
http://assert4soa.eu/ontology/usdl-sec#Digest
http://assert4soa.eu/ontology/usdl-sec#Signature
```

6.6 Limitations of Digital Security Certificates

Since the $\mathcal{CRT}$ language is designed in a way that it is security certification scheme agnostic, there are major issues that can arise out of that design choice:

- *Facilitate comparison among security certificates.* Given the flexibility and richness of certificate languages and ability to express similar security assertions in different ways, a certification authority may wish to define a certificate profile (e.g., by defining various certificate structure and content mandatory) to enforce uniformity of content of certificates when issued by accredited entities.

- *Facilitate production of security certificates compliant to specific certification criteria.* Given that a certificate language can support various certification schemes, a certification authority has to define its certification criteria in a certificate profile, so that all issued security certificates will conform to the criteria defined by the certificate profile.

- *Enable consumers to specify their security requirements for the services.* Similarly to CC-PP [8], the consumers or consumer groups may wish to define a certificate profile with domain-specific security requirements (criteria). When services conform to such certificate profiles, it eases the decision making process for the consumers as the conformance to a profile implies that their requirements are met by the service.

6.7 Digital Security Certificate Profiles

A Digital Security Certificate Profile, $\mathcal{CRTP}$, is a mechanism to specify the contents and semantics of a class of $\mathcal{CRT}$s. The main goal of a $\mathcal{CRTP}$ is to provide suitable means for creation of certificates by ensuring semantic uniformity of certificates for a specific (domain of) certification capturing any certification and evaluation specific aspects, vocabulary of products certified, security properties, or other aspects relevant to the semantics of $\mathcal{CRT}$s.

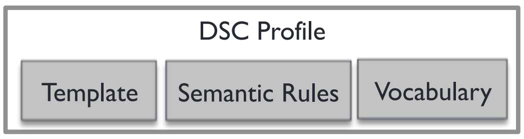


Figure 6.7: Digital Security Certificate Profile Structure

6.7.1 Conceptual Model of Digital Security Certificate Profile

In the following, we will introduce the profile structure [116]. A $\mathcal{CRTP}$ is composed of three parts:

(i) *Certificate Template*: specification of the common structure and the values of specific fields mandatory for a given certificate class

(ii) *Semantic Rules*: specification of the semantics of the certificate class in the form of semantic rules

(iii) *Vocabulary*: specification of vocabulary terms (ideally ontology-referenced terms) providing restrictions on use of vocabulary for language artefacts of security certificates of the given certificate class.

Figure 6.7 shows the abstract structure of the $\mathcal{CRTP}$. The three profile components provide certificates content uniformity in three different dimensions:

(i) certificate template ensures structural uniformity

(ii) semantic rules ensure integrity of intended semantics of certification

(iii) certificate vocabulary ensures common ontology-based ground of terms and ranges of possible values of certification (in a given domain).

Each of these different profile components are explained in detail in the following sections.

Certificate Template

The certificate template is a partially filled certificate that establishes the common structure and content of all certificates created based on a certificate profile. Therefore, any certificate conforming to a $\mathcal{CRTP}$ must include the fields, structure and values defined in the template of the profile. A certificate template specifies an incomplete realization of a $\mathcal{CRT}$

structure with respect to a given certificate syntax (e.g., XML schema). It is used as baseline for creating new certificates.

Alternatively, a certificate template can be considered as a set of implicit (semantic) rules. These rules are simple and easy to understand. For this reason, it is not required to represent a template as a set of rules, but used as a certificate template – a more intuitive notion for expressing predefined structure and values of profile elements.

Semantic Rules

The Semantic Rules define semantic constraints and dependencies between content of certificate artefacts within a given class of $\mathcal{CRT}$s. While the implicit rules defined by the certificate template are enough for structure-wise restrictions (requiring an optional element be mandatory, constrain specific structure or content of certificate artefacts, etc.), there are cases where more complex restrictions are needed such as to express artefact dependencies or artefact content constraints.

Semantic rules represent a solution allowing to formulate rules to ensure integrity of the intended semantics of a given certificate class, i.e., preserving specific semantics of certification artefacts. Semantic rules can be formulated in rule based languages (such as Schematron [156] or variants of OCL [131]) or imperative languages (such as Java or JavaScript) in function of the underlying certificate language and supported implementation. The choice of a language for expressing semantic rules has an important implication to achieve machine processability and reasoning of the rules. The language should allow rich fine-grained expression of *patterns* over certificates' content and structure.

Certificate Vocabulary

The certificate vocabulary part of the profile provides a means to define and restrict use of vocabularies on different certificate artefacts. One of the goals of the vocabulary part is to enable specific per profile (i.e., per a class of certificates) integration of the underlying certificate language with different ontology terms coming from different domains of knowledge. The ontology integration will enhance the semantic robustness among all certificates conforming to a given profile, which have been diminished by flexibility and openness of security certificate languages (models).

Ontologies provide not only a suitable source of semantically defined terms but also provide means to define relations between terms, and equivalences between different terms. That gives us a powerful way to query ontologies for different aspects of certification and related semantics.

Restricting the range of values of certificate artefacts to terms defined in ontology will make all certificates, stating conformance to the given profile, processable and comparable on those artefacts as their values are ontology terms with defined semantics and relations among them.

Similarly to the certificate template and semantic rules, one can see the certificate vocabulary section of the profile as a set of implicit rules each one restricting use of vocabulary for certificate artefacts. However, by defining explicit vocabulary section we have, first a more intuitive notion for expressing vocabulary restrictions and, second enable the use of dynamic values based on queries over ontologies, which otherwise would be difficult to achieve as semantic rules.

The certificate vocabulary section enables the use of *static* or *dynamic* vocabularies. A static vocabulary defines actual terms inside a profile. It is suitable for offline processing, but could be out-dated by an ontology evolution/ update. In contrast, a dynamic vocabulary defines actual terms by means of a query over ontology, which requires Internet connection for online processing. Ontology queries will be executed at the time of use of a given profile, i.e., the actual terms (values) will be dynamically retrieved from ontology when the profile is used.

An issuer of a profile may decide to enforce or not the use of vocabularies. When a vocabulary specification is defined mandatory the referenced language artefact must have a value from the vocabulary. If a vocabulary is optional the referenced language artefact should have a value from the vocabulary.

6.7.2 Representation of Digital Security Certificate Profile

We have realized a $\mathcal{CRTP}$ structure tailored to the `ASSERT` representation, which we will refer to as `ASSERT PROFILE` henceforth. Figure 6.8 shows the `ASSERT PROFILE` structure corresponding to the defined XML schema. For the sake of presentation, we show the profile structure as snippets abstracting away some irrelevant XML schema details to better focus on the actual data elements. We refer to the Appendix B for more details on the actual profile schema.

We will go through each of the main elements. The Issuer field identifies the entity issued (created) the ASSERT profile. The certificate template is called *ASSERTTemplate*. An *ASSERTTemplate* contains one element of type ASSERT certificate. Thus, an *ASSERTTemplate* contains an incomplete XML instance of an `ASSERT` (according to the ASSERT language schema). The semantic rules are implemented in Schematron [156]. The semantic

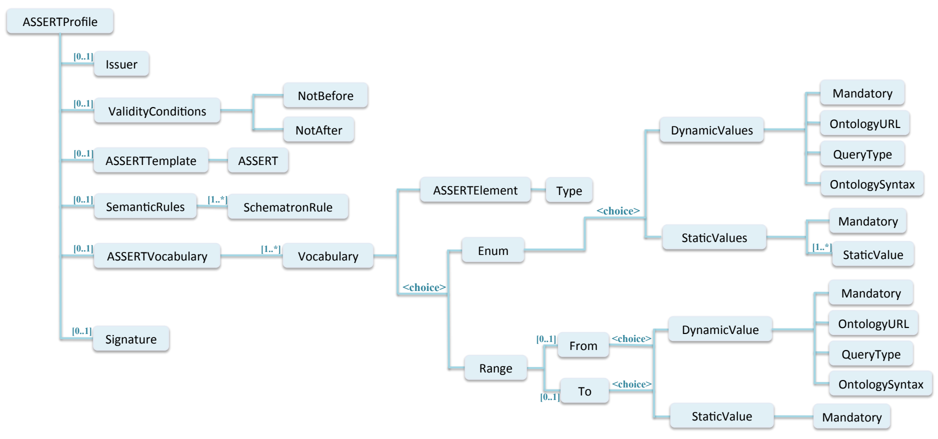


Figure 6.8: ASSERT PROFILE Structure

rules section contains a set of *SchematronRule* elements. Schematron is an ISO standard of rule-based validation language expressed in XML. Using Schematron, it is possible to make assertions about the presence or absence of patterns in XML trees.

The certificate vocabulary is called *ASSERTVocabulary*, which contains a set of *Vocabulary* elements, each one defining a specific vocabulary per an artefact (or set of artefacts) of an ASSERT. An *ASSERTElement*, part of the *Vocabulary*, identifies the ASSERT field(s) where specific vocabulary will be applied. Currently, we support the use of XPath [179] as a query language to identify nodes of ASSERTS where the vocabulary is to be applied. There is a choice of *Enumeration* or *Range* type of a *Vocabulary*. The former defines an explicit set of values, while the latter instead defines a range of values in the form of *From* and *To* boundaries, such as integer range, double range (e.g. , percentage), date range, etc. Each of the Enumeration and Range types are further defined as a choice of *DynamicValues* or *StaticValues* with an attribute field Mandatory indicating mandatory or optional use of the vocabulary data.

The *DynamicValues* artefact defines an OntologyURI of how to retrieve the ontology. The *OntologySyntax* specifies the ontology syntax. The *Query-Type* identifies the query language used to encode the query, and the actual query value. We currently support the use of SPARQL [165] as an RDF query language to retrieve information and manipulate data stored in RDF format. The *StaticValues* artefact defines a set of vocabulary terms as a simple list of values, or in case of a *Range* type a single vocabulary term.

6.7.3 ASSERT Profile Usage

Listing 6.15: **Assert Profile Example**

```
<ASSERTProfile>
    <ASSERTTemplate>
        <ASSERT>
            <ASSERTCore>
                <ASSERTIssuer>
                    O=Amazing Trust,OU=ASSERT_ISSUER,C=FR
                </ASSERTIssuer>
                <TargetOfCertification Type="http://example.org/cc/ontology/cc-language#Software-as-a-service"/>
            </ASSERTCore>
            <ASSERTTypeSpecific>
                <ASSERT-E/>
            </ASSERTTypeSpecific>
        </ASSERT>
    </ASSERTTemplate>
    <SemanticRules>
        <sch:schema queryBinding="xslt" xmlns:sch="http://purl.oclc.org/dsdl/schematron">
            <sch:pattern>
                <sch:rule context="ASSERT/ASSERTTypeSpecific/ASSERT-E/Property/PropertyName">
                    <sch:assert test="//ASSERT/ASSERTCore/SecurityProperty[
                        @PropertyAbstractCategory=current()]">
                        [Property integrity check] ASSERT.ASSERTCore.SecurityProperty.
                        PropertyAbstractCategory has to match the same value of ASSERT.
                        ASSERTTypeSpecific.ASSERT-E.Property.PropertyName
                    </sch:assert>
                </sch:rule>
            </sch:pattern>
        </sch:schema>
    </SemanticRules>
    <ASSERTVocabulary>
        <Vocabulary>
            <ASSERTElement Type="XPath">
                //ASSERTCore/SecurityProperty/@PropertyAbstractCategory
            </ASSERTElement>
            <Enum Mandatory="true">
                <StaticValues>
                    <StaticValue>http://example.org/cc/ontology/security#Confidentiality</StaticValue>
                    <StaticValue>http://example.org/cc/ontology/security#Integrity</StaticValue>
                    <StaticValue>http:/example.org/cc/ontology/security#Availability</StaticValue>
                </StaticValues>
            </Enum>
        </Vocabulary>
        <Vocabulary>
            <ASSERTElement Type="XPath">
                //ASSERTCore/SecurityProperty/@PropertyContext
            </ASSERTElement>
            <Enum Mandatory="false">
                <StaticValues>
                    <StaticValue>http://example.org/cc/ontology/cc-language#Storage</StaticValue>
                    <StaticValue>http://example.org/cc/ontology/cc-language#Transit</StaticValue>
                    <StaticValue>http://example.org/cc/ontology/cc-language#Usage</StaticValue>
                </StaticValues>
            </Enum>
        </Vocabulary>
    </ASSERTVocabulary>
</ASSERTProfile>
```

We present an example of the usage of the ASSERT PROFILE by a certification authority. We show the ability to express realistic constraints that must be enforced during the certification process.

Let us assume that there a certification authority, *AmazingTrust*, is authorized by national regulators to issue CC certifications. *AmazingTrust* is authorized to certify services that are offered as SaaS offerings and can issue certifications up to EAL4. Considering that *AmazingTrust* wants to issue digital security certificates (ASSERTS), they might want to ensure that all the ASSERTS produced must meet certain common guidelines such as:

1. Should only certify services that are offered as Software as a Service.
2. All ASSERTS produced must use the vocabulary that is based on an authorized ontology from the CC body contained as static values in the template
3. All ASSERTS produced at EAL4, must be produced by test-based certification process.
4. All ASSERTS issued must have the issuer field filled correctly

These constraints can be captured in an ASSERT profile as shown in Listing 6.15. The profile defines following constraints on ASSERTS issued by *AmazingTrust*. The *ASSERTTemplate* defines that all ASSERTS conforming to this profile must:

- be for software-as-a-service (SaaS) model services, i.e., all ASSERTS must have *TargetOfCertification* element with an attribute *Type* qualified as "http://example.org/cc/ontology/cc-language#Software-as-a-service"
- be issued by the *AmazingTrust* as issuer, i.e., all ASSERTS must have an *ASSERTIssuer* element with the defined value structure (conforming to X.509 subject structure) "O=AmazingTrust, OU=ASSERT_ISSUER, C=FR"
- be produced by a test-based certification process, i.e. must contain *ASSERT-E* type-specific structure, but without defining any particular content for the structure. This means that ASSERTS stating conformance to the profile can contain any specific *ASSERT-E* content

The *SemanticRules* section defines one Schematron rule, which forces the security property abstract category value as defined in the *SecurityProperty* element in the *ASSERTCore* match the value of the *PropertyName* of *Property* definition of *ASSERT-E*. Such an integrity constraint is difficult to enforce without a semantic rule.

The *ASSERTVocabulary* defines two vocabularies – one for the *PropertyAbstractCategory* attribute of the *SecurityProperty* element, and another one for the *PropertyContext* attribute of the same *SecurityProperty* element based on an ontology that is approved by CC bodies. The first vocabulary defines static values of the CIA triad – Confidentiality, Integrity and Availability – as terms from an ontology specific definition, and marks those as mandatory. The second vocabulary defines optional values for the artefact *PropertyContext*, such as *Storage, Transit and Usage*, as terms from an ontology-specific definition.

Conclusion

In this chapter, we presented the concept of Digital Security Certificate $\mathcal{CRT}$, which is designed to be a certification scheme independent, machine processable, descriptive security certificate. We have presented an XML based language (`ASSERT` language) to represent the contents of the $\mathcal{CRT}$ and explained the integration of external vocabularies into the `ASSERT`. In addition, we presented the concept of a Digital Security Certificate Profile, that helps in the generation of uniform $\mathcal{CRT}$s. We believe that a wider adoption of these concepts will provide security assurance in business-critical domains such as financial, defence and healthcare [72].

Publications

The contents of this chapter have been discussed in the following publications:

⋆ **S. P. Kaluvuri**, H. Koshutanski, F. D. Cerbo, and A. Maña. *Security assurance of services through digital security certificates*. In Web Services (ICWS), 2013 IEEE 20th International Conference on, pages 539–546. IEEE, 2013.

⋆ **S. P. Kaluvuri**, H. Koshutanski, F. Di Cerbo, R. Menicocci, and A. Maña.*A digital security certificate framework for services*. International Journal of Services Computing, 1(1), 2013.

⋆ V. Lotz, F. Di Cerbo, M. Bezzi, **S. P. Kaluvuri**, A. Sabetta, and S. Trabelsi. *Security certification for service-based business ecosystems*. The Computer Journal, page 101, 2013

Technical Reports

A more comprehensive discussion about the topics mentioned in this chapter are provided in the following technical reports:

⋆ H. Koshutanski, A. Maña, R. Harjani, M. Montenegro, **S.P. Kaluvuri**, F. Di Cerbo, E. Damiani, C. Ardagna, M. Anisetti, D. Presenza, S. Gürgens, R. Menicocci, V. Bagini, F. Guida and A. Riccardi. *ASSERT Language V1.2*. ASSERT4SOA Project Deliverable(D1.2), 2012.

⋆ A. Maña, H. Koshutanski,J. González, M. Montenegro, R. Menicocci, A. Riccardi, V. Bagini, F. Di Cerbo and **S.P. Kaluvuri**. *ASSERT Profiles*. ASSERT4SOA Project Deliverable (D1.3), 2012.

⋆ H. Koshutanski, **S.P. Kaluvuri**, A. Maña, M. Montenegro, M. Anisetti, C. Ardagna, E. Damiani, S. Gürgens, D. Presenza. *ASSERT Language V1.4*. ASSERT4SOA Project Deliverable(D1.4), 2014.

Chapter 7

Dynamic Security Certification Lifecycle

"Fear cannot be banished, but it can be calm and without panic; it can be mitigated by reason and evaluation."

— Vannevar Bush

The representation of a security certificate in a machine readable form is one of the many changes necessary to bring security certifications to service environments. In Section 5.3 we have discussed the limitations of the current certification schemes in service environments. While the usability issues are addressed using the Digital Security Certificate concept, the assurance gained from the certification schemes is still rather weak, given that currently security certification adopts a "point-in-time" approach. In such approaches, a version of a service can undergo the evaluation process and consequently be certified. However, consumers cannot gain assurance that during consumption, which can occur at any point after the certification of the service, that the service instance is the same version that is certified and the platform, on which the service operates, has not changed from the certified configurations.

In fact, in service environments, service providers update their service offerings frequently, almost on a daily/ weekly basis, to add new features to the service, or deploy optimized service implementations or fix bugs or vulnerabilities. The fact that the majority of the cloud service development happens using agile development methodologies contributes to this frequent update cycle and such dynamic environments is one of the fundamental advantages that developers gain from service environments. Since any change to the service happens in a transparent manner from a consumer's perspective, developers can exploit this advantage to reduce the *time to market* by providing the basic functionalities and adding more features overtime.

But current security certification schemes are rather rigid when it comes to changes to certified services. They consider any change to the certified service or its underlying operational environment (OE), which comprises both the IT aspects (IT-OE) and non-IT aspects (non-IT-OE) ,that is not part of the certified configurations to invalidate the certification gained. This poses a serious barrier to make it attractive for service providers to get their services certified.

In this chapter, we present an approach for the Digital Security Certificate Maintenance that can cope with frequent service updates and can monitor the IT- OE for any change. However, in service security certification, there are many different participating entities and the various trust relationships that exist between these entities have an implication on the security assurance that is gained from the certification and in particular have a direct impact on the security assurance over the IT (and non-IT) operational environment. Hence, we first present a few of the key scenarios of trust relationships that exist between the different entities in service environments.

7.1 Trust Relations in Service Environments

In this section, we consider the scenario of a service provider deploying its certified services on a platform offered as a service (PaaS) by another provider. We do not consider the case where the service provider and the platform provider are the same entity.

Typically, in cloud-based architectures, trust relations between the entities can be classified either as based on verifiable facts (explicitly described), or as (mostly implicit) assumptions on the trusted entity.

7.1.1 Distributed Trust Scenario

One of the approaches to consider the trust assumptions in traditional security certification schemes is of a consumer who trusts the certification authority to follow proper processes to certify a software and also trusts the software provider for providing the exact version of the software that has been certified. Analogously to service certification, the consumer can trust the Certification Authority to assess the service following proper processes and also trust the platform provider to be not malicious. This scenario is depicted in Figure 7.1. This trust relation can include the assumption that

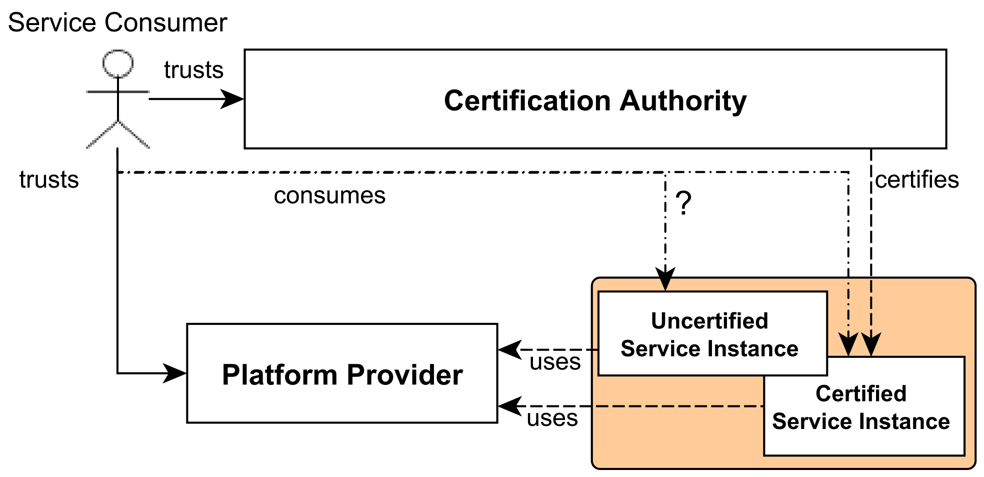


Figure 7.1: Distributed Trust Scenario

the platform provider treats the information sent to the service as confidential and that the platform does not snoop on the information exchange between the service and the consumer. The consumer can also trust the platform for service integrity, i.e., that the platform does not tamper with the service. Trust assumptions could be adapted based on the reputation of platform providers (as is the case for major platform providers such as Amazon Web Services, Google marketplace, Windows Azure, etc.).

However in this scenario, consumer's trust relations with the CA and with the platform are independent of each other: the trust of the consumer on the certification authority for assessing a service properly and the trust of the consumer on the platform provider do not depend on each other. Because of this independence, there is a clear threat that the most critical entity – the service – could mislead the consumer by getting a certain implementation of the service certified while serving another version to the consumer, masquerading the latter version as the certified version of the service. Of course, this scenario is based on the assumption that the service provider is not trusted, as there is no use for certification if the service provider is already trusted by the consumer.

7.1.2 Distributed Trust & Delegated Monitoring

Another approach is shown in Figure 7.2 where trust is still distributed, but involving a mesh of trust relations:

- The consumer trusts the CA for following the right procedures to cer-

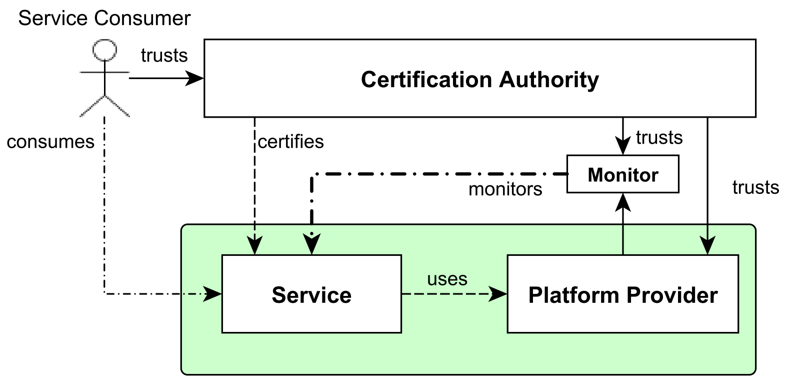


Figure 7.2: Distributed & Delegated Trust Scenario

tify the service as well as for monitoring the service operation all along its lifecycle

- The CA trusts the platform provider for not being malicious and for not tampering with the service that is deployed and certified; additionally the CA monitors the service through the technical means that the platform provides to the CA.

- The consumer *gains* trust on the platform provider for not being malicious because the CA trusts the platform provider

This approach has the advantage that the robustness of the certificate binding can be ensured at every instance of the service invocation, thereby eliminating the need to trust the service provider. In addition, the CA can verify the operational environment's integrity through the monitor. However, there is still a strong assumption that both the consumer and the CA trust the platform provider for not being malicious.

7.1.3 Centralized Trust Scenario

Finally, a third approach to gain security assurance over the OE is through the certification of the entire service stack (Service along with the OE) by distributing the trust between the Certification Authority (CA), the Platform Provider (PP) and the Service Consumer (SC) as shown in Figure 7.3. In such scenario, the consumer trusts *only* the certification authority for following the proper processes for certifying a service and to monitor the service – and its execution environment, which is controlled by the platform provider – throughout the service lifecycle.

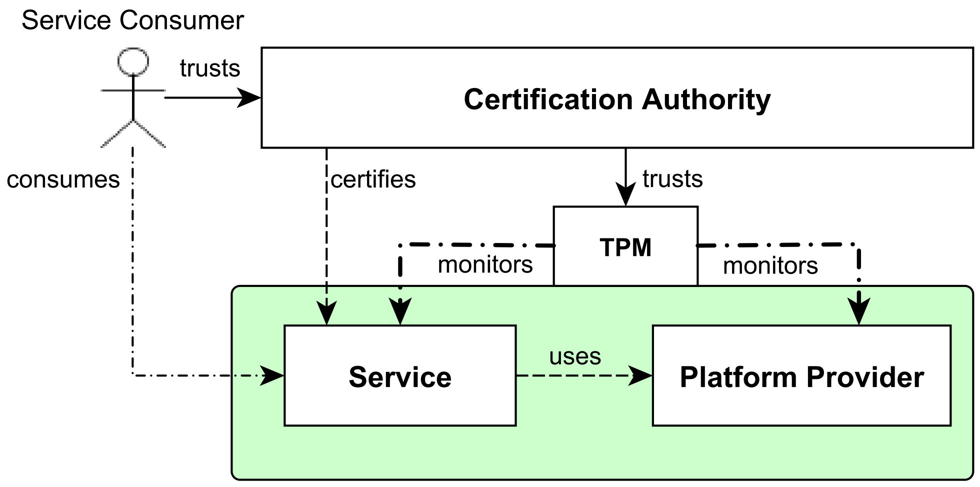


Figure 7.3: Centralized Trust Scenario

Certification Authorities can monitor the platform through purely technical means such as based on audit logs that are protected using Trusted Platform Module (TPM) [138]. In the case of service certification, the TPM should be used to protect a secure audit trail used by the platform to record any security relevant action performed either by the the service provider (e.g., deployment of another version of the service) or by the platform provider (e.g., change of some security relevant configurations on the software stack). The TPM allows the CA to check if the service implementation has not changed and the underlying architecture is still the same as the one considered at certification time. In other words, if a snapshot of the service implementation and the underlying software stack is certified, the TPM allows the CA to ensure that the service instance that is offered to consumers conforms to the version that was certified. However, this approach has two major drawbacks:

1. It requires the platform providers to add a TPM on every node in their cloud infrastructures, which is not always feasible. In addition, the technical overhead to maintain such a large network of TPMs and the associated costs hinder organizations such an approach.

2. It requires to ensure that the platform provider always registers security relevant activities on the audit trail that is protected by the TPM. In other words, the TPM can ensure the integrity of the audit trail, however, it cannot force the platform provider to log the relevant in-

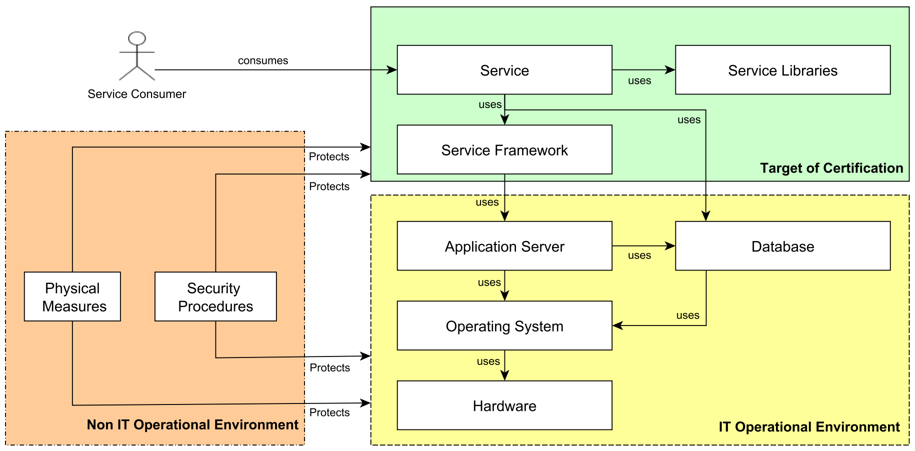


Figure 7.4: Simplied illustration of a Service and its Operational Environment

> formation in the audit trail, which is necessary in a scenario where the platform provider may be malicious. Hence, this approach still has an undercurrent, implicit assumption that the platform provider is not malicious.

In addition, there are several other technical mechanisms that can be used by certification authorities to monitor the platform such as *verifiable proofs* [160].

The technical mechanisms based on TPMs or other such approaches help to reduce the implicit trust assumptions between entities and establish trust based on verifiable facts.

7.2 Digital Security Certificate Lifecycle

In this section, we discuss an approach, based on the trust scenario discussed in Section 7.1.2, that can address the issues of re-certification of service implementations to handle the frequent update lifecyle as well as to provide assurance over the IT Operational Environment of the service.

A simplified scenario of a service and its operational environment is shown in Figure 7.4. In the figure, Service represents the deployed instance of the service by the providers. The service can make use of external service libraries which will be part of the *Target of Certification* (TOC) but might be out of the *Target of Evaluation* (TOE), that is, the external service libraries are not evaluated during the certification process. Service implementations typically make use of a *Service Framework* such as Apache CXF, Apache Axis and so on, to expose their software as a web service. The service provider has control over these three components as these are dependent on implementation choices and the nature of the service developed.

The service along with the external libraries and the service framework is deployed on an Application Server such as Apache Tomcat [11], IBM WebSphere [86], SAP Netweaver [158] and so on. In addition, the service can connect to the database to perform any operation that requires data persistence. The application server and database instances are executed on an operating system which runs on hardware IT equipment owned by the platform provider. All these components are typically under the control of the platform provider. In addition, the physical security measures (notably perimeter security) and security procedures (notably access control restrictions) are enforced by the platform providers.

Typically, platform providers maintain *server farms*, which consist of thousands of servers, to provide multi-tenancy to their platform consumers, who are typically the service providers that deploy their services on the platform. Figure 7.4 is a rather simplified scenario where services are running atop of a single server. However, it illustrates the different dependencies that exist between the various system and service components.

Platform providers maintain a Cloud Configuration Management system to better manage the different nodes (servers) on the server farms and to allocate the resources for a service instantiation based on the capabilities that the service provider has chosen. These capabilities can be either on the computing (such as the node should have 1 GB of RAM) or on technology (the node should be running on a Debian-based Operating system) or on specific configurations (the node should be using version 7.2 of Apache Tomcat).

Platform providers update the different software products (OS, Application Servers, Databases etc) to patch against a new vulnerability or fix bugs, which is essential for the secure operation of the platform. On the other hand, platform providers may upgrade their existing software products to newer versions to benefit from new features or better performance. Upgrades to products can have implications on the operation of the components that use them (Figure 7.4).

Security certificates of services must cope with these changes to the operational environment and determine which changes *might* affect the certified service and inform the certification authorities. In this context, the maintenance of security certificates becomes critical to the success of the overall certification scheme.

7.2.1 Digital Security Certificate Maintenance Framework

A high-level overview of the Digital Security Certificate Maintenance (DSCM) Framework is shown in Figure 7.5. The DSCM framework exposes several interfaces that can be accessed by various entities. Any operations on the certificates is performed in a component called *Certificate Registry Abstraction Layer* which in turn interacts with the Certificate Repository to store and retrieve certificates.

Similar to a PKI [119] or to the Common Criteria scheme [48]), the DSCM Framework allows certification authorities to register certificates that are issued or to revoke any certification in case of violations. The DSCM framework exposes the following interfaces:

- *Register Security Certificate:* This interface is used by the accredited certification authorities to register the security certificates of a service that has passed the evaluation.

- *Revoke Security Certificate:* This interface is used by the accredited certification authorities to revoke the security certificate of a service.

- *Fetch Security Certificate:* This interface is used by service consumers and by the certification monitors (Section 7.2) to find the security certificate associated to a service.

- *Verify Security Certificate:* This interface is used by the service consumers to validate the structural and semantic correctness of the certificate, and also to verify whether the current service instance violates the certified security properties.

- *Flag Security Certificate:* This is used by the *Certificate Monitors*(Section 7.2) to inform the DSCM framework that a service implementation *might* violate the certified security properties contained in the security certificate. The DSCM framework relays this information to the certification authorities who have issued the certificate to re-evalute the service.

- *Monitor Status Check:* This is used by the *Certificate Monitors*(Section 7.2) to inform the DSCM framework regarding its own integrity to ensure that the monitor has not been compromised.

We do not explain the internal components of the DSCM Framework, rather we focus on the more challenging service security monitor. Given that it is the most critical component that enables consumers to gain assurance in service environments.

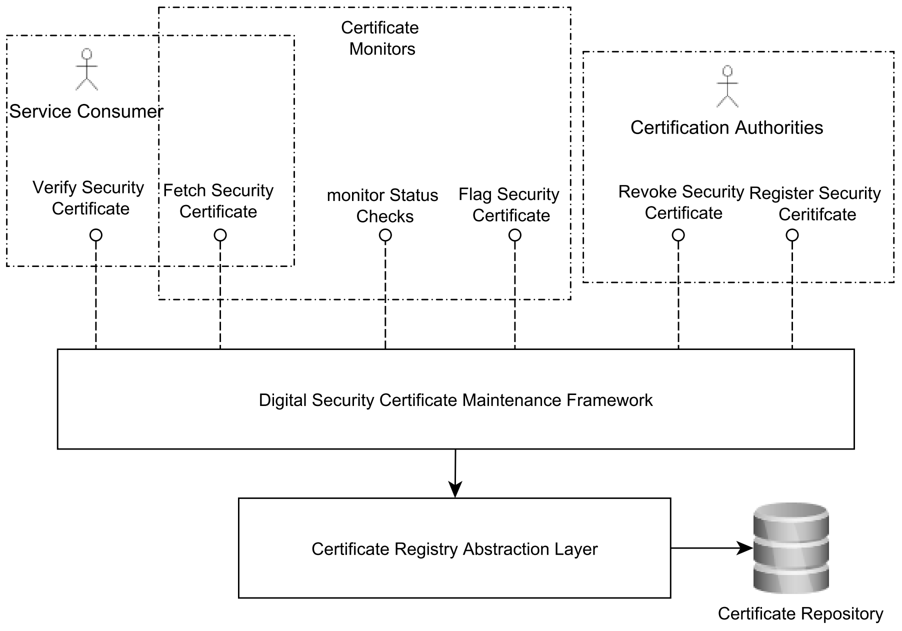

Certificate Monitors
Service Consumer
Certification Authorities
Verify Security Certificate
Fetch Security Certificate
monitor Status Checks
Flag Security Certificate
Revoke Security Certificate
Register Security Ceritifcate
Digital Security Certificate Maintenance Framework
Certificate Registry Abstraction Layer
Certificate Repository

Figure 7.5: Digital Security Certificate Maintenance Framework Overview

7.2.2 Service Certificate Monitor

We consider the distributed and delegated trust approach (Section 7.1.2), and treat the platform as a black-box while depending on the platform to provide access to its configuration management system to read information regarding the nodes on which the service could be instantiated. Moreover, we also depend on the platform to grant access to the deployed service instance in case the service has been redeployed with a different version than the certified one. In this section, we refer extensively to the conceptual model of a Digital Security Certificate ($\mathcal{CRT}$) presented in Section 6.2.

A high-level overview of the Service Certificate Monitor (SCM) and its interactions with the platform and the DSCM framework are shown in Figure 7.6. The SCM is a module running on the platform along with other applications, in a non-intrusive manner. It maintains a list of certified services ($\mathcal{S}$) that are deployed on the platform and monitors those services for any changes to its deployment.

The SCM performs periodical checks on the deployed service and the platform's configuration. The frequency of these checks can be adjusted based on the requirements of the platform provider and certification authorities. Moreover, the SCM can be explicitly invoked by a service consumer through the DSCM Framework. A detailed architecture of the (SCM)

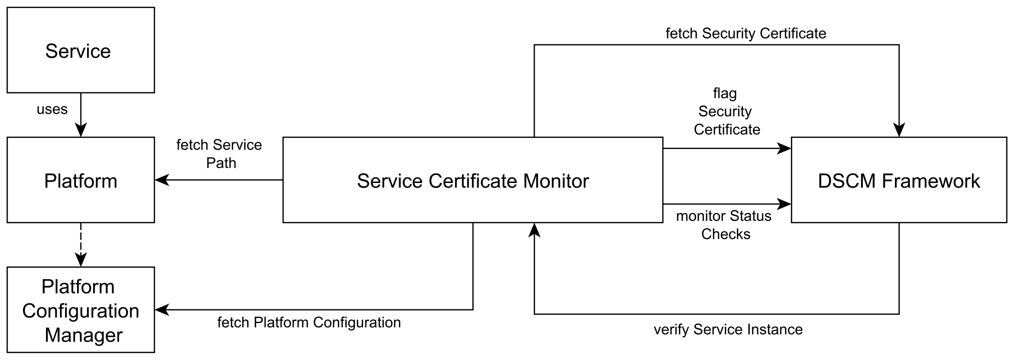

Service
uses
Platform
fetch Service Path
Service Certificate Monitor
fetch Security Certificate
flag Security Certificate
monitor Status Checks
DSCM Framework
Platform Configuration Manager
fetch Platform Configuration
verify Service Instance

Figure 7.6: Service Certificate Monitor Overview

is shown in Figure 7.7. It consists of five main modules:

- *Service Verification:* It is the master component that orchestrates the various checks within the SCM
- *Service Integrity Verifier*, that verifies the integrity of the service implementation by computing the *checksum* of the service implementation and its deployment environment.
- *Service Model Analyser*, that generates a formal model of the service from the deployed instance of the service (such as WAR) and then optimizes this model based on the *Target of Certification* contained in the certificate of the service. It extracts the model of the service from its certificate and compares it with the newly generated and optimized service model. It then analyses all the changes that have happened to the service implementation (or its deployment) that might affect the certified security properties.
- *Service Dynamic Evaluation*, that evaluates the current deployed instance of the service regarding certain properties that cannot be proven during certification time.
- *Certificate Parser:* parses the Digital Security Certificate and provides the different elements contained in the certificate to the *Service Verification Component*

When a consumer requests the DSCM framework to verify a service security certificate, the DSCM verifies the signatures, validity as well as the syntactic and semantic correctness of the certificate. It then requests the SCM module corresponding to the platform on which the service is deployed to verify the service instance to ensure that the current instance of the service still satisfies the certified structural properties. A more detailed description of the components inside the SCM module are explained in the following sections.

Service Verification (SV)

The Service Verification (SV) component is the key component within the SCM. The DSCM framework when it requests the SCM to verify the service instance, it sends a security certificate along with the request. The SV module invokes the *Certificate Parser* (CP) component to parse the security certificate ($\mathcal{CRT}$) of the service to extract the various contents of the security certificate. The SV component then sends the $\mathcal{TOC}$ to the *Service Integrity Verifier* (SIV) component. The SIV checks the $\mathcal{TOC}$ against the current service instance and the platform configuration and informs the SV regarding the integrity of the service and its platform. If both the platform and the service pass the integrity checks, the SV informs the DSCM framework that the security certificate of the service is still valid.

However, if the platform configurations are not in the certified configurations of the $\mathcal{TOC}$ the SV informs the DSCM framework to flag the security certificate for further verification. In case the deployed service instance fails the integrity check, i.e., the certified instance of the service has been modified (due to updates or upgrades), the SV component invokes the *Service Model Analyser* (SMA) component to check if the changes to the certified service can affect the certified security properties. In order to do this, the SV component fetches the $\mathcal{TOC}$ and $\mathcal{SPS}$ elements from the CP and sends it to the SMA component.

The SMA component informs the SV whether the changes to the certified service affect the security property and in case they do affect, the SV informs the DSCM framework to flag the security certificate in order to investigate the impact of the changes.

When the SV module receives a request to verify a certificate from the DSCM framework, it detects whether there are any dynamic checks to be performed by inspecting the $\mathcal{ESD}$ element of the $\mathcal{CRT}$ and if there are any dynamic checks to be performed it invokes the SDE component.

Finally, the SMA communicates with the DSCM framework through mechanisms such as *heart beats* [82] to prove its integrity. This is necessary to ensure the integrity of the monitor itself has not been compromised in case of security breaches on the platform.

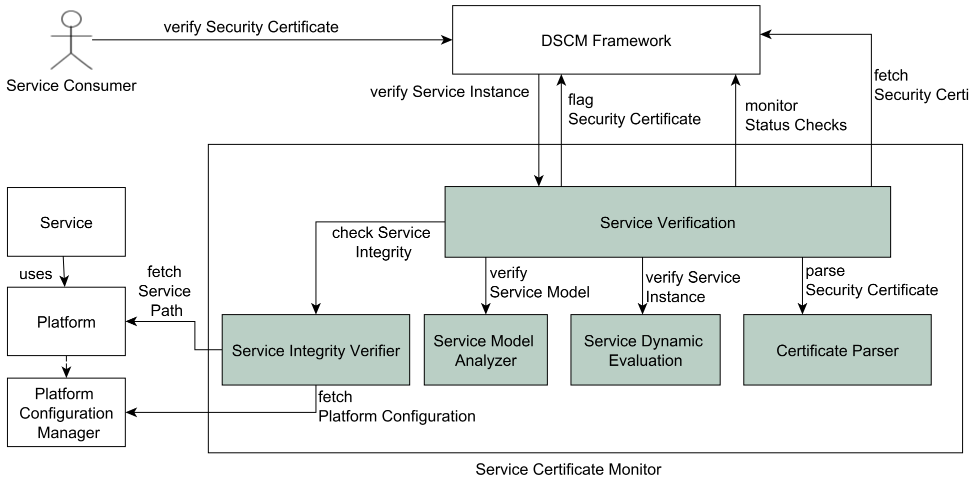

Service Consumer
verify Security Certificate
DSCM Framework
verify Service Instance
flag Security Certificate
monitor Status Checks
fetch Security Certificate
Service
uses
Platform
Platform Configuration Manager
fetch Service Path
check Service Integrity
Service Verification
verify Service Model
verify Service Instance
parse Security Certificate
Service Integrity Verifier
Service Model Analyzer
Service Dynamic Evaluation
Certificate Parser
fetch Platform Configuration
Service Certificate Monitor

Figure 7.7: Service Certificate Monitor Architecture

Service Integrity verifier (SIV)

The SIV component when it receives a request to verify the integrity of the deployed service, fetches the configuration of the platform from the *Platform Configuration Manager* and creates a Deployment Model($\mathcal{DM}'$) for the service instance. It then compares whether the configurations of the platform are part of the certified platform configurations $\mathcal{DM}$ contained in the $\mathcal{TOC}$ element of the $\mathcal{CRT}$. The algorithm for the platform integrity check is shown in Algorithm 1.

Algorithm 1 Configuration Verification

```
 1: procedure CheckConfigurations($\mathcal{DM}, \mathcal{DM}'$)
 2:     for each $sc$ in $\mathcal{DM}$ do
 3:         $c \leftarrow S_s \in sc$
 4:         flag ← false
 5:         for each $sc'$ in $\mathcal{DM}'$ do
 6:             $c' \leftarrow S'_s \in sc'$
 7:             if $c = c'$ then
 8:                 if $SC'_c \subseteq SC_c$ then
 9:                     flag ← true
10:         if flag=false then
11:             return false
12:     return true
```

If the configuration verification of the platform is successful, the SIV component checks the integrity of the service by generating a cryptographic *checksum* ($\mathcal{TH}'$) of the deployed service instance based on the algorithm that specified in the $\mathcal{CRT}$ ($\mathcal{TOC}$ to be precise). The newly computed checksum is then compared to the checksum contained in the $\mathcal{CRT}$, $\mathcal{TH}$, and if they are equal, ($\mathcal{TH}' = \mathcal{TH}$), it informs the SV that the integrity of the service and the platform are intact and the SV relays this information to the DSCM Framework.

If the configuration verification fails (returns *false*), the SIV component informs the SV that the platform integrity check has failed. It is necessary to specify explicitly at which layer (service or platform) the integrity check failed, because this impacts the further course of action taken by the SV component. When the platform integrity fails, the SV immediately raises a flag regarding the security certificate of the service to the DSCM framework.

In case the service instance has been modified, the SIV informs the SV that the service has failed the integrity check. The SV then initiates an analysis of the service to understand the implications of the changes made to the service which is performed by the *Service Model Analyser* component.

Service Model Analyser

The Service Model Analyser (SMA) component of the SCM module is responsible for extracting a formal model from the deployed instance of the service. It is assumed that the deployed instance will not contain the source code of the service and hence it should extract the service model from the compiled version of the service implementation.

The SV component passes the Target of Certification ($\mathcal{TOC}$) and Security Property Specification ($\mathcal{SPS}$) contained in the $\mathcal{CRT}$ to the SMA. In addition, it also passes the path of the deployed service instance. Based on programming language used to develop the service, the SCM module generates the static *call graph* of the service implementation.

The SMA finds the certified security properties ($\mathcal{P}$) from the $\mathcal{SPS}$ element, and for each security property (p), finds the set of assets that are secured ($\hat{\mathcal{A}}$). It creates a set of affected assets ($\hat{\mathcal{A}}'$) that is a union of all the assets secured by the various security properties in the $\mathcal{CRT}$.

$$\hat{\mathcal{A}}' = \hat{\mathcal{A}}' \cup \hat{\mathcal{A}} \quad \forall p \in \mathcal{P}, \quad where \quad p = \hat{p} \times \hat{\mathcal{A}} \tag{7.1}$$

For each asset in the affected assets set, the SMA finds the components that access these assets from the Asset-Component Identification and creates a set of affected components ($\mathcal{C}'$).

$$\forall a \in \hat{\mathcal{A}}', \quad \mathcal{C}' = \mathcal{C}' \cup \mathcal{C}, \quad where \quad \mathcal{C} \in \alpha \tag{7.2}$$

The SMA then finds the component model ($\mathcal{CM}$) for each of these affected components and compares it with the component model generated from the call graph of the deployed service instance. In order to analyse the *impact* of these changes in the deployed service, the SMA needs to compute the *delta* between the two components models, that precisely point to the changes that have happened in the deployed service instance.

The delta (Δ) for each component (C) is derived by isolating the additional interactions to the service implementation from the interactions that have been already modelled in the certified instance of the service.

$$\Delta_C = \mathcal{INV}' \in \mathcal{INT}' \mid \mathcal{INV}' \notin \mathcal{INT} \tag{7.3}$$

The SMA then proceeds to evaluate the effect of these additional interactions on the certified security properties. Since the SMA does not have access to the source code and has to generate a call graph based on the compiled version of the service implementation, it cannot infer the infer directly which abstract security property ($\hat{p}$) is affected by the changes to the service implementation. Hence, we focus on the contextual security properties of the service ($\hat{p}_c$).

Typically, static call graphs contain invocations from the top level web service APIs to calls to the native libraries provided by the specific platform. The SMA component contains a mapping of the native libraries provided by the platform to specific categories such as network access libraries, I/O libraries for any actions that relate to reading or writing data from and to the file storage on the platform and so on. Each category is then associated to a set of *Contexts*, for example, network library category is mapped to the context - *transit*, while database or I/O operations library categories are mapped to *storage* context. These categories we provided here is given for illustration, typically, these categorization must be provided by the DSCM framework to the SCM module based on the Software Development Kits (SDKs) provided by the platform to service developers. These SDKs along with the native programming language libraries are used by service developers to build their service logic.

When the Δ of a particular component (C) contains an invocation ($\mathcal{INV} \in \Delta_C$) that uses a library which is associated to a context that is among the certified security properties ($\mathcal{P}$), the SMA informs the SV that the deployed instance of the service *could* violate the certified security properties. The SV informs the DSCM framework to flag the service security certificate with the information that the deployed instance could affect the security property and the DSCM framework should take necessary action to remedy the situation such as inform the certification authorities to re-evaluate the service instance or revoke the certificate etc.

This approach will raise many false positives and that is precisely the reason the certificates are *flagged* to make all the different entities aware of the change. For example, from a service consumer point of view, they gain lower assurance when they see a service with a *flagged* certificate. From a certification authorities point of view, they can be aware that a service that has been certified by them, could potentially violate the certificate. Hence, they can intimate the service providers to get their new versions re-evaluated or they can revoke the issued certificates.

Service Dynamic Evaluation

In some cases it is not possible to evaluate a product during its certification phase, we anticipate the need to have dynamic evaluation of services and provide a feasibility in the $\mathcal{ESD}$ element of the $\mathcal{CRT}$ to specify tests that need to be carried out during a service's lifetime. The SCM module delegates the evaluation of the service to the Service Dynamic Evaluation (SDE) component. This feature of the DSCM framework is a huge departure from existing certification approaches, where products are evaluated statically at a point in time. For service consumers, this gives them rather high levels of assurance regarding the service's security.

7.2.3 Assurance over Operational Environment

The DSCM framework we proposed along with the SCM module helps in increasing quality of assurance that consumers gain from the Digital Security Certificates. Our framework improves the maintenance aspect of the security certificates lifecycle. We discussed in Section 5.3 the major obstacle of current security certification schemes to address the need of providing assurance over the operational environment of a service. Our proposal not only address this aspect, but also increases the transparency regarding the operational environment of a service during service consumption.

By introducing the notion of *flagged* certificates, we allow the assurance to *degrade* over time, if and when there are changes to the service or its operational environment that impacts the certified properties. This is very significant, as it allows consumers to perform their own risk analysis during service consumption whether they would want to consume a service with a flagged security certificate. Thus the user is empowered with up to date information regarding the service instance, rather than depend on a security certificate that has been issued sometime in the past.

The DSCM Framework is also able to cope with frequent updates to services and thus allowing service providers to get the core product certified and add more features in a manner that does not affect the certified security properties. This gives incentive to service providers to have their services undergo certification, and gives the whole certified service ecosystem a huge impetus.

Moreover, the DSCM framework and the SCM module we proposed are generic in nature and can be adapted to multiple security certification schemes as well as multiple platform providers. This once gain, allows the thesis contributions to have a wider impact on the security certification landscape.

7.3 Limitations & Directions

A key limitation of our approach is the strong trust assumptions that is needed to make the framework work - the platform provider is trusted to be non- malicious both by the Certification Authority and the Service consumer. Even though this seems realistic in many cases where the platform provider is reputed, such as in the cases of Microsoft Azure, Amazon Web Services and so on. However, these trust assumptions can be strengthened by allowing the platform providers to undergo regular audit and/ or gain ISO 27001 certification.

We proposed [95] a more closer integration between *Process based certification schemes* (such as ISO 27001) with the *Product based certification schemes* (such as CC). We proposed that a common vocabulary be used by the process certifications (to define security controls) and the product certifications (to define their Non-IT security objectives). This step would prove extremely beneficial, along with the Digital Security Certificate, and the DSCM framework to provide a more thorough and comprehensive security assurance to service consumers.

In addition, by using TPM-based monitoring approaches (Section 7.1.3) we can enhance the quality of assurance gained by consumers. TPM monitors extend the verification scope by allowing Certification Authorities to monitor the platform provider more thoroughly, rather than depending on trust assumptions.

Another key limitation of our approach is that the *Service Certificate Monitor* is mainly focused on the static interaction model of the service. Interaction models are able to detect changes that could affect the certified security properties, however, they raise many false positives. Our proposed solution can be made more precise by using data flow model and control flow models. Such approaches help in analysing the changes in a more precise manner reducing the overhead on the DSCM framework to handle requests to flag the security certificates of services.

Conclusions

In this chapter we have discussed the various trust relations that exist between different entities in service environments and how they impact the security assurance consumers gain from security certifications. Based on these trust assumptions, we propose a Digital Security Certification Maintenance Framework that allows efficient and improved maintenance of Digital Security Certificates of services. We also proposed the architecture of a Service Certificate Monitor that provides assurance regarding the Operational Environment of the service. These proposals allow consumers to gain increased security assurance from a service security certificate.

Publications

The contents of this chapter have been discussed in the following publications:

⋆ M. Bezzi, **S. P. Kaluvuri**, and A. Sabetta. *Ensuring trust in service consumption through security certification*. In Proceedings of the International Workshop on Quality Assurance for Service-Based Applications (QASBA), pages 40–43. ACM, 2011

Part III

Impact

Chapter 8

Exploitation & Impact

"Knowing is not enough; we must apply. Willing is not enough; we must do."

— Johann Wolfgang von Goethe

In this chapter, we present how the contributions of the thesis have been exploited within an industrial context. In addition, the contributions of the thesis have been used within a EU funded Project ASSERT4SOA [16], which focuses on consuming the digital security certificates and performing advanced reasoning on them to aid in the service discovery process.

We also present a research prototype that was built to generate the `ASSERTS`, and to verify their conformance against an `ASSERT` Profile. We showcase its funtionalities and present the ease with which security certificates can be generated.

Finally, we discuss the impact of the thesis contributions, in particular the Digital Security Certificate concept (and the `ASSERT` language) on the current security certification schemes. We present the results of a survey conducted on a focus group of certification experts and CC certification bodies to validate our claims regarding the usefulness of the Digital Security Certificate concept.

8.1 ASSERT Enabled Service Marketplace

SAP [150] is one of the major software providers in the world. It specializes in providing Enterprise Resource Planning (ERP), Business Analytics and High-speed Database solutions to major organizations around the world. In the past few years, SAP has been embracing the cloud paradigm to offer their software as a service [151]. SAP introduced the SAP HANA Cloud platform [152] that enables third party service providers to deploy services which offer specific business functionality. In addition, SAP runs an online SAP Store, where service providers can advertise their services and consumers can search and discover service solutions based on their requirements.

Within the context of the EU funded project ASSERT4SOA [16], we contributed to the development of a prototype of an ASSERT Enabled Service Marketplace (AESM) that makes use of the digital security certificates (ASSERTS) of services to help service consumers discover services that match their security requirements and needs.

The AESM is a service marketplace targeting the procurement of business software services. Cloud platforms provide an easy way for software vendors to offer services and service-based applications to their customers without the need to setup and maintain a costly IT infrastructure on their own. Because of the extensibility and the economic attractiveness of business models offered by the cloud paradigm, over the recent years, more and more companies are moving their software offerings to cloud-based solutions. The growing popularity of service-based and cloud applications is accompanied by the rise of the marketplace metaphor, adopted among many others by the Amazon Web Service Marketplace [5] and the Google Apps Marketplace [76]. These marketplaces are organized as open ecosystems, where each day new vendors (be they one-person businesses, SMEs, or large corporations) enter the marketplace and with whom potential customers might have never conducted business before. By their very nature, service marketplaces exacerbate even further the security and trust concerns that are typical of service-based systems. In such dynamic ecosystems, in which reputation alone cannot be reliably used to build trust, service consumption is problematic because important information about the security (or lack thereof) of services is not available. This problem might not be perceived as critical by private consumers, but it does represent a major stumbling block for corporate consumers, who must ensure that their own customers' security is guaranteed and who risk reputation and financial loss in case of security incidents. To mitigate these risks, businesses strive to be in the position of proving that they performed due diligence to ensure that the service they provide complies with the relevant security and data protection laws and regulations.

Based on these considerations, and loosely inspired by the SAP Service Marketplace (whose look and feel we replicate) [153], we designed and implemented an AESM, a proof-of-concept, that uses the results of the thesis

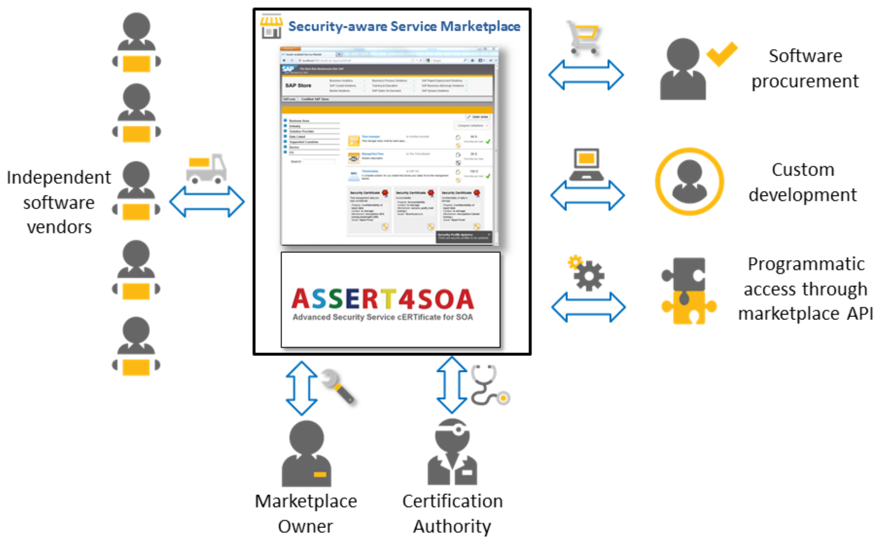

Security-aware Service Marketplace
Independent software vendors
Software procurement
Custom development
Programmatic access through marketplace API
ASSERT4SOA
Advanced Security Service cERTificate for SOA
Marketplace Owner
Certification Authority

Figure 8.1: ASSERT Enabled Service Marketplace (AESM) and its stakeholders.

and based on other results from the EU project ASSERT4SOA. A high-level overview of the AESM is shown in Figure 8.1 that presents the different stakeholders interacting with AESM.

The AESM prototype that we propose, demonstrates the application of the Digital Security Certificate concepts proposed in the thesis along with the service discovery concepts proposed in ASSERT4SOA project to a business-oriented service marketplace and highlights key benefits that these approaches can bring in such a scenario. Through the AESM, business consumers can browse a catalogue of certified applications and services, can express their security requirements and compare services based on their certified properties; also, using ASSERT4SOA matchmaking services, the AESM automatically ranks the available candidate services based on how well their certified properties match the user's security requirements. From the standpoint of a business consumer, the AESM covers three key functional areas, outlined below.

(i) **Defining explicit security requirements.** In order to take into account the security requirements of business consumers, the marketplace should allow for expressing these requirements explicitly. In real-life scenarios, there may be different actors that could access this function besides the user that directly subscribes to services from the marketplace (the buyer). In larger organizations there can be a central office in charge for defining the security policies that consumers must comply with when subscribing to a service.

(ii) **Searching based on security requirements.** Service subscribers should be able to use the security requirements to search the marketplace for the services that meets their requirements. The process of matching requirements with the certified properties of the services offered on the marketplace should be as simple and quick as possible, and the security characteristics of each certified service should be readily accessible to the user.

(iii) **Automated matching and approximate solutions.** Among the results of a search, the buyers should be supported in finding the best fitting solution. If no candidate matches the security requirements exactly, the marketplace should suggest the closest match to the security requirements, ultimately allowing the buyer to take a rational decision based on all the information available both about the functionality and the security properties of each candidate.

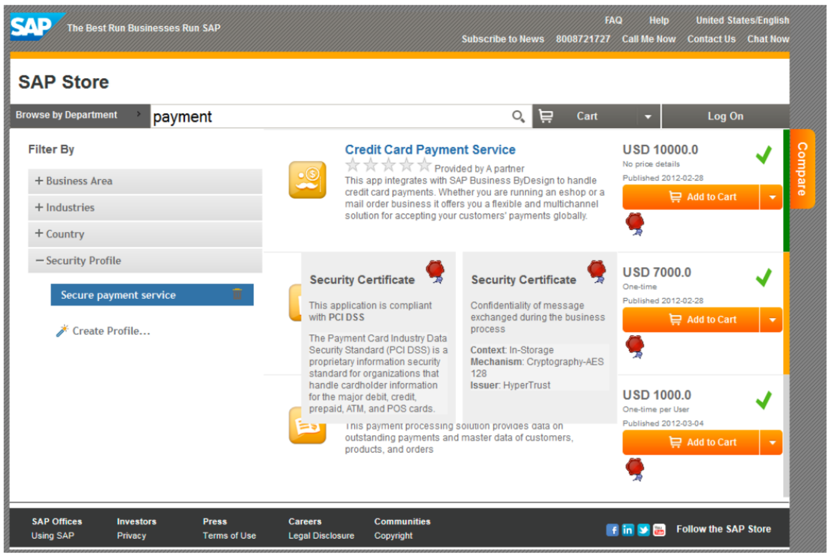
Each of these three points has been implemented in the prototype. Their usage is illustrated in the next section.

8.1.1 Typical usage scenario: a walk-through

The marketplace permits users to search for services on the store with multiple filters, including security properties, keywords and other store specific filters. The workflow below presents a typical usage scenario of the system, specifying how it caters for the requirements of the service consumers, who are responsible for subscribing to business services respecting the security requirements imposed by corporate policies, customers needs, applicable laws, and regulations.

Searching for secure business services

As in all marketplace store-fronts, the user can search the repository of available applications and services, browsing by categories (business areas, industries) and/or specifying one or more keywords in the search field. Differently from existing marketplaces, the security properties of the services in the result list represented explicitly as security certificates (ASSERTS).



Credit Card Payment Service
☆☆☆☆☆ Provided by A partner
This app integrates with SAP Business ByDesign to handle credit card payments. Whether you are running an eshop or a mail order business it offers you a flexible and multichannel solution for accepting your customers' payments globally.

USD 10000.0
No price details
Published 2012-02-28

Security Certificate
This application is compliant with **PCI DSS**
The Payment Card Industry Data Security Standard (PCI DSS) is a proprietary information security standard for organizations that handle cardholder information for the major debit, credit, prepaid, ATM, and POS cards.

Security Certificate
Confidentiality of message exchanged during the business process
Context: In-Storage
Mechanism: Cryptography-AES 128
Issuer: HyperTrust

USD 7000.0
One-time
Published 2012-02-28

This payment processing solution provides data on outstanding payments and master data of customers, products, and orders

USD 1000.0
One-time per User
Published 2012-03-04

Figure 8.2: Sorted (and color-coded) results obtained from applying a policy

Consumers can easily inspect them by clicking on the *seal* icon (Figure 8.2). For each certified service, not only the information about the high-level *certified security property* is available (e.g., "confidentiality of stored data"), but also the mechanism used to achieve that property (e.g., encryption) and its parameters (e.g., cryptography algorithm used: AES with key-length 256 bits). All this information is in fact contained in the ASSERT of the certified service and hence, consumers can trust this information since this is attested by the Certification Authorities. Finally, information about the entity that evaluated the service at hand is available (the certificate issuer) together with the dates of issuing and validity (not shown in the figure, but visible in the details screen for each service).

Refining the search using stored policies

The possibility to review and compare the security properties of the different candidate services is already a significant step forward from existing marketplace instantiations. However, leveraging the fact that ASSERTS are machine-readable, the marketplace can also offer automatic filtering of the candidate services based on the user's security requirements. Of course, a prerequisite for such a feature to work is that the consumer specifies their security requirements. For this purpose, our prototype offers a user-friendly interface (a wizard), partly shown in Figure 8.3.

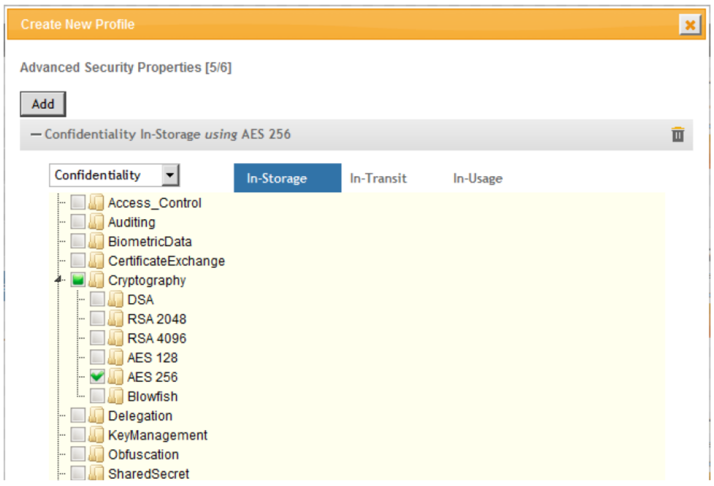


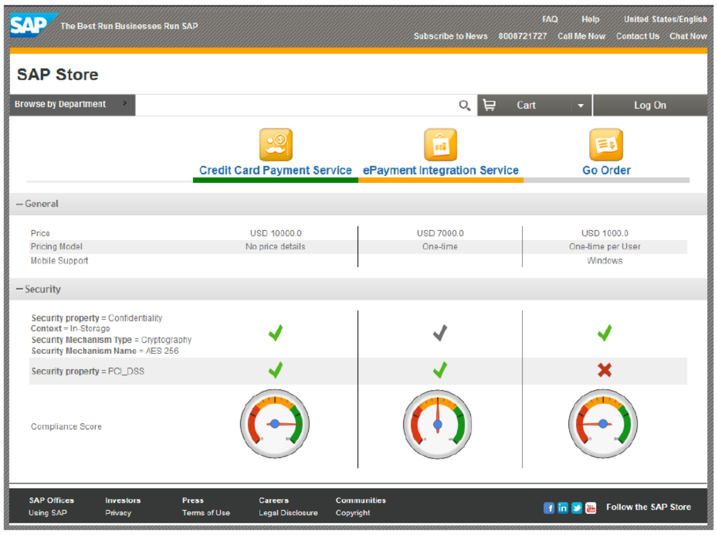
Figure 8.3: Specifying Security Policies (Requirements)

The figure depicts the step of the wizard where the user can express the security properties that candidate services must satisfy, and can indicate what mechanism should be used to achieve those properties. At the end of the wizard, a new policy is created and the corresponding button is added to the left-hand side menu (Figure 8.2). Different policies can be applied at the same time: each may contain constraints of different nature, or coming from different sources. For example, centrally-defined corporate policies could be loaded automatically when the user logs onto the marketplace; in addition to those policies, the user can use the wizard to define and apply additional customized policies covering requirements related to a particular domain, business area, or location. Under the hood, the active policies are translated by the system into a *Query* for the ASSERT4SOA matchmaker service.

After applying one or more policies, the candidate service list is filtered to show only those that match (at least a part of) the constraints expressed in the active policies. The results are ranked and colour-coded to indicate how close a candidate service is to the specified security criteria represented in the applied policies.

Finding the best match among candidate services

From the results list, it is also possible to compare two or more services side by side, to have a clear visual comparison of their security properties and to understand how they compare with the constraints of the active policies (Figure 8.4). In addition to the colour coding (as in the preceding



	Credit Card Payment Service	ePayment Integration Service	Go Order
— General			
Price	USD 10000.0	USD 7000.0	USD 1000.0
Pricing Model	No price details	One-time	One-time per User
Mobile Support			Windows
— Security			
Security property = Confidentiality **Context** = In-Storage **Security Mechanism Type** = Cryptography **Security Mechanism Name** = AES 256	✓	✓	✓
Security property = PCI_DSS	✓	✓	✗
Compliance Score			

Figure 8.4: Comparing the certified security properties of results

step), the score is also visualized by a gauge, indicating in a intuitive way whether the corresponding candidate is a perfect match, a weak match, or a partial match (see the figure, from left to right). Intuitively, a weak match is a candidate that is certified for all the security properties required by the active policies, but that achieves at least one of them using a mechanism other (weaker) than the one specified in the policy. A partial match is a candidate that achieves (perfectly or weakly) at least one of the properties required by the active policies while missing at least one of the other required properties. Services can also have additional ASSERTS, which are not required to meet the security requirements; those ASSERTS are mentioned separately (not shown in Figure 8.4), below the comparison score.

Thus, the AESM prototype showcases the benefits of *consumption* of Digital Security Certificates in service scenarios such as discovery, selection and comparison. A similar approach for certified service discovery, selection and comparison is presented in [147], but it is more oriented towards the developers of service based applications.

8.2 ASSERT Management Tool

In this section, we present a research prototype that is based on the ASSERT language presented in Section 6.4, to generate the Digital Security Certificates (DSCs) and to check the conformance of a certificate against a Digital Security Certificate Profile. We have contributed to the development of this prototype within the context of the EU funded project ASSERT4SOA. While the AESM prototype showcases the ease of consumption of Digital Security Certificates, the ASSERT Management Tool (AMT) illustrates the ease of generation of Digital Security Certificates.

We need tool support for the core profile-based certificate management operations:

- *standalone generation* of DSCs
- *profile-based generation* of DSCs
- *profile conformance verification* of DSCs.

These are the most relevant DSC management operations a DSC issuer would need to perform when issuing DSCs.

8.2.1 Standalone generation of Digital Security Certificates

Standalone generation of Digital Security Certificates is the most simplest way to generate a certificate, where the Certification Authority captures the information from the certification process and represents in the DSC instance conforming to the ASSERT language schema. However, certificates produced without conforming to any DSC profiles increase the complexity in comparing different certified services.

The process of generation of a standalone-ASSERT is straightforward through our prototype, wherein the various fields can be filled using different ontologies that are integrated within the tool.

8.2.2 Profile-based Generation of Digital Security Certificate

Figure 8.5 shows the process for profile- based generation of DSCs. When a DSC profile is selected and loaded, there is a pre-processing step for all dynamic vocabulary specifications. If some dynamic vocabulary specifications depend on other artefacts and values in order to be processed, these vocabularies are processed at the time when the issuer creates the corresponding artefacts.

- **Step 1:** *Initialize DSC content*. Once the profile is processed, first duplicate the certificate template and create an bare-bones certificate instance with an initial structure and content of the duplicated template data.

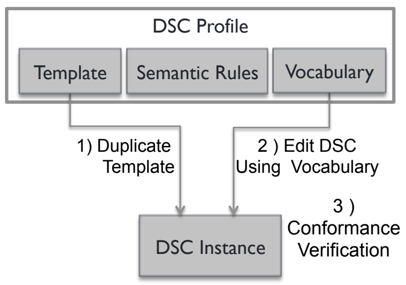


Figure 8.5: Profile-based Generation of Digital Security Certificate

- **Step 2:** *Edit DSC content*. Next step is the actual process of editing the certificate artefacts and creation of new artefacts as needed by the issuer. This step heavily relies on the use of certificate vocabulary defined in the profile. When an artefact's vocabulary is specified as mandatory, the tool will enforce the choice of the vocabulary terms. Otherwise, if optional, the tool will recommend/ suggest a choice of terms but leaving the issuer to specify own terms when as found necessary.

- **Step 3:** *Profile Conformance Verification*. The final certificate instance is verified for conformance to the profile (presented in the next subsection). Any violations of the DSC instance against the DSC profile including the used artefacts and corresponding vocabularies are reported to the Certificate Issuer. Moreover, it reverts back to the DSC edit mode (Step 2) until no further violations are found.

8.2.3 Profile Conformance Verification of DSC

A prerequisite to conformance verification is to ensure whether the certificate instance conforms to the syntax of a given certificate model, that is, verification of the syntactic correctness of the certificate instance. Otherwise, the verifier should not proceed with the verification process.

Figure 8.6 shows the three main steps of conformance verification process.

- **Step 1:** *Structure validation*. DSC structure is validated if it contains all required elements and values as declared in the profile template.

- **Step 2:** *Vocabulary Validation*. DSC vocabulary is validated for compliance with the vocabulary defined in the profile. Prerequisite to this step is to first process all dynamic vocabularies. That is, retrieving all certificate artefacts' vocabulary terms from the corresponding

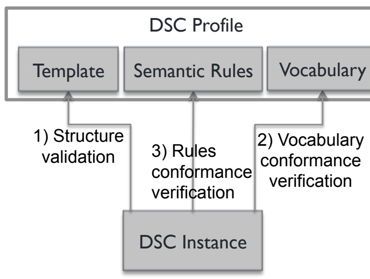


Figure 8.6: Profile Conformance Verification of DSC

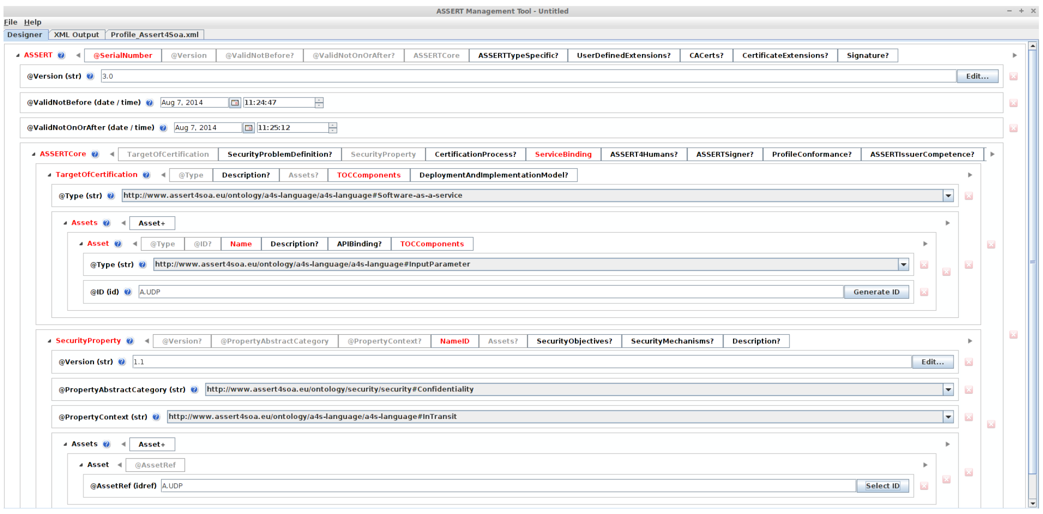
ontologies by executing the queries. Once dynamic vocabularies are instantiated, all certificate artefacts' vocabulary terms within the vocabulary part are checked against the corresponding artefacts' content in the certificate instance. All certificate artefacts defined to have an optional (non-mandatory) vocabulary will not be verified for conformance.

- **Step 3:** *Semantic Validation*. DSC structure and vocabulary is validated for compliance with the profile rules, that is, if all constraints are satisfied. All semantic rules are processed, checked if satisfied by the certificate structure and content. Since the semantic rules of the profile may depend on the actual content (vocabulary) of a certificate artefacts in order to determine the semantic integrity of the certificate content, it is important to verify vocabulary conformance first, and then the semantic rules conformance.

8.2.4 DSC Management Tool Realization

A DSC management tool has been developed to support the management of ASSERTS from the perspective of ASSERT issuers. The tool is called ASSERT Management Tool (AMT) [15]. The AMT is designed to provide an intuitive GUI for standard ASSERT management operations including complete support of *multi-profile* based ASSERT management. We will present the GUI of the AMT for the case of profile-based creation of ASSERTS.

Figure 8.7 shows the AMT designer view. The designer view provides content-centric ASSERT management: abstracting issuers from underlying XML representation. All mandatory ASSERT language elements are coloured in red for convenience of issuers. A help icon is shown to each element name describing the rationale of the selected element. The AMT has the ASSERT language schema built-in. A designer pane shows all language elements as buttons. Upon pressing a button, the corresponding data element



@Version (str) 3.0
@ValidNotBefore (date / time) Aug 7, 2014 11:24:47
@ValidNotOnOrAfter (date / time) Aug 7, 2014 11:25:12
@Type (str) http://www.assert4soa.eu/ontology/a4s-language/a4s-language#Software-as-a-service
@Type (str) http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InputParameter
@ID (id) A.UDP
@Version (str) 1.1
@PropertyAbstractCategory (str) http://www.assert4soa.eu/ontology/security/security#Confidentiality
@PropertyContext (str) http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InTransit
@AssetRef (idref) A.UDP

Figure 8.7: AMT Designer View

is created.

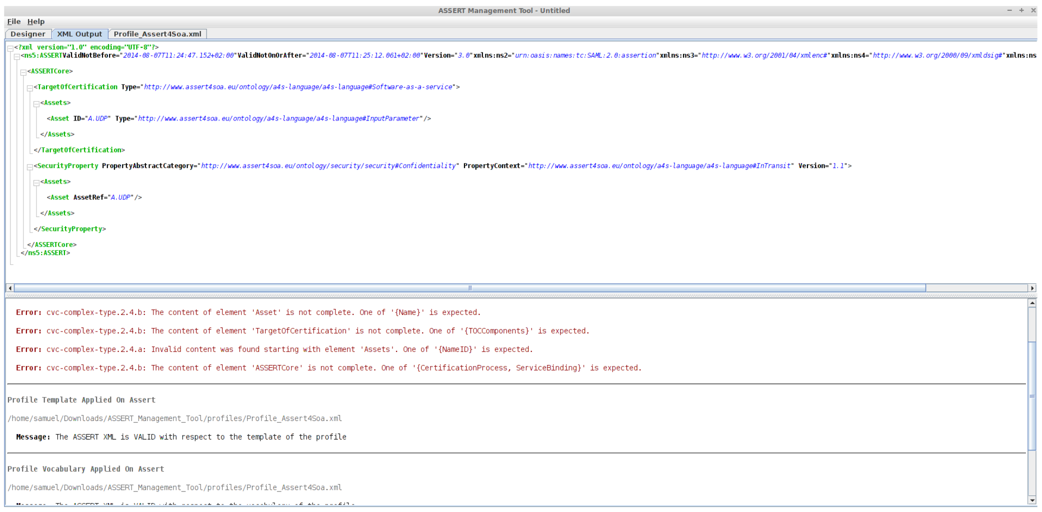
As we can see in Figure 8.7, the ASSERT profile has been already loaded and the corresponding profile vocabulary processed showing the vocabulary terms defined for the *PropertyContext* element of the *SecurityProperty*

Figure 8.8 shows the ASSERT XML view. The upper part of the XML Output shows a non-editable XML view of the current ASSERT structure. The bottom part shows the result of ASSERT validation against the ASSERT language, and the results of profile conformance verification against the selected ASSERT profile. The messages of profile conformance verification are grouped into the corresponding verification steps as described earlier (ref. Figure 8.6). Grouping messages greatly facilitates issuers especially when dealing with creation of ASSERTS based on multiple profiles.

Figure 8.9 shows the AMT profile view. The upper part of the profile view shows the content of the selected profile, while the bottom part of the view shows messages of the profile validation process. The profile validation process ensures the issue that the selected profile has well-formed vocabulary section and semantic rules sections, in our case syntactically correct Schematron rules.

We note that in case of multiple profiles, the AMT loads each profile in a separate profile view (tab) so that the issuer can at any moment check if any of the profiles is well-formed and what messages are shown.

Thus, we show the ease with Digital Security Certificates can be generated and can be validated for consistency and conformance against DSC Profiles using tool support, that exploits the machine processability of the



```
<?xml version="1.0" encoding="UTF-8"?>
<ns5:ASSERTValidNotBefore="2014-08-07T11:24:47.152+02:00"ValidNotOnOrAfter="2014-08-07T11:25:12.061+02:00"Version="3.0"xmlns:ns2="urn:oasis:names:tc:SAML:2.0:assertion"xmlns:ns3="http://www.w3.org/2001/04/xmlenc#"xmlns:ns4="http://www.w3.org/2000/09/xmldsig#"xmlns:ns

  <ASSERTCore>

    <TargetOfCertification Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#Software-as-a-service">

      <Assets>

        <Asset ID="A.UDP" Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InputParameter"/>

      </Assets>

    </TargetOfCertification>

    <SecurityProperty PropertyAbstractCategory="http://www.assert4soa.eu/ontology/security/security#Confidentiality" PropertyContext="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InTransit" Version="1.1">

      <Assets>

        <Asset AssetRef="A.UDP"/>

      </Assets>

    </SecurityProperty>

  </ASSERTCore>
</ns5:ASSERT>
```

```
Error: cvc-complex-type.2.4.b: The content of element 'Asset' is not complete. One of '{Name}' is expected.

Error: cvc-complex-type.2.4.b: The content of element 'TargetOfCertification' is not complete. One of '{TOCComponents}' is expected.

Error: cvc-complex-type.2.4.a: Invalid content was found starting with element 'Assets'. One of '{NameID}' is expected.

Error: cvc-complex-type.2.4.b: The content of element 'ASSERTCore' is not complete. One of '{CertificationProcess, ServiceBinding}' is expected.

Profile Template Applied On Assert

/home/samuel/Downloads/ASSERT_Management_Tool/profiles/Profile_Assert4Soa.xml

Message: The ASSERT XML is VALID with respect to the template of the profile

Profile Vocabulary Applied On Assert

/home/samuel/Downloads/ASSERT_Management_Tool/profiles/Profile_Assert4Soa.xml
```

Figure 8.8: AMT XML Output View

ASSERT artefacts.

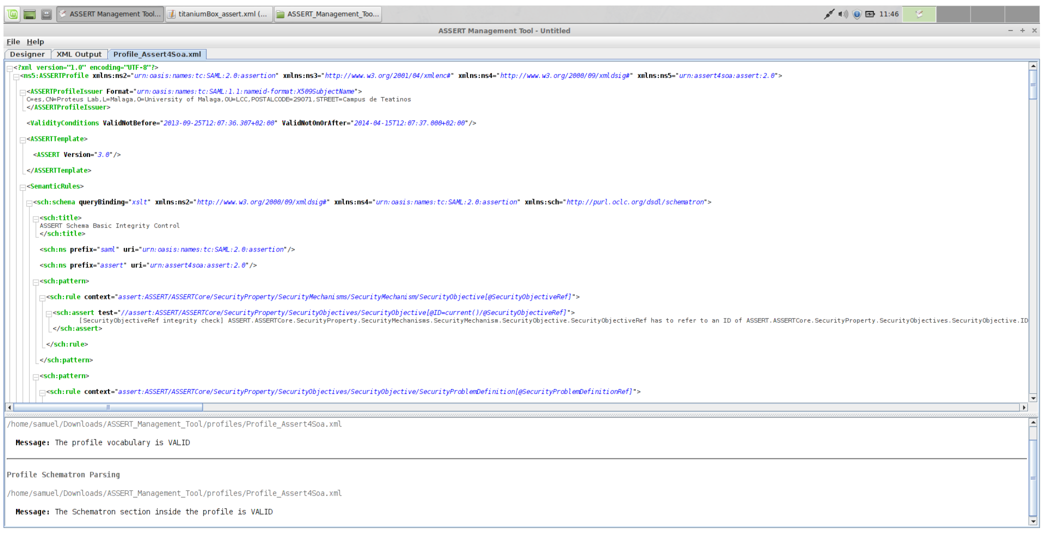
8.3 DSC Impact on Security Certification Practices

The current security certification practices would be profoundly impacted by the $\mathcal{CRT}$ concept we proposed. The $\mathcal{CRT}$ model allows certification results to be captured in a machine processable form thereby allowing automated reasoning to be performed on them. Automated reasoning on security certificates opens up hitherto unexplored scenarios such as instantaneous comparison of certified products, requirements compliance by a certified products, fine grained and precise description of secured elements within a certified product leading to less ambiguity or errors in interpretation of the security certificates.

In addition, the AMT tool can be used by the certification authorities to issue security certificates in a much faster manner reducing the previously time consuming process of producing certificate artefacts.

8.3.1 From the Certification Authority's Perspective

The AMT tool and its output artefacts were evaluated by a focus group composed of domain experts, two Common Criteria certifiers and two experts in security certification. The activities comprised an explanation of


ASSERT Management Tool...
titaniumBox_assert.xml (...
ASSERT_Management_Too...
11:46
ASSERT Management Tool - Untitled
File
Help
Designer
XML Output
Profile_Assert4Soa.xml

```
<?xml version="1.0" encoding="UTF-8"?>
 <ns5:ASSERTProfile xmlns:ns2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:ns3="http://www.w3.org/2001/04/xmlenc#" xmlns:ns4="http://www.w3.org/2000/09/xmldsig#" xmlns:ns5="urn:assert4soa:assert:2.0">

  <ASSERTProfileIssuer Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
  C=es,CN=Proteus Lab,L=Malaga,O=University of Malaga,OU=LCC,POSTALCODE=29071,STREET=Campus de Teatinos
  </ASSERTProfileIssuer>

  <ValidityConditions ValidNotBefore="2013-09-25T12:07:36.307+02:00" ValidNotOnOrAfter="2014-04-15T12:07:37.000+02:00"/>

  <ASSERTTemplate>

   <ASSERT Version="3.0"/>

  </ASSERTTemplate>

  <SemanticRules>

   <sch:schema queryBinding="xslt" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:sch="http://purl.oclc.org/dsdl/schematron">

    <sch:title>
    ASSERT Schema Basic Integrity Control
    </sch:title>

    <sch:ns prefix="saml" uri="urn:oasis:names:tc:SAML:2.0:assertion"/>

    <sch:ns prefix="assert" uri="urn:assert4soa:assert:2.0"/>

    <sch:pattern>

     <sch:rule context="assert:ASSERT/ASSERTCore/SecurityProperty/SecurityMechanisms/SecurityMechanism/SecurityObjective[@SecurityObjectiveRef]">

      <sch:assert test="//assert:ASSERT/ASSERTCore/SecurityProperty/SecurityObjectives/SecurityObjective[@ID=current()/@SecurityObjectiveRef]">
            [SecurityObjectiveRef integrity check] ASSERT.ASSERTCore.SecurityProperty.SecurityMechanisms.SecurityMechanism.SecurityObjective.SecurityObjectiveRef has to refer to an ID of ASSERT.ASSERTCore.SecurityProperty.SecurityObjectives.SecurityObjective.ID
      </sch:assert>

     </sch:rule>

    </sch:pattern>

    <sch:pattern>

     <sch:rule context="assert:ASSERT/ASSERTCore/SecurityProperty/SecurityObjectives/SecurityObjective/SecurityProblemDefinition[@SecurityProblemDefinitionRef]">
```

```
/home/samuel/Downloads/ASSERT_Management_Tool/profiles/Profile_Assert4Soa.xml

  Message: The profile vocabulary is VALID

Profile Schematron Parsing

/home/samuel/Downloads/ASSERT_Management_Tool/profiles/Profile_Assert4Soa.xml

  Message: The Schematron section inside the profile is VALID
```

Figure 8.9: AMT Profile View

the basic underpinnings for DSC, ASSERT and ASSERT Profile concepts, as well as an evaluation of the tool operations and outputs. The evaluation consisted of a questionnaire composed mostly by closed-answer questions using 5-points Likert scale [101], as well as a number of free answers questions. The latter allowed for the expression of feedback on specific DSC aspects, which was used later on for improving the AMT tool. The questionnaire was structured in two sections: the first part was devoted to an usability assessment [146], while the second aimed at assessing the suitability of AMT tool outputs and procedures (e.g. ASSERTs and ASSERT validation against an ASSERT Profile) for their application in a security certification process operations.

The results of an analysis of the focus group can be summarized as follows. Regarding the usability of the AMT tool, it was assessed positively, especially considering its nature of research prototype. Nevertheless, several suggestions were proposed, and some of them have been incorporated in the prototype; others were deemed interesting for future commercial exploitation of the concept. With respect to the suitability of ASSERTs and AMT tool operations to security certification operations (especially considering Common Criteria), we will focus our attention on a selection of the focus group questions, which are:

Q-15 From the point of view of a Certifier, an ASSERT is suitable to represent a typical security certificate.

Q-16 The AMT speeds up the ASSERT management process.

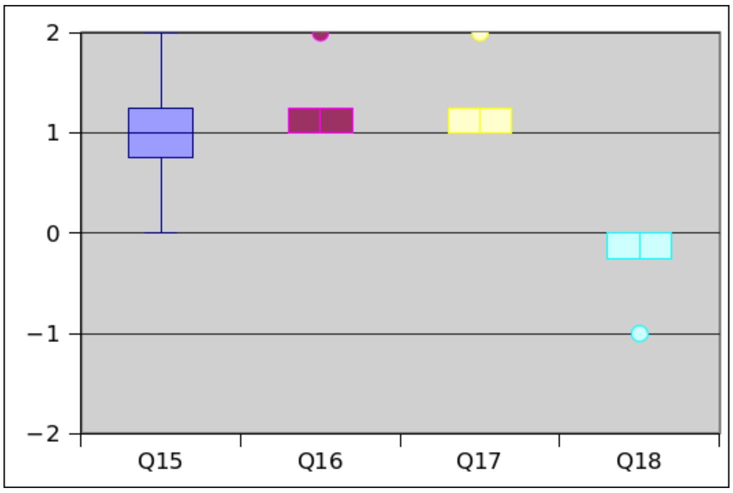


Figure 8.10: BoxPlot of selected DSC Tool assessment questions

Q-17 The AMT improves the control over the ASSERT management process.

Q-18 From the point of view of a Certifier, the automation of a typical security certificate management process is a priority.

Their proposed answers (and their mapping to the 5-points Likert scale) were : I fully agree (2), I mostly agree (1), I neither agree nor disagree (0), I mostly disagree" (-1) and I completely disagree" (-2). The answers are represented in Fig. 8.10.

The answers to Q-15, Q-16 and Q-17 are particularly encouraging: Q-15 essentially confirms the suitability of the ASSERT and DSC concepts for representing security certificate contents, while Q-16 and Q-17 assess positively the AMT tool operations.

From Q-18 and open answer questions, it is possible to derive the following findings. From the point of view of the technical quality of the AMT, most of the respondents generally agreed that the representation capability of an ASSERT is suitable from the certifier's point of view. However, from the point of view of its adoption and relevance, the main output of the validation session indicates that current real world certification processes (and especially Common Criteria) are probably not ready to embrace digital certificates in their current forms. For Common Criteria, this is mainly due to its complexity and difficulty to automate its operations. However, the advantages of using ASSERT and DSC Tool can represent a stimulus for the evolution of a debate inside the Common Criteria community.

8.3.2 From the Consumer's Perspective

Typically, service consumers range from regular end-users to security experts with varying understanding, awareness, needs and requirements on service security. $\mathcal{CRT}$ caters to each type of consumer by allowing them to understand the certified security properties of a service at varying levels of details.

Moreover, $\mathcal{CRT}$ helps security conscious consumers to inspect in detail the security measures implemented in the service. Security experts can investigate the architecture of the service through the *Target of Certification* element, which clearly identifies the different components of the services. The *Target of Certification* informs the consumers whether the service is composed of other external services. In fact, consumers can find out whether the certified security properties of a service depend on the security properties of any external services that are used. Consumers can then verify whether the external service possesses a security certificate with those certified security properties. With additional tool support this process can be completely automated, and a consumer can easily verify a chain of security certificates.

The DSCM framework allows consumers to verify, during consumption time, whether a service still has a valid security certificate. This allows consumers to know the *current* state of the service's certified security properties, rather than depend on certified properties based on the service implementation during certification time. This significantly improves the *usefulness* of the security certification schemes in a service environment.

Conclusions

In this chapter we have presented various prototypes that were built exploiting the concepts proposed in this thesis. We have presented an industrial prototype of a Service marketplace that allows service consumers to search, discover and compare services based on their certified security properties - thus showcasing the consumption of the Digital Security Certificates. In addition, we have presented a research prototype of a tool that is used to generate Digital Security Certificates, showing the ease with which certification authorities can issue and validate certificates.

Finally, we have presented a brief overview of the survey conducted on a focus group consisting of certification experts the results of which further encourage our belief that the contributions of this thesis can have a broad and positive impact on the security certification landscape.

Publications

The contents of this chapter are further discussed in the following publication:

⋆ **S. P. Kaluvuri**, H. Koshutanski, F. Di Cerbo, R. Menicocci, and A. Maña.*A digital security certificate framework for services*. International Journal of Services Computing, 1(1), 2013.

Technical Reports

A more comprehensive discussion about the topics mentioned in this chapter are provided in the following technical reports:

⋆ H. Koshutanski, **S.P. Kaluvuri**, A. Maña, M. Montenegro, M. Anisetti, C. Ardagna, E. Damiani, S. Gürgens, D. Presenza. *ASSERT Language V1.4*. ASSERT4SOA Project Deliverable(D1.4), 2014.

⋆ M. Bezzi, S. D'Agostini, **S.P. Kaluvuri**, A. Maña, C. Pandolfo, G. Pujol, A. Sabetta. *Architecture and High-level Design of ASSERT4SOA Framework*. ASSERT4SOA Project Deliverable(D6.1), 2014.

Chapter 9

Conclusions and Future Directions

"I discovered that searching can be as interesting as finding."

— Paulo Coelho

Security assurance is an important aspect to be addressed in service environments in order to facilitate a much wider adoption of service based solutions, especially in business critical domains. In this thesis, we examined the various means to gain security assurance in traditional provisioning models, and analysed their applicability in service environments. We showed that security certification is a scalable and efficient solution to gain security assurance in service environments.

The first part of this thesis, presents a comprehensive analysis of the security certification landscape. We categorize the security certification schemes into process and product based certification schemes and explain the assurances that can be gained through these certification schemes. We focus on Common Criteria security certification scheme, since it is the most widely used, recognized and broadly applicable scheme. We explained in detail about the CC certification *scheme* – the certification process, the certification documents that result from the certification process, the evaluation methodologies among others.

One of the key contributions of the thesis is the quantitative analysis of the Common Criteria security certification *practice* that we have performed in order to understand whether the CC *practice* stays true to the intent of the CC *scheme*. We presented the results of the analysis which highlighted several limitations of the CC practice that raises concerns over the assurance consumers gain through CC certification.

We closely examined the various security certification schemes – both that cater to traditional provisioning models and those that are tailored for service environments – to evaluate their applicability and effectiveness in service environments. We discovered that most of these certification schemes do not take into account the usability of security certificates in service environments, such as service discovery and service selection and they do not address a key requirement in service environments – assurance over the IT Operational environment, where the service is executed.

In the second part of the thesis, we presented the conceptual model of a Digital Security Certificate. Based on the conceptual model, we have implemented an XML based language that allows service security certificates to represented in a machine readable format. We further showcased the flexibility of our language to capture information coming from various certification schemes. This facilitates a wider adoption of the Digital Security Certificate concept by different security certification schemes. Moreover, we proposed the concept of a *Digital Security Certificate Profile* which can be used by certification authorities to prescribe certain constraints on the security certificates that are issued. Digital Security Certificate Profiles can also be used by consumers to state their security requirements and verify, with tool support, whether a security certificate conforms to their profile.

The thesis presented an approach that facilitates certification authorities to monitor certified services and its operational environment to ensure that the certified properties are always satisfied. Our approach also takes into account the frequent updates that are made to services and thus has a mechanism that deduces whether an update to a service implementation violates any certified properties.

In the final part of the thesis, we present a scenario of a secure service marketplace that demonstrates that the Digital Security Certificate concept can cater to the needs of service environments and in particular to service discovery process.

The contributions of the thesis have the potential to enable service adoption in business critical domains such as healthcare, defence, finance among others. It also enables standardization of security certificate representation among various schemes.

In fact, within the research community the contributions of the thesis have been used as a basis for digital security certificate artefacts used within several research projects (ASSERT4SOA [16], OPTET [134], CUMULUS [56]).

Future Directions

There are several open issues that must be further investigated such as the criteria that is used in different certification scheme for evaluation software is not standardized and still some of these have to be tested or proven manually. Thus extending the time to gain certification which can prove critical in service environments where time to market is extremely crucial. Hence, we plan to investigate further on different approaches that can automatically evaluate the services for certain security properties.

Another open issue that needs to be investigated further is the security certificate composition. Reasoning mechanisms need to be developed that can determine at runtime whether a certified service can bind with another certified service in a manner that the security properties of the first service are not violated.

Bibliography

[1] NIST National Vulnerability Database. National vulnerability database, 2012. Available at `http://nvd.nist.gov/`.

[2] Agence nationale de la sécurité des systèmes d'information (ANSSI). Certification de Sécurité de Premier Niveau, 2014. Accessed on 26-09-2014. Available at `http://www.ssi.gouv.fr/fr/certification-qualification/cspn/`.

[3] M. Almorsy, J. Grundy, and I. Müller. An analysis of the cloud computing security problem. In *Proceedings of APSEC 2010 Cloud Workshop, Sydney, Australia, 30th Nov*, 2010.

[4] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services*. Springer, 2004.

[5] Amazon Web Services. Aws marketplace. Available at `https://aws.amazon.com/marketplace`.

[6] Amazon Web Services. Aws compliance, 2007. Available at `http://aws.amazon.com/compliance/`.

[7] Amazon Web Services. Amazon web services, 2014. Available at `http://aws.amazon.com/`.

[8] J. P. Anderson. Computer security technology planning study. volume 2. Technical report, DTIC Document, 1972.

[9] M. Anisetti, C. Ardagna, and E. Damiani. Defining and matching test-based certificates in open SOA. In *Proc. of the Second International Workshop on Security Testing (SECTEST 2011)*, pages 520–522, 2011.

[10] M. Anisetti, C. A. Ardagna, E. Damiani, C. Pandolfo, and A. Maña. D4.1 Design and description of evidence-based certificates artifacts for services. Technical report, ASSERT4SOA Project, 2011. Available at http://www.assert4soa.eu/deliverable/D4.1.pdf.

[11] Apache Software Foundation. Apache tomcat, 2014. Available at `http://tomcat.apache.org/`.

[12] Apple inc. Apple Answers FCC's Questions. http://www.apple.com/hotnews/apple-answers-fcc-questions/. Available at `http://www.apple.com/hotnews/apple-answers-fcc-questions/`.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[14] C. Arthur. Nsa scandal: what data is being monitored and how does it work. *The Guardian*, 2013.

[15] ASSERT4SOA. Assert management tool. Available at `http://proteus.lcc.uma.es/projects/assert4soa/software/`.

[16] ASSERT4SOA. Assert4soa - advanced security service certifcate for soa. Available at `http://assert4soa.eu/`.

[17] Assurance Assessment Services, CESG. Commercial Product Assurance International Aspects, 2012. Accessed on 10-09-2012. Available at `https://www.cesg.gov.uk/Publications/Documents/cpa_international_aspects.pdf`.

[18] J. Bacon, D. Eyers, T. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch. Information flow control for secure cloud computing. 2014.

[19] D. Barrera and P. Van Oorschot. Secure software installation on smartphones. *Security & Privacy, IEEE*, 9(99):42–48, 2010.

[20] Baymard Institute. Which site seal do people trust the most? (2013 survey results), 2013. Available at `http://baymard.com/blog/site-seal-trust`.

[21] B. Beckert, D. Bruns, and S. Grebing. Mind the gap: Formal verification and the common criteria (discussion paper).

[22] D. Beimborn, T. Miletzki, and S. Wenzel. Platform as a service (paas). *Business & Information Systems Engineering*, 3(6):381–384, 2011.

[23] P. Benassi. Truste: an online privacy seal program. *Communications of the ACM*, 42(2):56–59, 1999.

[24] K. Beznosov and P. Kruchten. Towards agile security assurance. In *Proceedings of the 2004 workshop on New security paradigms*, pages 47–54. ACM, 2004.

[25] M. Bezzi, A. Sabetta, and G. Spanoudakis. An architecture for certification-aware service discovery. pages 14–21, 2011.

[26] S. Bhardwaj, L. Jain, and S. Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

[27] D. Birk and C. Wegener. Technical issues of forensic investigations in cloud computing environments. In *Systematic Approaches to Digital Forensic Engineering (SADFE), 2011 IEEE Sixth International Workshop on*, pages 1–10. IEEE, 2011.

[28] M. Bishop. What is computer security? *Security & Privacy, IEEE*, 1(1):67–69, 2003.

[29] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, 2009.

[30] M. A. Branstad, D. Brewer, C. Jahl, H. Kurth, and C. Pfleeger. Apparent differences between the us tcsec and the european itsec. In *Proc. 14th National Computer Security Conference*, pages 45–58, 1991.

[31] S. Brew and C. Preece. Is dropbox fit for business?, 2014. Available at `http://www.itpro.co.uk/cloud/20333/dropbox-fit-business`.

[32] Bruce Guptill. Saugatuck 2012 SaaS Survey Indicates More User Focus on Business Improvement, With Big Changes Coming Soon, 2012.

[33] U. BSI. Moving from iso/iec, 27001: 2005 to iso/iec, 27001: 2013, 2013.

[34] L. A. Bygrave. *Data protection law: approaching its rationale, logic and limits*. Kluwer Law Intl, 2002.

[35] L. A. Bygrave. Privacy and data protection in an international perspective. *Scandinavian studies in law*, 56:165–200, 2010.

[36] CBBB. Council of better business bureaus, 2014. Available at `http://www.bbb.org/council/about/council-of-better-business-bureaus/`.

[37] D. Challener, K. Yoder, R. Catherman, D. Safford, and L. Van Doorn. *A practical guide to trusted computing*. Pearson Education, 2007.

[38] S.-C. Chang and C.-F. Fan. Construction of an ontology-based common criteria review tool. In *Computer Symposium (ICS), 2010 International*, pages 907–912, 2010.

[39] K.-K. R. Choo. The cyber threat landscape: Challenges and future research directions. *Computers & Security*, 30(8):719 – 731, 2011.

[40] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 85–90. ACM, 2009.

[41] Cisco and Intel. Common criteria embrace, reform, extend. discussion draft 1.0. 2011.

[42] Cluley Associates Limited. Dropbox told about vulnerability in november 2013, only fixed it when the media showed interest, 2014. Available at `http://grahamcluley.com/2014/05/dropbox-vulnerability-privacy/`.

[43] Common Criteria. Common Criteria Assurance Continuity:CCRA Requirements, 2004. Available at `http://www.commoncriteriaportal.org/files/supplements/2004-02-009.pdf`.

[44] Common Criteria. Common criteria: Certified products list - statistics, 2012.

[45] Common Criteria. Common Criteria Part 1: introduction and general model, 2012. Accessed on 03-03-2012. Available at `http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf`.

[46] Common Criteria. Common Criteria Part 2: security functional requirements, 2012. Accessed on 03-03-2012. Available at http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf.

[47] Common Criteria. Common Criteria Part 3: security assurance requirements, 2012. Accessed on 03-03-2012. Available at http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf.

[48] Common Criteria. Common criteria portal, 2012. Available at http://www.commoncriteriaportal.org/.

[49] Common Criteria. Common criteria protection profile list, 2012. Available at http://www.commoncriteriaportal.org/pps/.

[50] Common Criteria. Common criteria recognition agreement, 2012.

[51] Common Criteria. Common criteria security targets, 2012. Available at http://www.commoncriteriaportal.org/products/.

[52] E. Communities-Commission et al. Itsec: Information technology security evaluation criteria (provisional harmonised criteria, version 1.2, 28 june 1991), 1991.

[53] CPA Canada. Webtrust: Overview of trust services, 2014. Available at http://www.webtrust.org/overview-of-trust-services/item64420.aspx.

[54] CSA. Cloud security alliance, 2014. Available at https://cloudsecurityalliance.org/.

[55] CSA. Csa security, trust and assurance registry (star), 2014. Available at https://cloudsecurityalliance.org/star/.

[56] CUMULUS. Cumulus - certification infrastructure for multi-layer cloud services. Available at http://www.cumulus-project.eu/.

[57] E. Damiani, C. A. Ardagna, and N. El Ioini. *Open source systems security certification*. Springer, 2008.

[58] F. Di Cerbo and S. Trabelsi. USDL-SEC homepage. https://github.com/linked-usdl/usdl-sec (3 March 2013), 2012.

[59] T. Dillon, C. Wu, and E. Chang. Cloud computing: issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27–33. Ieee, 2010.

[60] S. Dresner. Data protection roundup. *Privacy Laws and Business Newsletter (January)*, pages 2–8, 1996.

[61] Dropbox inc. Dropbox security overview, 2013. Accessed on 03-03-2013. Available at `http://www.dropbox.com/dmca#security`.

[62] Dropbox Inc. Introducing pyston: an upcoming, jit-based python implementation, 2014. Available at `https://tech.dropbox.com/2014/04/introducing-pyston\-an-upcoming-jit-based-python-implementation/`.

[63] Dropbox Inc. Web vulnerability affecting shared links, 2014. Available at `https://blog.dropbox.com/2014/05/web-vulnerability-affecting\-shared-links/`.

[64] L.-E. Easley. Spying on allies. *Survival*, 56(4):141–156, 2014.

[65] A. Ekelhart, S. Fenz, G. Goluch, and E. R. Weippl. Ontological mapping of common criteria's security assurance requirements. In *SEC*, pages 85–95, 2007.

[66] M. M. Eloff and S. H. Von Solms. Information security management: an approach to combine process certification and product evaluation. *Computers & Security*, 19(8):698–709, 2000.

[67] H. Farrell. Constructing the international foundations of e-commerce—the eu-us safe harbor arrangement. *International Organization*, 57(02):277–306, 2003.

[68] S. Fenz, G. Goluch, A. Ekelhart, B. Riedl, and E. R. Weippl. Information security fortification by ontological mapping of the iso/iec 27001 standard. In *PRDC*, pages 381–388, 2007.

[69] Forbes. Yes, Celebs Had Their iCloud Accounts Hacked. No, You Shouldn't Shut Yours Off, 2014. Available at `http://www.forbes.com/sites/markrogowsky/2014/09/03/the-celeb-hack-has-people-telling-you-to-turn\-off-cloud-backup-ignore-them/`.

[70] B. Friedman, P. H. Khan Jr, and D. C. Howe. Trust online. *Communications of the ACM*, 43(12):34–40, 2000.

[71] A. Fuchs and S. Gürgens. D5.1 Formal models and model composition. Technical report, ASSERT4SOA Project, 2011. Available at http://www.assert4soa.eu/deliverable/D5.1.pdf.

[72] Gartner. Forecast overview: Public cloud services. report G00234817, 2012.

[73] Gartner Inc. Gartner Says SAS 70 Is Not Proof of Security, Continuity or Privacy Compliance, 2010. Available at http://www.gartner.com/newsroom/id/1400813.

[74] Gartner Inc. Gartner Says Worldwide Traditional PC, Tablet, Ultramobile and Mobile Phone Shipments to Grow 4.2 Percent in 2014, 2014. Available at http://www.gartner.com/newsroom/id/2791017.

[75] Google inc. Evaluate a marketplace app's security. https://support.google.com. Available at https://support.google.com/a/bin/answer.py?hl=en&answer=180490&topic=1056394&ctx=topic.

[76] Google Inc. Google apps marketplace. Available at https://www.google.com/enterprise/marketplace/.

[77] Google Inc. 2012 Update from the CEO, 2012. Available at http://investor.google.com/corporate/2012/ceo-letter.html.

[78] D. GREILING and K. Spraul. Accountability and the challenges of information disclosure. *Public Administration Quarterly*, pages 338–377, 2010.

[79] T. O. Group. Socci framework technical standard : Cloud computing characteristics, 2014. Available at http://www.opengroup.org/soa/source-book/socci/cc_char.htm.

[80] D. S. Herrmann. *Using the Common Criteria for It Security Evaluation*. CRC Press, Inc., Boca Raton, FL, USA, 2002.

[81] A. Herzog, N. Shahmehri, and C. Duma. An ontology of information security. *International Journal of Information Security*, 1(4):1–23, 2007.

[82] H. Hoffmann, J. Eastep, M. D. Santambrogio, J. E. Miller, and A. Agarwal. Application heartbeats for software performance and health. In *ACM Sigplan Notices*, volume 45, pages 347–348. ACM, 2010.

[83] R. Housley and T. Polk. *Planning for PKI: best practices guide for deploying public key infrastructure*. John Wiley & Sons, Inc., 2001.

[84] Z. Huanguo, L. Jie, J. Gang, Z. Zhiqiang, Y. Fajiang, and Y. Fei. Development of trusted computing research. *Wuhan University Journal of Natural Sciences*, 11(6):1407–1413, 2006.

[85] K.-L. Hui, H. H. Teo, and S.-Y. T. Lee. The value of privacy assurance: an exploratory field experiment. *Mis Quarterly*, pages 19–33, 2007.

[86] IBM. WebSphere Software, 2014. Available at `http://www-01.ibm.com/software/websphere/`.

[87] A. Inc. icloud, 2014. Available at `http://icloud.com`.

[88] G. Inc. Gmail, 2014. Available at `http://google.com/mail`.

[89] M. Inc. Onedrive plans, 2014. Available at `https://onedrive.live.com/about/en-us/plans/`.

[90] ISMG. ISMG Announces Release of 2012 Cloud Computing Security Survey Results, 2014. Available at `http://www.ismgcorp.com/press/ismg-announces-release-\2012-cloud-computing-security-survey-results-p-315`.

[91] I. Iso. Iec 27001: 2005. *Information Technology. Security Techniques. Information Security Management Systems. Requirements*, 2005.

[92] T. Jordan and P. Taylor. A sociology of hackers. *The Sociological Review*, 46(4):757–780, 1998.

[93] E. Joyce. Software bugs: a matter of life and liability. *Datamation*, 33(10):88–92, 1987.

[94] J. Kallberg. Common criteria meets realpolitik - trust, alliances, and potential betrayal. *Security Privacy, IEEE*, PP(99):1, 2012.

[95] S. P. Kaluvuri, M. Bezzi, A. Sabetta, Y. Roudier, R. Menicocci, V. Bagini, A. Riccardi, and M. Orazi. Applying Common Criteria

(CC) to Service Oriented Architectures (SOA). In *ICCC 2012, International Common Criteria Conference, September 18-20, 2012, Paris, France,* Paris, FRANCE, 09 2012.

[96] Kaylene Hong. Dropbox reaches 300m users, 2014. Available at `http://thenextweb.com/insider/2014/05/29/dropbox-reaches-300m-users-adding-100m-users-just-six-months/`.

[97] Kerberos. The Kerberos network authentication service (v5), 2005. IETF RFC 4120.

[98] K. M. Khan and Q. Malluhi. Establishing trust in cloud computing. *IT professional*, 12(5):20–27, 2010.

[99] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee. Trustcloud: A framework for accountability and trust in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 584–588. IEEE, 2011.

[100] H. Koshutanski, A. Maña, R. Harjani, M. Montenegro, S. P. Kaluvuri, F. Di Cerbo, E. Damiani, C. A. Ardagna, M. Anisetti, D. Presenza, S. Gürgens, R. Menicocci, V. Bagini, F. Guida, and A. Riccardi. ASSERT language v2. Project Deliverable D1.2, ASSERT4SOA Consortium, 2012.

[101] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[102] C. LITTLE. Cloud computing and agile methodology – driving a revolution for developers, 2012. Available at `http://www.bmc.com/blogs/agile-in-the-cloud/`.

[103] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2(2):211–229, 1993.

[104] V. Lotz, S. P. Kaluvuri, F. Di Cerbo, and A. Sabetta. Towards security certification schemas for the internet of services. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–5. IEEE, 2012.

[105] M. A. Mahmood, F. Arslan, J. Dandu, and G. Udo. Impact of cloud computing adoption on firm stock price–an empirical research. 2014.

[106] M. Maidl, D. von Oheimb, P. Hartmann, and R. Robinson. Formal security analysis of electronic software distribution systems. In *Computer Safety, Reliability, and Security*, pages 415–428. Springer, 2008.

[107] E. Mate Bacic. The canadian trusted computer product evaluation criteria. In *Computer Security Applications Conference, 1990., Proceedings of the Sixth Annual*, pages 188–196. IEEE, 1990.

[108] Matt Brian. Apple hits 125 million iCloud users, 2012. Available at `http://thenextweb.com/apple/2012/04/24/apple-now-has-over-125-million-icloud-users/`.

[109] C. Mayerl, T. Vogel, and S. Abeck. Soa-based integration of it service management applications. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005.

[110] McAfee. Mcafee SECURE, 2007. Available at `http://www.mcafee.com/us/mcafeesecure/index.html`.

[111] B. Menkus. Us government agencies belatedly address information system security issues. *Computers & Security*, 7(4):361–366, 1988.

[112] Microsoft inc. Market. http://msdn.microsoft.com/en-us/library/gg490776.aspx. Available at `http://msdn.microsoft.com/en-us/library/gg490776.aspx`.

[113] Microsoft inc. Windows azure: Terms of use. https://datamarket.azure.com/terms. Available at `https://datamarket.azure.com/terms`.

[114] Microsoft inc. Windows marketplace. http://www.windowsphone.com/marketplace. Available at `http://www.windowsphone.com/marketplace`.

[115] Microsoft Inc. Microsoft by numbers, 2014. Available at `http://www.microsoft.com/en-us/news/bythenumbers/index.html`.

[116] M. Montenegro, A. Maña, and H. Koshutanski. Improving security assurance of services through certificate profiles. In *3rd International Workshop on Adaptive Services for the Future Internet*, 2013.

[117] T. Moores. Do consumers understand the role of privacy seals in e-commerce? *Communications of the ACM*, 48(3):86–91, 2005.

[118] K. Murray. *Microsoft Office 365: Connect and Collaborate Virtually Anywhere, Anytime*. Microsoft Press, 2011.

[119] A. Nash, W. Duane, and C. Joseph. *PKI: Implementing and Managing E-security*. McGraw-Hill, Inc., 2001.

[120] National Institute of Standards and Technology. SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES, 2001.

[121] National Institute of Standards and Technology. Special Publication 800-78-3 — Cryptographic Algorithms and Key Sizes for Personal Identification Verification (PIV), 2010.

[122] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen. A preliminary survey on software testing practices in australia. In *Software Engineering Conference, 2004. Proceedings. 2004 Australian*, pages 116–125. IEEE, 2004.

[123] F. Niessink and H. Van Vliet. Software maintenance from a service perspective. *Journal of Software Maintenance*, 12(2):103–120, 2000.

[124] J. M. Nilles. Opportunities and threats from the personal computer. *Futures*, 11(2):172–176, 1979.

[125] Nokia. Nokia ovi store content guidelines. Available at `http://support.publish.nokia.com/wp-content/uploads/2011/11/Ovi_Store_Content_Guidelines_1.3_November_25.pdf`.

[126] Nokia. Packaging and signing. http://www.developer.nokia.com/. Available at `http://www.developer.nokia.com/`.

[127] T. B. Nygaard. Common criteria design toolbox. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2007. Supervised by Professor Robin Sharp and Assoc. Professor Michael R. Hansen, IMM, DTU.

[128] OASIS. OASIS WS-Security specification, 2006. Available at `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss`.

[129] OASIS. OASIS WS-SecurityPolicy specification, 2007. Available at `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html`.

[130] OASIS. OASIS WS-Trust specification, 2007. Available at `http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html`.

[131] Object Constraint Language. ISO/IEC 19507:2012. Available at `http://www.omg.org/spec/OCL`.

[132] U. S. G. A. Office. Information assurance: National partnership offers benefits, but faces considerable challenges. Technical Report GAO 06-392, Report, March 2006.

[133] N. Olivero and P. Lunt. Privacy versus willingness to disclose in e-commerce exchanges: The effect of risk awareness on the relative role of trust and control. *Journal of Economic Psychology*, 25(2):243–262, 2004.

[134] OPTET. Optet - operation trustworthiness enabling technologies. Available at `http://www.optet.eu/`.

[135] Oracle. Jersey - restful web services in java, 2014. Available at `https://jersey.java.net/`.

[136] M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 3–12. IEEE, 2003.

[137] M. P. Papazoglou and J. Dubray. A survey of web service technologies. 2004.

[138] S. Pearson and B. Balacheff. *Trusted computing platforms: TCPA technology in context*. Prentice Hall Professional, 2003.

[139] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693–702. IEEE, 2010.

[140] C. Pedrinaci and T. Leidig. Linked-USDL core, 2012. Available at `http://linked-usdl.org/ns/usdl-core`.

[141] R. Perez, R. Sailer, L. van Doorn, et al. vtpm: virtualizing the trusted platform module. In *Proc. 15th Conf. on USENIX Security Symposium*, pages 305–320, 2006.

[142] S. Ramgovind, M. M. Eloff, and E. Smith. The management of security in cloud computing. In *Information Security for South Africa (ISSA), 2010*, pages 1–7. IEEE, 2010.

[143] B. Randell. *System structure for software fault tolerance*. Springer, 1978.

[144] K. Rihaczek. The harmonized itsec evaluation criteria. *Computers & Security*, 10(2):101–110, 1991.

[145] RIMM inc. BlackBerry app world. http://us.blackberry.com/developers/appworld/.

[146] J. Rubin and D. Chisnell. *Handbook of usability testing how to plan, design, and conduct effective tests*. Wiley Pub., Indianapolis, IN, 2008. 00000.

[147] A. Sabetta, M. Bezzi, and S. P. Kaluvuri. Towards a development environment to orchestrate services with certified security properties. In *Proceedings of the 2012 ACM SIGSOFT symposium on Industry Day*, pages 1–4. ACM, 2012.

[148] Salesforce. Security review - developer.force.com., 2012. Available at `http://wiki.developerforce.com/page/Security_Review`.

[149] SAML Specification. SAML specification, 2012. Available at `http://saml.xml.org/saml-specifications`.

[150] SAP SE. Sap - the best-run businesses run sap. Available at `http://www.sap.com/index.html`.

[151] SAP SE. Sap cloud applications. Available at `http://www.sap.com/pc/tech/cloud/software/cloud-applications/index.html`.

[152] SAP SE. Sap hana cloud platform: The in-memory platform-as-a-service. Available at `http://hcp.sap.com/index.html`.

[153] SAP SE. Sap service marketplace. Available at `https://websmp201.sap-ag.de/`.

[154] SAP SE. The Cloud Built for Businesses, 2014. Available at `http://www.sap.com/pc/tech/cloud.html`.

[155] SAS 70. SAS 70 Overview, 2014. Available at `http://sas70.com/sas70_overview.html`.

[156] Schematron. ISO/IEC 19757-3:2006. Available at http://www.schematron.com.

[157] M. J. Schwartz. 5 dropbox security warnings for businesses, 2014. Available at http://www.networkcomputing.com/networking/5-dropbox-security-warnings-for-businesses/d/d-id/1105782?

[158] S. SE. SAP Netweaver Technology Platform, 2014. Available at http://scn.sap.com/community/netweaver.

[159] A. W. Services. Amazon s3, 2014. Available at http://aws.amazon.com/s3/.

[160] S. Setty, A. J. Blumberg, and M. Walfish. Toward practical and unconditional verification of remote computations. In *Proceedings of the 13th USENIX conference on Hot topics in operating systems, HotOS*, volume 13, pages 29–29, 2011.

[161] J. Shapiro. Understanding the windows eal4 evaluation. *Computer*, 36(2):103–105, 2003.

[162] Z. Shen and Q. Tong. The security of cloud computing system enabled by trusted computing technology. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 2, pages V2–11. IEEE, 2010.

[163] R. E. Smith. Trends in security product evaluations. *Inf. Sys. Sec.*, 16(4):203–216, July 2007.

[164] SOX 404. Sarbanes-Oxley Act Section 404, 2014. Available at http://www.soxlaw.com/s404.htm.

[165] SPARQL. SPARQL query language for RDF, 2008. Available at http://www.w3.org/TR/rdf-sparql-query/.

[166] SpiderOak. Spideroak engineering matters, 2014. Accessed on 26-09-2014. Available at https://spideroak.com/engineering_matters.

[167] SPKI. SPKI certificate theory, 1999. IETF RFC 2693.

[168] S. Srinivasan. Hidden aspects of a cloud computing contract. In *Cloud Computing Basics*, pages 119–139. Springer, 2014.

[169] SSAE 16. SSAE 16 Overview, 2014. Available at `http://ssae16.com/SSAE16_overview.html`.

[170] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11, 2011.

[171] D. Tcsec. Trusted computer system evaluation criteria. *DoD 5200.28-STD*, 83, 1985.

[172] M. Turner, D. Budgen, and P. Brereton. Turning software into a service. *Computer.*, 36(10):38–44, 2003.

[173] N. C. S. C. (US). *A Guide to Procurement of Trusted Systems*, volume 3. DIANE Publishing, 1993.

[174] US Department of Commerce. U.s.-eu safe harbor overview, 2014. Available at `http://www.export.gov/safeharbor/eu/eg_main_018476.asp`.

[175] J. Voas, J. B. Michael, and M. van Genuchten. The mobile software app takeover. *Software, IEEE*, 29(4):25–27, 2012.

[176] K. Wallnau. *Software component certification: 10 useful distinctions*. Technical note. Carnegie Mellon University, Software Engineering Institute, Pittsburg, PA, USA, 2004.

[177] T. Wood, A. Gerber, K. Ramakrishnan, P. Shenoy, and J. Van der Merwe. The case for enterprise-ready virtual private clouds. *Usenix HotCloud*, 2009.

[178] X.509. The directory: Public-key and attribute certificate frameworks, 2005. ITU-T Recommendation X.509:2005 | ISO/IEC 9594-8:2005.

[179] XPath. XML path language. Available at `http://www.w3.org/TR/xpath/`.

[180] H. Yajima, M. Murata, N. Kai, and T. Yamasato. Consideration of present status and approach for the widespread of cc certification to a private field cases in japan.

[181] C. Zhou and S. Ramacciotti. Common criteria: Its limitations and advice on improvement. *Information Systems Security Association ISSA Journal*, pages 24–28, 2011.

[182] K. Zickuhr and A. Smith. Digital Differences, 2012. Available at http://www.pewinternet.org/2012/04/13/digital-differences/#internet-adoption-over-time.

Appendices

Appendix A

TitaniumBox Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ns2:Assertion ID="ID_56dcd2ac-29c6-4e04-b764-273e5297fd72"
    IssueInstant="2012-12-20T15:25:13.653+01:00" Version="2.0"
    xmlns:ns2="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"
    xmlns:ns4="http://www.w3.org/2001/04/xmlenc#" xmlns:ns5="urn:assert4soa:assert:2.0">
    <ns2:Issuer>C=us,CN=Thawte CA,O=Thawte Inc.</ns2:Issuer>
    <ns2:Subject>
        <ns2:NameID>C=us,L=San Francisco,O=Dropbox Inc.,ST=California</ns2:NameID>
    </ns2:Subject>
    <ns2:Conditions NotBefore="2012-12-20T10:18:42.722+01:00" NotOnOrAfter="2013-12-20T10:18:44.671+01:00"/>
    <ns2:Statement SerialNumber="3165321393" Version="2.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns5:ASSERTSAMLAssertionStatementType">
        <ASSERTCore>
            <TargetOfCertification
                   Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#Software-as-a-service">
            <Assets>
                <Asset ID="A_REQUEST" Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#InputParameter">
                    <Name Format="FormDataContentDisposition">requestDetail</Name>
                    <APIBinding>WIDL/upload/requestDetail</APIBinding>
                    <TOCComponents>
                        <TOCComponent TOCComponentRef="C_TITANIUMBOXAPI"/>
                    </TOCComponents>
                </Asset>
            </Assets>
            <TOCComponents>
                <TOCComponent ID="C_TITANIUMBOXAPI" InTargetOfEvaluation="true"/>
                <ComponentModel/>
            </TOCComponents>
        </TargetOfCertification>
        <SecurityProblemDefinition>
            <ProblemDefinition ID="SPD_DISCLOSURE">
                 <Description>
                    The security problem stemming from potential unauthorized access by
                    third parties to user data stored by Dropbox violating user
                    privacy and confidentiality.
                </Description>
                <Assets>
                    <Asset AssetRef="A_REQUEST"/>
                </Assets>
                <Rationale>
                    <Threats>
                        <Threat ID="T_INFDISCLOSURE"
                               Type="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#DISCLOSURE">
                          <Description>A TitaniumBox user (or a third-party user be that Amazon personnel)
                           accesses information stored by the
                           TitaniumBox service in the Amazon S3 persistent storage without the permission
                            of TitaniumBox who has responsibility for protecting the data.
                         </Description>
                         <Name>Unauthorized data access</Name>
                        </Threat>
                    </Threats>
                </Rationale>
            </ProblemDefinition>
        </SecurityProblemDefinition>
        <SecurityProperty
                PropertyAbstractCategory="http://www.assert4soa.eu/ontology/security/security#Confidentiality"
                PropertyContext="http://www.assert4soa.eu/ontology/a4s-language/a4s-language#Transit" Version="V1">
            <NameID>Confidentialy of Data in Transit</NameID>
            <Assets>
                <Asset AssetRef="A_REQUEST"/>
            </Assets>
            <SecurityObjectives>
                <SecurityObjective ID="sec_obj_user_data_protection" Type="TOE">
                    <Name>SECURE_CHANNEL</Name>
                    <SecurityProblemDefinition SecurityProblemDefinitionRef="SPD_DISCLOSURE"/>
                </SecurityObjective>
            </SecurityObjectives>
            <SecurityMechanisms>
                <SecurityMechanism Type="http://www.assert4soa.eu/ontology/usdl-sec#Cryptography">
                    <Name>AES-256</Name>
                    <Description>High-grade symmetric encryption using algorithm AES-256 with 256 bits key size.</Description>
                    <SecurityObjective SecurityObjectiveRef="sec_obj_user_data_protection"/>
                </SecurityMechanism>
            </SecurityMechanisms>
        </SecurityProperty>
        <ServiceBinding>
            <WIDLBinding>
                <ServiceName ID="service_name_userdatastorage">TitaniumBox</ServiceName>
                <ServiceAddress>http://example.org/TitaniumBox/upload</ServiceAddress>
                <PortType ID="port_type_userdatastorage" Name="UserDataStorage">
                    <OperationName ID="operation_name_uploaduserfile">upload</OperationName>
                </PortType>
                <DigestValue DigestAlgorithm="http://www.w3.org/2000/09/xmldsig#sha1">YFh8lwNzqQdfIyiwd/6cZV6mhlM=</DigestValue>
            </WIDLBinding>
        </ServiceBinding>
    </ASSERTCore>
    <ASSERTTypeSpecific>
        <ASSERT-E>
            <Description>The evaluation-specific details such as the test suites used to evaluate the security solution provided by
                TitaniumBox, and the results of the testing are presented as an outcome of the performed test-based type evaluation (ASSERT-E).</Description>
            <Property/>
            <ServiceModel/>
        </ASSERT-E>
    </ASSERTTypeSpecific>
    </ns2:Statement>
</ns2:Assertion>
```

Appendix B

ASSERT Schema

```
<?xml version="1.0" encoding="UTF-8"?>

<schema attributeFormDefault="unqualified" blockDefault="substitution"
    elementFormDefault="unqualified" targetNamespace="urn:assert4soa:assert:2.0"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:assert="urn:assert4soa:assert:2.0"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:profile="urn:assert4soa:profile:1.0">


  <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd" />
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd" />

  <element name="ASSERT" type="assert:ASSERTStandaloneType" />

  <!-- ASSERT Main Types -->
  <complexType name="ASSERTStandaloneType">
    <sequence>
      <element name="ASSERTCore" type="assert:StandaloneCoreType" />
      <element name="ASSERTTypeSpecific" type="assert:TypeSpecificType" minOccurs="0"/>
      <element name="UserDefinedExtensions" type="assert:UserDefinedExtensionsType" minOccurs="0" />
      <element name="CACerts" type="assert:CACertsType" minOccurs="0" />
      <element name="CertificateExtensions" type="assert:CertificateExtensionsType" minOccurs="0" />
      <element ref="ds:Signature" minOccurs="0"/>
    </sequence>
    <attribute name="SerialNumber" type="integer" use="required" />
    <attribute name="Version" type="assert:stringNonEmpty" use="required" fixed="3.0" />
    <attribute name="ValidNotBefore" type="dateTime" use="optional" />
    <attribute name="ValidNotOnOrAfter" type="dateTime" use="optional" />
  </complexType>

  <complexType name="UserDefinedExtensionsType">
    <sequence>
      <element maxOccurs="unbounded" name="UserDefinedExtension" type="assert:ExtensionType"/>
    </sequence>
  </complexType>

  <complexType name="CertificateExtensionsType">
    <sequence>
      <element maxOccurs="unbounded" name="CertificateExtension"
               type="assert:ExtensionType"/>
    </sequence>
  </complexType>

  <complexType name="CACertsType">
    <sequence>
      <element maxOccurs="unbounded" name="CACert" type="assert:CertType"/>
    </sequence>
  </complexType>

  <complexType name="StandaloneCoreType">
```

```
    <complexContent>
      <extension base="assert:CoreType">
        <sequence>
          <element name="ServiceProvider" type="assert:NameIDType" minOccurs="0" />
          <element name="ASSERTIssuer" type="assert:NameIDType" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <complexType name="ASSERTSAMLAssertionStatementType">
    <complexContent>
      <extension base="saml:StatementAbstractType">
        <sequence>
          <element name="ASSERTCore" type="assert:CoreType" />
          <element name="ASSERTTypeSpecific" type="assert:TypeSpecificType" minOccurs="0"/>
          <element name="UserDefinedExtensions" type="assert:UserDefinedExtensionsType" minOccurs="0" />
          <element name="CACerts" type="assert:CACertsType" minOccurs="0" />
          <element name="CertificateExtensions" type="assert:CertificateExtensionsType" minOccurs="0" />
        </sequence>
        <attribute name="SerialNumber" type="integer" use="required" />
        <attribute name="Version" type="assert:stringNonEmpty" use="required" fixed="3.0"/>
      </extension>
    </complexContent>
  </complexType>

  <complexType name="CertType">
    <sequence>
      <element name="Type" type="assert:CertTypeType" />
      <element name="Data" type="base64Binary" />
      <element name="Description" type="assert:stringNonEmpty" minOccurs="0"/>
    </sequence>
    <attribute name="UserImported" type="boolean" use="optional" default="true" />
  </complexType>

  <simpleType name="CertTypeType">
    <restriction base="string">
      <enumeration value="X.509" />
      <enumeration value="SAML" />
    </restriction>
  </simpleType>

  <!-- Core Language -->
  <complexType name="CoreType">
    <sequence>
      <element name="TargetOfCertification" type="assert:TargetOfCertificationType" />
      <element name="SecurityProblemDefinition" type="assert:SecurityProblemDefinitionType" minOccurs="0" />
      <element name="SecurityProperty" type="assert:SecurityPropertyType" />
      <element name="CertificationProcess" type="assert:CertificationProcessType" minOccurs="0" />
      <element name="ServiceBinding" type="assert:ServiceBindingBindingMechanismType" />
      <element name="ASSERT4Humans" type="assert:ASSERT4HumansType" minOccurs="0" />
      <element name="ASSERTSigner" type="assert:NameIDType" minOccurs="0" />
      <element name="ProfileConformance" type="assert:ProfileConformanceType" minOccurs="0" />
      <element name="ASSERTIssuerCompetence" type="assert:ASSERTIssuerCompetenceType" minOccurs="0" />
    </sequence>
  </complexType>

  <complexType name="ProfileConformanceType">
    <sequence>
      <element name="Profile" maxOccurs="unbounded" type="assert:ProfileRefType"/>
    </sequence>
  </complexType>

  <complexType name="ProfileRefType">
    <simpleContent>
      <extension base="string">
        <attribute name="ProfileURI" type="assert:anyURINonEmpty" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

  <complexType name="ASSERTIssuerCompetenceType">
    <sequence>
      <element name="CompetenceToken" maxOccurs="unbounded" type="assert:CertType"/>
    </sequence>
  </complexType>

  <simpleType name="anyURINonEmpty">
    <restriction base="anyURI">
      <minLength value="1"/>
    </restriction>
  </simpleType>

```

```xml
<simpleType name="stringNonEmpty">
  <restriction base="string">
    <minLength value="1"/>
  </restriction>
</simpleType>

<complexType name="SecurityPropertyType">
  <sequence>
    <element name="NameID" type="assert:NameIDType" />
    <element name="Assets" minOccurs="0" type="assert:SPAssetsType"/>
    <element name="SecurityObjectives" minOccurs="0" type="assert:SecurityObjectivesType"/>
    <element name="SecurityMechanisms" minOccurs="0" type="assert:SecurityMechanismsType"/>
    <element name="Description" type="assert:stringNonEmpty" minOccurs="0" />
  </sequence>
  <attribute name="Version" type="assert:stringNonEmpty" use="optional" />
  <attribute name="PropertyAbstractCategory" type="assert:anyURINonEmpty" use="required"/>
  <attribute name="PropertyContext" type="assert:anyURINonEmpty" use="optional"/>
</complexType>

<complexType name="SPAssetsType">
  <sequence>
    <element name="Asset" maxOccurs="unbounded" type="assert:SPAssetType"/>
  </sequence>
</complexType>

<complexType name="SecurityObjectivesType">
  <sequence>
    <element name="SecurityObjective" maxOccurs="unbounded" type="assert:SecurityObjectiveType"/>
  </sequence>
</complexType>

<complexType name="SecurityMechanismsType">
  <sequence>
    <element name="SecurityMechanism" maxOccurs="unbounded" type="assert:SecurityMechanismType"/>
  </sequence>
</complexType>

<complexType name="SPAssetType">
  <attribute name="AssetRef" type="IDREF" use="required"/>
</complexType>

<complexType name="SecurityObjectiveType">
  <sequence>
    <element name="Name" type="assert:NameIDType" />
    <element name="Description" minOccurs="0" type="assert:stringNonEmpty" />
    <element name="SecurityProblemDefinition" minOccurs="0" maxOccurs="unbounded"
          type="assert:SOSecurityProblemDefinitionType" />
  </sequence>
  <attribute name="ID" type="ID" use="optional"/>
  <attribute name="Type" type="assert:anyURINonEmpty" use="optional"/>
  <attribute name="Scope" type="assert:anyURINonEmpty" use="optional"/>
</complexType>

<complexType name="SOSecurityProblemDefinitionType">
  <attribute name="SecurityProblemDefinitionRef" type="IDREF" use="required"/>
</complexType>

<complexType name="SecurityMechanismType">
  <sequence>
    <element name="Name" type="assert:NameIDType" />
    <element name="Value" minOccurs="0" type="assert:NameIDType" />
    <element name="Description" minOccurs="0" type="assert:stringNonEmpty" />
    <element name="SecurityObjective" minOccurs="0" maxOccurs="unbounded" type="assert:SMSecurityObjectiveType" />
    <element name="SecurityFunctionalCriteria" minOccurs="0" maxOccurs="unbounded"
          type="assert:SecurityFunctionalCriteriaType" />
  </sequence>
  <attribute name="ID" type="ID" use="optional"/>
  <attribute name="Type" type="assert:anyURINonEmpty" use="optional"/>
</complexType>

<complexType name="SecurityFunctionalCriteriaType">
  <simpleContent>
    <extension base="assert:stringNonEmpty">
      <attribute name="Scheme" type="assert:anyURINonEmpty" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="SMSecurityObjectiveType">
  <attribute name="SecurityObjectiveRef" type="IDREF" use="required"/>
</complexType>

<complexType name="ASSERT4HumansType">
```

```
    <sequence>
      <element name="Description" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="Keywords" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="OriginalDocument" type="assert:anyURINonEmpty" />
    </sequence>
  </complexType>

  <complexType name="TypeSpecificType">
    <sequence>
      <choice>
        <element name="ASSERT-E" type="assert:E_ASSERTType" />
        <element name="ASSERT-M" type="assert:M_ASSERTType" />
      </choice>
    </sequence>
  </complexType>

  <complexType name="CertificationCriteriaType">
    <sequence>
      <element name="NameID" type="assert:NameIDType" />
      <element minOccurs="0" name="Version" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="Authority" type="assert:NameIDType" />
    </sequence>
  </complexType>

  <complexType name="TOCComponentsRefType">
    <sequence>
      <element name="TOCComponent" maxOccurs="unbounded" type="assert:TOCComponentRefType"/>
    </sequence>
  </complexType>

  <complexType name="TOCComponentRefType">
    <attribute name="TOCComponentRef" type="IDREF" use="required"/>
  </complexType>

  <complexType name="TOCComponentType">
    <sequence>
      <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="ComponentModel" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="TechnicalModel" type="assert:TOCTechnicalModelType" />
    </sequence>
    <attribute name="ID" type="ID" use="optional"/>
    <attribute name="InTargetOfEvaluation" type="boolean" use="optional" default="true"/>
    <attribute name="ComponentServiceRef" type="IDREF" use="optional"/>
  </complexType>

  <complexType name="TOCAssetsType">
    <sequence>
      <element name="Asset" maxOccurs="unbounded" type="assert:AssetType"/>
    </sequence>
  </complexType>

  <complexType name="AssetType">
    <sequence>
      <element name="Name" type="assert:NameIDType" />
      <element name="Description" minOccurs="0" type="assert:stringNonEmpty" />
      <element name="APIBinding" minOccurs="0" type="assert:stringNonEmpty" />
      <element name="TOCComponents" type="assert:TOCComponentsRefType" />
    </sequence>
    <attribute name="Type" type="assert:anyURINonEmpty" use="required"/>
    <attribute name="ID" type="ID" use="optional"/>
  </complexType>

  <complexType name="TOCTechnicalModelType">
    <sequence>
      <element name="Description" type="assert:stringNonEmpty" />
      <element name="Configurations" type="assert:TOCConfigurationsType" minOccurs="0"/>
    </sequence>
  </complexType>

  <complexType name="TOCConfigurationsType">
    <sequence>
      <element name="Configuration" maxOccurs="unbounded" type="assert:TOCConfigurationType"/>
    </sequence>
  </complexType>

  <complexType name="TOCConfigurationType">
    <simpleContent>
      <extension base="assert:stringNonEmpty">
        <attribute name="Name" type="assert:stringNonEmpty" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

```

```
<complexType name="TargetOfCertificationType">
  <sequence>
    <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
    <element minOccurs="0" name="Assets" type="assert:TOCAssetsType" />
    <element name="TOCComponents" type="assert:TOCComponentsType" />
    <element minOccurs="0" name="DeploymentAndImplementationModel" type="assert:DeploymentAndImplementationModelType" />
  </sequence>
  <attribute name="Type" type="assert:anyURINonEmpty" use="required" />
</complexType>

<complexType name="TOCComponentsType">
  <sequence>
    <element name="TOCComponent" maxOccurs="unbounded" type="assert:TOCComponentType"/>
  </sequence>
</complexType>

<complexType name="DeploymentAndImplementationModelType">
  <sequence>
    <element name="Description" type="assert:stringNonEmpty"/>
  </sequence>
</complexType>

<complexType name="SecurityProblemDefinitionType">
  <sequence>
    <element name="ProblemDefinition" maxOccurs="unbounded" type="assert:SPDDefinitionType"/>
    <element minOccurs="0" name="ScopeSpecification" type="assert:ScopeSpecificationType"/>
  </sequence>
</complexType>

<complexType name="ScopeSpecificationType">
  <sequence>
    <element maxOccurs="unbounded" name="Scope" type="assert:ScopeType"/>
  </sequence>
</complexType>

<complexType name="SPDDefinitionType">
  <sequence>
    <element name="Description" type="assert:stringNonEmpty" minOccurs="0"/>
    <element name="Assets" type="assert:SPDAssetsType"/>
    <element name="Rationale" type="assert:SPDRationaleType" />
  </sequence>
  <attribute name="ID" type="ID" use="optional"/>
</complexType>

<complexType name="SPDAssetsType">
  <sequence>
    <element maxOccurs="unbounded" name="Asset" type="assert:SPDAssetType"/>
  </sequence>
</complexType>

<complexType name="SPDAssetType">
  <attribute name="AssetRef" type="IDREF" use="required"/>
</complexType>

<complexType name="SPDRationaleType">
  <sequence>
    <element minOccurs="0" name="Description" type="assert:stringNonEmpty"/>
    <element minOccurs="0" name="Threats" type="assert:ThreatsType"/>
    <element minOccurs="0" name="SecurityPolicies" type="assert:SecurityPoliciesType"/>
  </sequence>
</complexType>

<complexType name="ThreatsType">
  <sequence>
    <element maxOccurs="unbounded" name="Threat" type="assert:ThreatType"/>
  </sequence>
</complexType>

<complexType name="SecurityPoliciesType">
  <sequence>
    <element maxOccurs="unbounded" name="SecurityPolicy" type="assert:SecurityPolicyType"/>
  </sequence>
</complexType>

<complexType name="ScopeType">
  <sequence>
    <element name="Description" minOccurs="0" type="assert:stringNonEmpty" />
    <element name="Name" type="assert:NameIDType" />
    <element name="Value" minOccurs="0" type="assert:NameIDType" />
  </sequence>
  <attribute name="Type" type="assert:anyURINonEmpty" use="optional"/>
  <attribute name="ID" type="ID" use="optional"/>
</complexType>
```

```

  <complexType name="ThreatType">
    <sequence>
      <element name="Description" minOccurs="0" type="assert:stringNonEmpty" />
      <element name="Name" type="assert:NameIDType" />
      <element name="Value" minOccurs="0" type="assert:NameIDType" />
    </sequence>
    <attribute name="Type" type="assert:anyURINonEmpty" use="optional"/>
    <attribute name="ID" type="ID" use="optional"/>
  </complexType>

  <complexType name="SecurityPolicyType">
    <sequence>
      <element name="Description" minOccurs="0" type="assert:stringNonEmpty" />
      <element name="Name" type="assert:NameIDType" />
      <element name="Value" minOccurs="0" type="assert:NameIDType" />
    </sequence>
    <attribute name="Type" type="assert:anyURINonEmpty" use="optional"/>
    <attribute name="ID" type="ID" use="optional"/>
  </complexType>

  <complexType name="CertificationProcessType">
    <sequence>
      <element minOccurs="0" name="CertificationCriteria" type="assert:CertificationCriteriaType" />
      <element minOccurs="0" name="PerformedBy" type="assert:NameIDType" />
      <element minOccurs="0" name="EvaluationDate" type="assert:StartEndDateTime" />
      <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
    </sequence>
  </complexType>
 <!-- ASSERT-E Type -->
  <complexType name="E_ASSERTType">
    <sequence>
      <element name="Description" type="assert:stringNonEmpty" minOccurs="0" />
      <element name="Property" type="assert:E_PropertyType" />
      <element name="ServiceModel" type="assert:E_ServiceModelType" />
      <element maxOccurs="unbounded" minOccurs="0" name="TestEvidence"
                type="assert:E_TestEvidenceType"/>
      <element name="TestMetrics" type="assert:E_TestMetricsType" />
      <element name="ElementForExtension" type="assert:E_ElementForExtensionType" minOccurs="0"/>
    </sequence>
  </complexType>

  <complexType name="E_PairType">
    <sequence>
      <element name="Name" type="assert:stringNonEmpty" />
      <element name="Value" type="assert:stringNonEmpty" />
    </sequence>
  </complexType>

  <complexType name="E_PropertyType">
    <sequence>
      <element name="PropertyName" type="assert:anyURINonEmpty" />
      <element name="ClassAttribute" type="assert:E_PairType"
                maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="E_ServiceModelType">
    <sequence>
      <element name="ServiceModelLink" type="assert:anyURINonEmpty" minOccurs="0"/>
      <element name="ModelType" type="assert:WSModelType" />
      <element maxOccurs="unbounded" name="Index" type="assert:E_PairType"/>
    </sequence>
  </complexType>

  <complexType name="E_TestEvidenceType">
    <sequence>
      <element name="TestCategory" type="assert:stringNonEmpty" />
      <element name="TestType" type="assert:stringNonEmpty" />
      <element name="TestGenerationModelLink" type="assert:anyURINonEmpty" />
      <element maxOccurs="unbounded" name="TestAttribute"
                type="assert:E_PairType"/>
      <element maxOccurs="unbounded" minOccurs="0" name="TestCase"
                type="assert:E_TestCaseType"/>
    </sequence>
  </complexType>

  <complexType name="E_TestCaseType">
    <sequence>
      <element name="ID" type="assert:stringNonEmpty" />
      <element name="Description" type="assert:stringNonEmpty" />
      <element maxOccurs="unbounded" name="TestInstance"
                type="assert:E_TestInstanceType"/>
  </sequence>
</complexType>

<complexType name="E_TestInstanceType">
  <sequence>
    <element name="Preconditions" type="assert:stringNonEmpty" />
    <element name="HiddenCommunications" type="assert:stringNonEmpty" />
    <element name="Input" type="assert:stringNonEmpty" minOccurs="0"/>
    <element name="ExpectedOutput" type="assert:stringNonEmpty" minOccurs="0"/>
    <element name="PostConditions" type="assert:stringNonEmpty" />
  </sequence>
  <attribute name="Operation" type="assert:stringNonEmpty" />
</complexType>

<complexType name="E_TestMetricsType">
  <sequence>
    <element name="OperationCoverage" type="assert:stringNonEmpty" />
    <element name="InputPartitionCovarage" type="assert:stringNonEmpty" />
    <element name="BranchCoverage" type="assert:stringNonEmpty" />
    <element name="ConditionCoverage" type="assert:stringNonEmpty" />
    <element name="PathCoverage" type="assert:stringNonEmpty" />
    <element name="AttackCoverage" type="assert:stringNonEmpty" />
    <element name="Others" type="assert:stringNonEmpty" minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="E_ElementForExtensionType">
  <sequence>
    <element name="Environment" type="assert:stringNonEmpty" />
    <element name="TestingTool" type="assert:stringNonEmpty" />
    <element name="Code" type="assert:stringNonEmpty" minOccurs="0"/>
    <element name="Others" type="assert:stringNonEmpty" minOccurs="0"/>
  </sequence>
</complexType>


<!-- ASSERT-M Type -->
<complexType name="M_ASSERTType">
  <sequence>
    <element name="ServiceModel" type="assert:M_ServiceModelType" />
    <element maxOccurs="unbounded" name="PropertyFormalSpecification" type="assert:M_PropertyFormalSpecificationType" />
  </sequence>
</complexType>

<complexType name="M_PropertyFormalSpecificationType">
  <sequence>
    <choice>
      <element name="FormalSpecification" type="assert:M_FormalSpecificationAbstractType" />
      <element name="FormalSpecificationRef" type="assert:anyURINonEmpty" />
    </choice>
    <element name="Evaluation" type="assert:M_EvaluationType" minOccurs="0"/>
  </sequence>
  <attribute name="ID" type="ID" use="optional"/>
</complexType>

<complexType name="M_FormalSpecificationAbstractType" abstract="true"/>

<complexType name="M_xSeMFFormalSpecificationType">
  <complexContent>
    <extension base="assert:M_FormalSpecificationAbstractType">
      <sequence>
        <element name="Data" type="assert:stringNonEmpty" />
      </sequence>
      <attribute name="Type" type="assert:stringNonEmpty" use="required" fixed="xSeMF"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="M_GenericFormalSpecificationType">
  <complexContent>
    <extension base="assert:M_FormalSpecificationAbstractType">
      <sequence>
        <element name="Data" type="assert:stringNonEmpty" />
      </sequence>
      <attribute name="Type" type="assert:stringNonEmpty" use="optional"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="M_EvaluationType">
  <sequence>
    <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
    <element name="Assumptions" type="assert:M_AssumptionsType"/>
```

```
      <element minOccurs="0" name="ProofRef" type="assert:anyURINonEmpty" />
    </sequence>
  </complexType>

  <complexType name="M_AssumptionsType">
    <sequence>
      <element maxOccurs="unbounded" name="Assumption"
               type="assert:M_AssumptionType"/>
    </sequence>
  </complexType>

  <complexType name="M_AssumptionType">
    <sequence>
      <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
      <element name="AssumptionModel" type="assert:stringNonEmpty"/>
      <element name="FormalSpecification" type="assert:stringNonEmpty" />
    </sequence>
  </complexType>

  <complexType name="M_ServiceModelType">
    <sequence>
      <element minOccurs="0" name="Name" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
      <choice minOccurs="0">
        <element name="FormalDescription" type="assert:M_ServiceModelDescriptionAbstractType" />
        <element name="FormalDescriptionLink" type="assert:anyURINonEmpty" />
      </choice>
    </sequence>
    <attribute name="Type" type="assert:stringNonEmpty" use="required" fixed="WSDL"/>
  </complexType>

  <complexType name="M_ServiceModelDescriptionAbstractType" abstract="true"/>

  <complexType name="M_xSeMFServiceModelDescriptionType">
    <complexContent>
      <extension base="assert:M_ServiceModelDescriptionAbstractType">
        <sequence>
          <element name="ModelExtension" type="assert:stringNonEmpty" />
        </sequence>
        <attribute name="Type" type="assert:stringNonEmpty" use="required" fixed="xSeMF"/>
      </extension>
    </complexContent>
  </complexType>

  <complexType name="M_GenericServiceModelDescriptionType">
    <complexContent>
      <extension base="assert:M_ServiceModelDescriptionAbstractType">
        <sequence>
          <element name="Data" type="assert:stringNonEmpty" />
        </sequence>
        <attribute name="Type" type="assert:stringNonEmpty" use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <simpleType name="WSModelType">
    <restriction base="string">
      <enumeration value="WSDL" />
      <enumeration value="WSCL" />
      <enumeration value="Implementation" />
      <enumeration value="Other" />
    </restriction>
  </simpleType>

  <complexType name="ExtensionType">
    <sequence>
      <element name="Name" type="assert:NameIDType" />
      <element minOccurs="0" name="Value" type="assert:stringNonEmpty" />
      <element minOccurs="0" name="Description" type="assert:stringNonEmpty" />
    </sequence>
    <attribute name="ID" type="ID" use="optional"/>
    <attribute name="Critical" type="boolean" default="false" use="optional"/>
  </complexType>

  <complexType name="OpenExtensionType">
    <sequence>
      <any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
    </sequence>
  </complexType>

  <complexType name="StartEndDateTime">
    <sequence>
      <element name="Start" type="dateTime" minOccurs="0"/>
    <element name="End" type="dateTime" minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="NameIDType">
  <simpleContent>
    <extension base="assert:stringNonEmpty">
      <attribute name="NameQualifier" type="assert:stringNonEmpty" use="optional"/>
      <attribute name="Format" type="assert:anyURINonEmpty" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="ServiceBindingBindingMechanismType">
  <sequence>
    <element name="WSDLBinding" type="assert:WSDLBindingType" />
  </sequence>
</complexType>

<complexType name="WSDLBindingType">
  <sequence>
    <element name="ServiceName" type="assert:ServiceNameType" />
    <element name="ServiceAddress" type="assert:anyURINonEmpty" />
    <element name="PortType" type="assert:WSDLPortType" minOccurs="0"/>
    <element name="DigestValue" type="assert:DigestValueType" minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="ServiceNameType">
  <simpleContent>
    <extension base="assert:stringNonEmpty">
      <attribute name="ID" type="ID" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="DigestValueType">
  <simpleContent>
    <extension base="base64Binary">
      <attribute name="DigestAlgorithm" type="assert:anyURINonEmpty" use="optional"
            default="http://www.w3.org/2000/09/xmldsig#sha1"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="WSDLPortType">
  <sequence>
    <element name="OperationName" type="assert:WSDLOperationNameType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Name" type="assert:stringNonEmpty" use="required" />
  <attribute name="ID" type="ID" use="optional" />
</complexType>

<complexType name="WSDLOperationNameType">
  <simpleContent>
    <extension base="assert:stringNonEmpty">
      <attribute name="ID" type="ID" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<!-- ASSERT PROFILE DATA TYPES START HERE -->

<element name="ASSERTProfile" type="assert:ASSERTProfileType" />

<complexType name="ASSERTProfileType">
  <sequence>
    <element name="ASSERTProfileIssuer" type="assert:NameIDType"
           minOccurs="0" />
    <element name="ValidityConditions" type="assert:ValidityConditionsType"
           minOccurs="0" />
    <element name="ASSERTTemplate" type="assert:ASSERTTemplateType"
           minOccurs="0" />
    <element name="SemanticRules" type="assert:SemanticRulesType"
           minOccurs="0" />
    <element name="ASSERTVocabulary" type="assert:ASSERTVocabularyType"
           minOccurs="0" />
    <element name="CACerts" type="assert:CACertsType" minOccurs="0" />
    <element ref="ds:Signature" minOccurs="0" />
  </sequence>
</complexType>

<complexType name="ASSERTVocabularyType">
```

```
    <sequence>
      <element name="Vocabulary" type="assert:VocabularyType"
                 maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="VocabularyType">
    <sequence>
      <element name="ASSERTElement" type="assert:ASSERTElementType"/>
      <choice>
        <element name="Enum" type="assert:EnumType" />
        <element name="IntRange" type="assert:IntRangeType" />
        <element name="DoubleRange" type="assert:DoubleRangeType" />
        <element name="DateRange" type="assert:DateRangeType" />
      </choice>
    </sequence>
  </complexType>

  <complexType name="ASSERTElementType">
    <simpleContent>
      <extension base="assert:stringNonEmpty">
        <attribute name="Type" type="assert:stringNonEmpty" use="required"
                   fixed="XPath" />
      </extension>
    </simpleContent>
  </complexType>

  <complexType name="EnumType">
    <sequence>
      <choice>
        <element name="StaticValues" type="assert:StaticValuesType" />
        <element name="DynamicValues" type="assert:DynamicValuesType" />
      </choice>
    </sequence>
    <attribute name="Mandatory" type="boolean" use="optional" default="true"/>
  </complexType>

  <complexType name="IntRangeType">
    <sequence>
      <element name="From" type="assert:IntRangeValueType" minOccurs="0" />
      <element name="To" type="assert:IntRangeValueType" minOccurs="0" />
    </sequence>
  </complexType>

  <complexType name="IntRangeValueType">
    <sequence>
      <choice>
        <element name="StaticValue" type="integer"/>
        <element name="DynamicValue" type="assert:DynamicValuesType"/>
      </choice>
    </sequence>
    <attribute name="Type" type="assert:rangeType" use="required"/>
  </complexType>

  <complexType name="DoubleRangeType">
    <sequence>
      <element name="From" type="assert:DoubleRangeValueType" minOccurs="0" />
      <element name="To" type="assert:DoubleRangeValueType" minOccurs="0" />
    </sequence>
  </complexType>

  <complexType name="DoubleRangeValueType">
    <sequence>
      <choice>
        <element name="StaticValue" type="double"/>
        <element name="DynamicValue" type="assert:DynamicValuesType"/>
      </choice>
    </sequence>
    <attribute name="Type" type="assert:rangeType" use="required"/>
  </complexType>

  <complexType name="DateRangeType">
    <sequence>
      <element name="From" type="assert:DateRangeValueType" minOccurs="0" />
      <element name="To" type="assert:DateRangeValueType" minOccurs="0" />
    </sequence>
  </complexType>

  <complexType name="DateRangeValueType">
    <sequence>
      <choice>
        <element name="StaticValue" type="dateTime"/>
        <element name="DynamicValue" type="assert:DynamicValuesType"/>
      </choice>
    </sequence>
    <attribute name="Type" type="assert:rangeType" use="required"/>
  </complexType>

  <simpleType name="rangeType">
    <restriction base="xs:string">
      <enumeration value="inclusive" />
      <enumeration value="exclusive" />
    </restriction>
  </simpleType>

  <complexType name="DynamicValuesType">
    <simpleContent>
      <extension base="assert:stringNonEmpty">
        <attribute name="OntologyURI" type="assert:anyURINonEmpty"
                      use="required" />
        <attribute name="QueryType" type="assert:stringNonEmpty"
                      use="required" fixed="SPARQL" />
        <attribute name="OntologySyntax" type="assert:OntologySyntaxType"
                      use="required" />
      </extension>
    </simpleContent>
  </complexType>

  <simpleType name="OntologySyntaxType">
    <restriction base="string">
      <enumeration value="RDF/XML" />
      <enumeration value="OWL/XML" />
    </restriction>
  </simpleType>

  <complexType name="StaticValuesType">
    <sequence>
      <element name="StaticValue" type="string" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="ASSERTTemplateType">
    <sequence>
      <element name="ASSERT" type="assert:ASSERTStandaloneType" />
    </sequence>
  </complexType>

  <complexType name="ValidityConditionsType">
    <attribute name="ValidNotBefore" type="dateTime" use="optional" />
    <attribute name="ValidNotOnOrAfter" type="dateTime" use="optional" />
  </complexType>

  <complexType name="SemanticRulesType">
    <sequence>
      <any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
    </sequence>
  </complexType>
</schema>
```

Appendix C

USDL-SEC

USDL-SEC [58] is a specification that can be used to fulfil the need to have an explicit representation of the security features of a service. It uses Linked Data (LD) semantic web principles and technologies [29] to interconnect security information with other descriptions and contents.

USDL-SEC acts as *trait d'union* with business and functional information, thus contributing to the constitution of a comprehensive description of a service. It provides an overview of a service's security features, that can in-turn refer to a more exhaustive and fine-grained description.

In this way, processes like service discovery and provisioning can make use of service's multi-faceted descriptions (comprising functional, business and security aspects), to better assist users in their decision making process.

LD consists in a set of recommendations, maintained by the World Wide Web Consortium[1], for making data published on the Web "machine-readable, its meaning explicitly defined, linkable with other external data sets, and that can in turn be linked to from external data sets" [29]. LD enables the semantic interconnection among different data sources, so that pieces of information organized and presented on a web site can be discovered and interpreted by another web site; the result is the integration of the two knowledge bases, and this process can be reiterated with other LD-enabled resources, to create new information exchanges and thus new knowledge.

Through its LD design, USDL-SEC enables an easy integration of its security descriptions with other LD-described data and in LD-enabled applications. USDL-SEC aims primarily at extending the more business oriented service description standard called USDL [140], although it is not limited to this extent, as it could support also other service description languages.

[1]http://w3.org

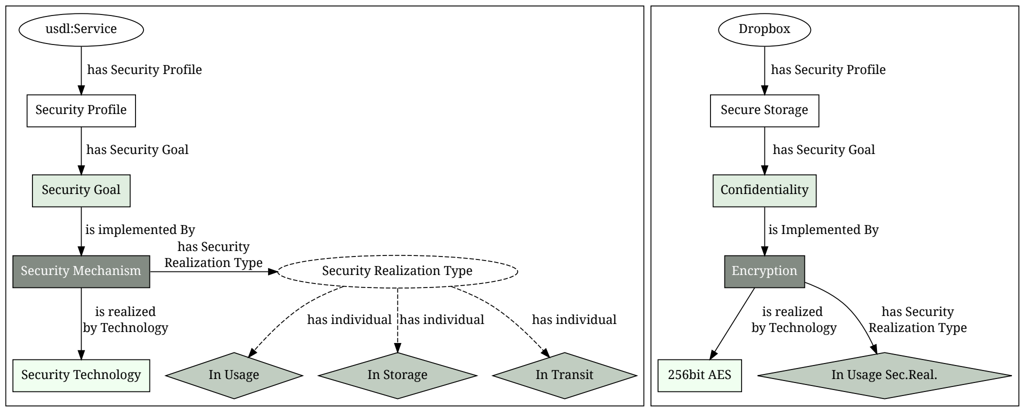


Figure C.1: Abstract Representation and an Example of USDL-SEC Description

USDL covers the main concepts and relationships characterising services, through a family of specifications: USDL-Core, USDL-SLA, USDL-Pricing. In particular, USDL-Core permits to indicate in the same structure a service's functional aspects (e.g. URL, WSDL and so on) as well as its business information (service provider, pricing policy, service-level agreements and so on).

USDL-SEC extends USDL by integrating a service's USDL description with an LD indication of its security features, that can be detailed by its digital security certificate information. LD applications can make use of this interconnected knowledge, as in the mentioned example of service discovery processes. USDL-SEC can be used by service providers to describe the security features of their offerings; while consumers then can use these descriptions through specific engines, finding among multiple alternatives the ones that fulfil their requirements and expectations.

The USDL-SEC model offers different abstraction layers to describe security features and functionalities: The higher abstraction layer offers a very high level description with no reference to any technical aspect, the lowest one enters into the configuration details of the security technologies. Figure C.1 represents this hierarchical model described below:

- Service: This element represents the USDL description of a standard service and represent the entry point to the USDL-SEC description

- Security Profile: is the root node of a USDL-SEC description of a service. This node does not contain yet any concrete information of the security feature, but just the given name of the security feature. This name is not necessarily semantically related to any security vocabulary.

- Security Goal: the highest abstraction layer referring to a security topic. This element describes the basic security concepts like anonymity, confidentiality, privacy, etc. These concepts are not yet materialized into real solutions at this level. A taxonomy for goals was defined for USDL-SEC,inspired from a previous work [81] and extended with new goals that were not defined. The Security Goals taxonomy is listed in Table C.1; it is to be noticed, however, that it can be extended at will using LD semantic relations (e.g., inheritance,equivalence and so on).

- Security Mechanism: is a set of security solutions that can achieve a security goal (a security goal can be achieved by more than one mechanism). These mechanisms are theoretical solutions that answer to specific security requirements like access control, cryptography, certification, etc. For example the confidentiality goal can be achieved by the encryption mechanism. These solutions can be applied under three realization levels: in transit (e.g. at the network level), in usage (during service computation activities), and in storage. An extensible taxonomy of Security Mechanisms is included in Table C.2.

- Security Technology: the lower abstraction layer of the model, it refers to the real implementation of the security mechanisms. Many implementations can concern the same security mechanism like for example RSA, AES are solutions to implement encryption. For this element we do not propose an ontology due to the huge number of existing security implementations, it would be impossible to catalog them and maintain the list. For this reason we kept this element open for self-edition. Beside the reference to the implementation trade name, this element contains additional parameters related to the configuration details of the technology, like for example the key length for the RSA.

In order to explain how the USDL-SEC model can be applied to a real service, consider the example of the secure storage functionality in the online file storage service DropBox. In its security overview page [61], the DropBox team claims to encrypt user information before storing them onto a third party infrastructure, Amazon S3. A simplified USDL-SEC description of this functionality is rendered in the right side of Fig. C.1, and can be described as follows:

Accountability	Anonymity	Authentication
Authenticity	Authorization	Availability
Confidentiality	Correctness	Identification
Integrity	Non-Repudiation	Policy Compliance
Privacy	Trust	...

Table C.1: USDL-SEC Security Goals Taxonomy

Access Control	Auditing	Biometric Data
Certificate	Certificate Exchange	Certification
Challenge-Response	Checksum	Control Code
Cryptography	Delegation	Digest
Digital Signature	Filtering	Key Management
Load Balancing	Logging	Monitoring
Obfuscation	Obligation	Password Exchange
Pseudonym	Recommendation	Replication
Reputation	Shared Secret	Steganography
Token	Usage Control	...

Table C.2: USDL-SEC Security Mechanisms Taxonomy

- Service: DropBox
- Security Profile: Secure storage
- Security Goal: Confidentiality
- Security Mechanism: Encryption realized at the service level
- Security Technology: 256-bit AES

This example provides a concise overview of the security functionality. However, it is worth noting that this is a single element in a USDL description; more security functionalities can be described with USDL-SEC at the same time, and they can reference to a more detailed security description. In this way, service discovery operations can make use of USDL/USDL-SEC information, while in-depth security analysis would find their source of information in the referenced security description.

Résumé

Les modèles de fourniture de logiciel ont subi un changement de paradigme durant cette dernière décennie - le logiciel est aujourd'hui diffusé en tant que Service. Les services protègent les consommateurs de la complexité de procuration, installation, configuration et maintenance des systèmes logiciels complèxes pour leurs infrastructures matérielles - essentiellement en offrant des solutions logicielles hors site et à la demande. Ces dernières années, l'adoption des services a progressé rapidement, poussée principalement par la croissance explosive des appareils mobiles qui fonctionnent souvent à base de services. Les solutions basées services (SBS) tels que Gmail [77], DropBox [96], OneDrive [115], et iCloud [108], qui ont des millions des utilisateurs, illustrent la popularité des services aujourd'hui.

Même s'il y a des nombreuses définitions des services [4], nous considérons la suivante largement acceptée : un service logiciel (ou simplement service) est une unité fonctionnelle qui peut être accedée d'une manière programmée sur un réseau.

Les fournisseurs logiciel offrent de plus en plus des applications basées services (ABS), qui sont par essence des clients légers côté consommateur qui ap-

pellent un logiciel, encapsulé sous forme d'un service, tournant sur un serveur à distance et affichent le résultat à l'utilisateur [136]. Les ABS externalisent le traitement vers un serveur à distance et se concentrent sur la présentation de l'information à ses consommateurs. Des exemples bien connus de telles ABS sont les navigateurs web et les applications mobiles.

Il y a plusieurs raisons pour la croissance et l'adoption rapide des ABS. Une cause importante, même si assez évidente, est que l'adoption de l'internet s'est rapidement accrue ces 10 dernières années. Mais la raison principale qui a donné une énorme impulsion à l'adoption des APS a été la production des smartphones et tablettes et leur croissance exponentielle [74]. Les smartphones et les tablettes n'ont pas de grande puissance de calcul par rapport à un ordinateur de bureau ou portable parce qu'ils tournent sur des processeurs de puissance plus faible et avec une mémoire réduite. Ils établissent une relation symbiotique avec les services déployés sur Internet, en externalisant le traitement et en se concentrant sur la seule présentation de l'information aux utilisateurs [175]. Comme ces derniers sont arrivés à posséder plusieurs dispositifs en plus d'ordinateurs de bureau et portables, la nécessité de synchroniser les données sur ces dispositifs est une autre raison importante pour les utilisateurs de transférer leurs données vers des services ou ce qui est plus souvent nommé "le cloud".

L'orientation services a emergé comme un catalyseur d'une série d'offres de calcul Cloud, telles que le Logiciel en tant que Service (SaaS) [172], la Plateforme en tant que Service (PaaS) [22] et l'Infrastructure en tant que Service (IaaS) [26]. Un trait commun - intrinsèque à la nature des systèmes orientés services et commun à tous les paradigmes mentionnés avant - est que le déploiement du logiciel implique de nombreuses organisations, et au-

cune entité unique ne contrôle un système entier. Les solutions basées services offrent des nombreux avantages tant pour les consommateurs que pour les fournisseurs.

Du point de vue du consommateur, l'avantage principal est que les services sont économiques puisqu'ils n'a pas besoin de se procurer et de déployer le logiciel sur son matériel, ce qui permet des économies considérables. En plus, les services sont typiquement basés sur des abonnements offrant la possibilité de s'ajuster selon la demande. Ainsi, les consommateurs peuvent choisir de payer juste pour ce dont ils ont besoin qui permet de nouveau des économies. Il y a aussi moins de problèmes de maintenance parce que les consommateurs n'ont pas besoin de réparer leur logiciel vu que les fournisseurs des services s'assurent que la version la plus récente du service soit disponible et comprenne des correctifs aux défauts ou vulnérabilités récemment découvertes. Du point de vue des fournisseurs, les services leur permettent de cibler un ensemble plus grand de consommateurs. Par example, le service Gmail répond tant aux utilisateurs individuels qu'aux organisations, en permettant ainsi aux fournisseurs d'entrer dans de nouveaux marchés. En plus, ces fournisseurs n'ont plus besoin de traiter les nombreux problèmes de mauvaise configuration qui se produisent souvent sur les systèmes d'utilisateurs mal informés, puisqu'ils peuvent assurer que le service est déployé comme il se doit. Cela permet aux fournisseurs de services d'assurer que ceux-ci fonctionnent comme attendu. Cela réduit aussi les efforts de maintenance du logiciel parce que les fournisseurs de services ont le contrôle sur le fonctionnement du logiciel (dans le cas du Logiciel en tant que service). Ce sont des avantages d'un point de vue pécunier. Concernant la technique, les services ouvrent des perspectives nouvelles. Les services offrent une énorme flexibilité et les applications peuvent utiliser des services de fournisseurs différents pour construire des applications composites et offrir *des fonctionnalités métier aux utilisateurs. L'interopérabilité entre les organisations, qui était limitée il y a peu à juste à la collaboration de seulement quelques partenaires, est maintenant possible d'une manière beaucoup plus large.*

Les organisations adoptent le changement alors que les modèle de fourniture des services s'éloignent en même temps des modèles traditionnels de fourniture du logiciel. Des organisations comme SAP ou Microsoft ont par exemple adopté le paradigme de fourniture de services pour diffuser leur logiciel - aussi bien pour des logiciels de professionnels (comme les services Business Analytics) que des suites bureautiques (par example Microsoft Office offert comme service - Office 365 [118]) et dans le cas de SAP, même des solutions ERP [154], qui sont aujourd'hui accessibles comme des services. Tous ces faits démontrent que les services sont déjà déployés et vont rester utilisés. Ils ne sont plus "la grande chose de l'avenir", ils sont la grande chose du présent.

Techniquement, l'informatique orientée services est perçue comme une évolution de l'informatique orientée composants, modulaire, mais d'un point de vue commercial, il a été très dérangeants, changer les modèles d'activité, le changement du statu quo, l'abaissement des barrières à l'entrée parmi de nombreux autres changements. Toutefois, cette perturbation n'est pas assez prononcée du point de vue du consommateur, surtout pour les consommateurs mobiles. Ceci a conduit à une situation où les consommateurs ont donné trop d'information pour les fournisseurs de services afin d'obtenir gratuitement/ faible coût des services tels que Google, Facebook parmi d'autres. Ces organisations ont commencé à utiliser les données recueillies pour générer des revenus, en utilisant leurs informations à fournir des publicités personnalisées ou annonces localisées. Comme ces services ont commencé à recueillir des données au sujet *d'un gros pourcentage de la population , les gouvernements du monde entier ont commencé à concevoir et à appliquer des règlements [60, 35, 34] que nécessaire à ces organisations de divulguer aux consommateurs des données recueillies. Par conséquent, les consommateurs sont de plus en plus conscience de leurs données en acces, stockées et traitées par ces organisations. Toutefois, avec conscience surgit sur la préoccupation - manque de contrôle [133]. Au cours de leurs données, menant à des préoccupations sur la sécurité et la confidentialité de leurs données - si le fournisseur d'accès utilise ces données à des fins commerciales ou pire - fins malveillantes.*

Les scandales récents que ce soit la National Security Agency (NSA) eavesdropping [14, 64] ou du piratage de stockage cloud des célébrités [69] Comptes à accéder à leurs données privées ont rendu les consommateurs prudents en utilisant des services cloud. Ces incidents, qui deviennent de plus en plus fréquentes, illustrent la nécessité de services de cloud computing sécurisée. D'autre part, pour les utilisateurs finaux, la possibilité de se désabonner de services cloud devient de plus en plus contre-productive, compte tenu de l'étroite intégration avec des services cloud qui est intégré dans de nombreux systèmes d'exploitation courants. Par exemple, Windows 8 et Windows Phone ont une intégration étroite avec Microsoft OneDrive de cloud storage service, bien qu'iOS (qui s'exécute sur le très populaire iPhones et iPads) et OS X ont une intégration étroite avec service de stockage de nuage d'Apple iCloud, et enfin, Android (qui s'étend sur environ 70% des appareils mobiles) est intégré avec GoogleDrive. À une époque où les consommateurs sont de plus en plus posséder plusieurs périphériques (téléphones, comprimés, ordinateurs portables/de bureau) la nécessité de disposer de données et synchronisés sur tous les appareils est la conduite des consommateurs à utiliser ces services cloud. Cela dit, les fournisseurs de services tentent d'assurer la sécurité et la confi-

dentialité des données utilisateur est adressée au sein de leurs services. Ils sont même éduquer leurs consommateurs sur les mesures de sécurité qui ont été mises en oeuvre dans les services [61, 166]. Actuellement, la plupart des fournisseurs de services ont des pages dédiées sur leurs sites web que discuter de leurs politiques de sécurité qui expliquent comment l'information des consommateurs est protégée à divers niveaux d'une abstraction. De telles mesures par les fournisseurs de service pour divulguer leurs politiques de sécurité, bien que nécessaires, ne sont pas suffisantes pour dissiper les inquiétudes des consommateurs.

Le problème est double : a) pour bien connus, des prestataires de services établis - les consommateurs peuvent ne pas douter de l'intention du fournisseur de services de sécuriser les données des utilisateurs ni les revendications faites à propos des mesures de sécurité mises en oeuvre dans le service - cependant, elles peuvent avoir trait à la question de savoir si ces mesures sont mises en oeuvre correctement, sans aucune des vulnérabilités et que les mesures mises en oeuvre sont suffisantes pour fournir le niveau de sécurité requis; b) tandis que dans le cas de prestataires de services, sans une réputation établie, les consommateurs ont des préoccupations non seulement si le service a été mis en oeuvre sans les failles de sécurité, mais également qu'ils ont trait à la question de savoir si le prestataire de services est non malveillantes en premier lieu - manque fondamental de confiance sur les intentions du prestataire de services. Cela conduit à un scénario indésirable où, les consommateurs peuvent trust services offerts par les fournisseurs bien connus, même lorsqu'ils ne peuvent pas être sécurisées, tout en ignorant les services que peut-être sécurisé, mais fournis par les fournisseurs de service inconnu.

Ce problème - qui, essentiellement, est un déficit de confiance sur la sécurité *d'un service - s'étend à même des organisations : lorsqu'ils ont à utiliser des solutions basées sur le service au sein de leurs organisations, ils préfèrent utiliser les services fournis par les fournisseurs connus, ou recourir à établir des accords de niveau de service (SLA) avec certains fournisseurs de services - pour combler le déficit de confiance. Accords de niveau de service s'adressent aux scénarios qui sont encore enracinés dans les modèles de fourniture de logiciel traditionnel, où des organisations, qui ont besoin d'assurance de niveau élevé sur les services (ou logiciel) qu'ils consomment, recourir à établir les SLA comme un moyen de maintenir le fournisseur de services peut être tenu pour responsable en cas de violation des termes et conditions convenues.*

Toutefois, Service Oriented Computing a créé un changement révolutionnaire dans les modèles d'affaires où les services d'un fournisseur de services externes de faire usage de provenant de plusieurs autres fournisseurs conduisant à une chaîne de service orchestration. Dans de tels scénarios, l'utilisateur final peut avoir à dépendre de plusieurs SLA pour consommer un service unique - ce n'est ni scalable ni souhaitable.

En outre l'assurance acquise de SLA est a posteriori et basé sur le fait qu'en cas de toute violation des termes et conditions convenues, le fournisseur de services ne peut être tenu responsable. Une présomption critique, dans ces scénarios, est que les violations aux termes et conditions sont détectées et prouvable. Cependant, dans les environnements de service, puisque les consommateurs n'ont pas de contrôle sur le service ou son environnement opérationnel, il devient plus difficile pour les consommateurs de détecter de telles violations. Par conséquent SLA, bien qu'essentielle, ne peut à lui seul fournir l'assurance requise dans les environnements de service.

Un autre moyen pour les consommateurs d'obtenir la confiance de la sécurité *d'un service est d'avoir une tierce partie de confiance de vérifier et de valider un service et atteste que le service répond à certaines propriétés de sécurité. C'est similaire à un processus de certification de la sécurité où les tiers de confiance est, généralement, l'autorité de certification. C'est une solution évolutive pour service basé les écosystèmes comme il l'adresse l'assurance de sécurité question à un niveau modulaire, il n'y PAR, permettant des applications composites à être construit en utilisant plusieurs "certifié" services. Il existe plusieurs régimes de certification de sécurité en pratique, aujourd'hui, telles que des critères communs pour la technologie de l'information (CC) [45], produit commercial Assurance (CPA) [17], La Certification Sécuritaire de Premier Niveau(CSPN) [2] et ainsi de suite. Critères communs (CC) est la plus populaire, reconnu et utilisé scheme parmi les existants. Il y a plus de 1900 produits [44] qui sont certifiés par le schéma des Critères communs. Toutefois, après un examen attentif des produits certifiés par des critères communs et l'autre des systèmes de certification existants, il est remarqué qu'aucun service basé solutions ont des certifications de sécurité. Étant donné la popularité et l'utilisation très répandue des solutions basées sur des services et des préoccupations de sécurité de solutions basées sur des services, il est important de comprendre les raisons derrière ce manque apparent de solutions de service certifiés de sécurité.*

Nous avons examiné les plans de certification de sécurité actuellement utilisé afin de comprendre les raisons du manque d'intérêt des fournisseurs de service de subir la certification de sécurité.

Énoncé du problème

Certaines des principales limitations pour l'adaptation des régimes de certification de sécurité pour les environnements de service sont indiquées ici:

(a) *Une des raisons principales pour des limitations des schémas de certification existantes dans les environement services est le fait qu'ils ont tous été conçus pour des modèles de provisionnement du logiciel traditionaux. Service Provisioning modèles amènent avec eux une complexité en fournissant l'assurance de sécurité pour les consommateurs. Les services s'exécutent sur une infrastructure qui est en partie contrôlé par les fournisseurs de service et, dans certains cas, le fournisseur de service n'a aucun contrôle sur eux. Dans les deux cas, le consommateur n'a aucun contrôle sur l'exploitation du service ni aucun contrôle sur l'environnement d'exécution du service. Il s'agit d'un point de départ clé de modèles de fourniture de logiciel traditionnel, où les régimes de certification actuels déléguent certaines responsabilités au consommateur pour "l'opération sécurisée" du produit certifié*

(b) *Un autre défi clé est qu'actuellement, les systèmes de certification sont à l'échelle du système les certifications - où qu'ils certifient grands produits, ces produits ont un long cycle de développement et suivez la cascade traditionnels [24] modèles pendant leur développement et ces produits sont mis à jour de manière planifiée et progressive. Toutefois, avec les services, ce scénario ne pas contenir de l'eau puisqu'ils sont dynamiques, construit en utilisant les méthodologies Agile, fréquemment mis à jour sur une base hebdomadaire quotidienne/ soit à fournir plus de fonctionnalités ou pour résoudre des bugs qui ont été trouvé ou pour assigner les vulnérabilités.*

Et ces changements aux produits certifiés typiquement invalider la certification acquise qui suscite des préoccupations sur l'utilité pratique de l'attestation de sécurité en premier lieu.

(c) *Services modulaires et elles consistent généralement en plusieurs parties pendant le provisionnement et la consommation - et régimes de certification traditionnels ne répondent pas aux besoins de tels scénarios comme ils supposent que le vendeur d'un produit certifié est le propriétaire de l'ensemble du produit qui est en cours d'évaluation [57].*

(d) *Enfin, les architectures orientées service comptent sur raisonnement automatisé pour la découverte de service, service binding et composition de service (dans certains cas) et dans de tels scénarios certifications de sécurité traditionnels ne semblent bien s'en sortir, car ils sont principalement représentés dans documents lisible par l'homme en les rendant inutilisables dans des situations où pourraient être essentiels [25].*

Question de recherche

Les restrictions mentionnées dans la section précédente façonne le but de la recherche pour la thèse :

Comment pouvons-nous appliquer la sécurité des systèmes de certification pour les environnements de service?

Afin d'atteindre l'objectif principal de la thèse, il y a quelques questions clés auxquelles il faut répondre. Comme expliqué dans la Section 9, Service

Provisioning est très différente des logiciels traditionnels les modèles de provisionnement. En outre, les méthodologies de développement de service diffèrent aussi crûment de méthodologie de développement des systèmes logiciels traditionnels. Par conséquent, une question importante à laquelle il faut répondre est :

> *Comment le processus de certification pourrait-il être modifié afin de le mettre en service pour des environnements qui sont dynamiques et modulaire?*

Le processus de certification, mentionné dans la question ci-dessus, encapsulent à la fois la production et le maintien de la certification de sécurité. Toutefois, le maintien de la certification est un aspect clé, étant donné que les services s'exécutent à distance et ne sont pas sous le contrôle du service consommateur.

Une autre question clé à laquelle il faut répondre est :

> *Comment peuvent les certificats de sécurité participer aux scénarios typiques aux service tels que service discovery et composition de service?*

Cette question porte sur le dernier point mentionné dans la Section 9, c'est, comment devraient être formulées et les certificats de sécurité représentés qu'ils permettent d'automatiser le raisonnement à exécuter à leur égard. Les certifications de sécurité qui permettent de raisonnement automatisé à exécuter à leur égard peuvent avoir un impact profond non seulement sur les processus de certification de sécurité du service, mais il peut changer la manière dont *les services sont consommés, similaire à la façon dont les certificats d'identité ont facilité la façon dont nous transmettons et consommer des informations à partir de serveurs distants.*

Contribution de thèse

Afin de répondre aux questions de recherche, la thèse se concentre sur la production, l'entretien et la consommation des certifications de sécurité pour les services. À cet égard, les principales contributions de la thèse peuvent être résumées comme suit:

Paysage de certification de sécurité

Les termes "sécurisées" ou "Confiance" sont utilisés de façon très libérale par les fournisseurs de logiciels alors que la publicité de leurs produits. Toutefois, en réalité, les produits logiciels sont rarement protégés même lorsque le produit fournisseurs mettent en oeuvre la meilleure sécurité de pointe des mesures pour la protéger. Cela est dû au simple fait que menace pour l'environnement est dynamique de logiciels [39] - il y a toujours des gens essayant de contourner les meilleures mesures de sécurité mises en place pour voler des informations, causer des dommages à l'information ou le système d'information lui-même, altérer les informations ou le fonctionnement du système d'information elle-même - ou simplement parce qu'ils apprécient le défi de contourner le "logiciel sécurisé" [92]. Par conséquent, il est important que le produit est mis à jour avec les derniers correctifs ou des correctifs pour empêcher le produit d'être exploitées par des attaquants.

Plus souvent qu'autrement, les fournisseurs de produits ne mettent pas en oeuvre toutes les mesures de sécurité nécessaires requis pour contrer les men-

aces. Dans de tels cas, la tâche d'un pirate devient beaucoup plus simple, clairement, car il y a des échappatoires grâce auquel l'ensemble des mesures de sécurité du logiciel système peut être contourné. Par conséquent, il est important que le produit possède les mesures de sécurité nécessaires pour contrer les menaces identifiées.

Une autre question importante est que les mesures de sécurité mises en oeuvre par les développeurs de produits peuvent contenir des défauts d'implémentation [93] ou plus communément appelées des "defauts", et s'ils passent inaperçus dans la phase d'essai, ces bogues finissent dans le logiciel. Les mesures de sécurité qui sont viciées deviennent très graves menaces à la sécurité du logiciel. En tant qu'attaquant qui réussit à exploiter une vulnérabilité non seulement contourne la mesure de sécurité, mais peuvent en tirer profit à l'intérieur de l'accès à modifier le fonctionnement du reste des mesures de sécurité. Enfin, les menaces aux systèmes logiciels ne sont pas dues uniquement à des attaques actives ou passives par peuple déterminé, ils peuvent surgir en raison de l'possèdent une mauvaise conception conduisant à l'échec ou le dysfonctionnement [143], entraînant la perte de données, ou de modification de données involontaires de divulgation de données à des personnes non autorisées. Il est donc d'une importance vitale que les mesures de sécurité sont exempts de défauts de conception et de mise en oeuvre.

Pendant l'obtention d'un logiciel, les consommateurs sont généralement axés sur la sécurité du produit et même lorsque le logiciel prestataires prétendent que le logiciel est protégé (contre certaines menaces identifiées), les consommateurs ont encore des préoccupations de l'utiliser. Cela est dû au déficit de confiance qui existe entre les fournisseurs de logiciels et les consommateurs. Les consommateurs ont besoin d'une certaine assurance à étayer les réclamations faites par les éditeurs de logiciels à propos de la sécurité du logiciel.

Généralement, déficit de confiance pour les consommateurs survient lorsqu'elles ont à se procurer et utiliser des logiciels tiers pour leurs besoins informatiques. Les grandes organisations disposent en interne de testeurs [122] Afin d'évaluer un produit avant qu'il est utilisé au sein de leurs organisations. Ces tests se concentre principalement sur la boîte noire approches, étant donné qu'ils n'ont pas accès au code source du logiciel tiers.

En outre, toutes les nouvelles mises à jour du logiciel ont à subir le processus de test. Toutefois, comme le nombre de logiciels tiers utilisés au sein d'une organisation augmente de telles solutions de test interne devient impossible, pour les organisations, car le coût de maintien des équipes pour faire face à différents logiciels est trop élevée.

Par conséquent, de nombreuses grandes organisations entrent en accords de niveau de service avec les fournisseurs de logiciels tiers. Accords de niveau de service offrent un niveau un niveau acceptable de confiance pour les logiciels consommateurs, car ils peuvent assurer que le fournisseur de logiciel peut être tenu pour responsable en cas de dommages causés par le produit logiciel. Toutefois, les accords de niveau de service n'évoluent pas bien et ne peut pas faire face à la nature dynamique des environnements de service. En outre, les accords de niveau de service ne peut pas fournir "a priori" assurance, au lieu de cela, ils sont utilisés comme un moyen de tenir le fournisseur de services peut être tenu pour responsable en cas de violation.

Un autre moyen de parvenir à la sécurité assurance est par l entremise d une certification de logiciels [57]. La certification de sécurité implique généralement un tiers, habituellement l'autorité de certification, trusted tant par les consommateurs et les fournisseurs de logiciels, qui évalue le logiciel en fonction de certains critères et, par conséquent, certifie qu'il possède certaines fonctions de sécurité. Le consommateur peut faire confiance les revendications *faites par les autorités de certification et l'assurance de sécurité requises de gain. À un niveau très élevé de régimes de certification de sécurité peut classer en deux grandes catégories : processus fondé et produit à base [66].*

Certifications de sécurité basée sur les processus se concentrent généralement sur la non-il environnement opérationnel du logiciel. Ils vérifient si l'organisation a des stratégies de gestion de la sécurité adéquate, des mesures de sécurité physiques mises en place et sont utilisés correctement. Ils fournissent habituellement l'assurance a posteriori comme, ils peuvent vérifier s'il y a eu des violations qui se sont produites dans le passé. Des régimes tels que ISO 27001 [91], SAS 70 [155] entrent dans cette catégorie ainsi que l'audit basé certifications.

D'autre part, produit en fonction des certifications permettent aux consommateurs d'obtenir l'assurance, c'est a priori, le consommateur peut savoir avant-la main si le produit certifié répond à leurs exigences en matière de sécurité et que le logiciel a été mis en oeuvre sans failles. Il soulage également les fournisseurs de logiciels et les consommateurs afin d'établir les SLA les uns avec les autres, étant donné qu'ils peuvent faire confiance aux déclarations du vendeur qui sont étayés par la certification délivrée par les autorités de certification.

Nous nous sommes concentrés sur les certifications de sécurité basées sur du produit et notamment analysé les Critères communs (CC) certification près - à la fois le régime de certification et la certification Pratique. C'est parce qu'entre la sécurité actuelle des systèmes de certification pour le logiciel, Critères communs est le plus largement utilisé [44], reconnu et [50] est applicable à un large éventail de catégories de produits. Il est reconnu dans 26 pays, assurant ainsi une immense valeur aux fournisseurs de produits par l'obtention d'un produit certifié une fois et l'avoir reconnu dans le monde entier. Le CC régime est l'aboutissement d'années de travail par divers organes gouvernementaux *du monde entier afin d'harmoniser les critères pour évaluer les produits informatiques.*

La certification Common Criteria Scheme provenaient de Trusted Computer System Evaluation Criteria (TCSEC) [171], la Loi Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) [107] et la sécurité de la Information Technology Security Evaluation Criteria (ITSEC) [52]. Il a unifié ses prédécesseurs avec une série de critères d'évaluation de sécurité. Il découple la spécification des exigences relatives à l'assurance et de sécurité fonctionnelles, cela est en opposition directe avec l'approche de son prédécesseur TCSEC où la sécurité fonctionnelle et d'exigences en matière d'assurance sont couplés [171] ensemble afin de fournir une assurance "équilibré"à l'égard d'un système. Ce découplage est nécessaire comme CC cible un marché de sécurité commerciale, tandis que l'TCSEC se limitait aux produits conçus pour nous gouvernementales, et a besoin d'être générique afin de certifier différentes sortes de produits qui vont des logiciels, des micrologiciels du matériel. Le CC régime permet les fournisseurs de produits pour décrire les Exigences Fonctionnelles de Sécurité (EFS) pour le produit et pour prouver que l'ensemble de EFS sont en mesure de contrer les menaces identifiées pour une Cible d'évaluation, qui identifie les aspects spécifiques du produit qui sera évaluée. En outre, le régime permet aux vendeurs de produits CC pour choisir des configurations particulières du produit qui seront évaluées et ces configurations "doré" font aussi partie de la cible d'évaluation. Cette information est recueillie dans un document appelé "cible de sécurité" (CC-ST) qui peut être considérée comme la partie descriptive de la CC la certification [21]. Le fournisseur du produit puis définit l'ensemble des Exigences d'assurance de Sécurité (EAS) - qui spécifient les actions à réaliser par l'évaluation de laboratoires. Basé sur les EAS sélectionné pour le produit les autorités de certification de déterminer le *niveau d assurance de l évaluation (AE).*

L'inconvénient de cette approche est que AE peut uniquement spécifier le niveau de détail de l'évaluation a été effectuée, mais il ne répond pas à la question de "est le logiciel secure?". La réponse à cette question peut être fournie par la EFS qui sont mis en oeuvre dans le produit.

Le CC régime est générique et ne pas imposer des exigences spécifiques pour différents types de produits. D'où les fournisseurs de produits peuvent mettre en oeuvre certaines fonctionnalités de sécurité (EFS) et obtenir des parties spécifiques de leur système évalué d'une certaine façon (EAS) et, par conséquent, certifié, qui peuvent ne pas répondre aux exigences des consommateurs. Pour résoudre ce problème, CC permet aux consommateurs d'utiliser les profils de protection (CC-PP) qui contiennent une combinaison de sfr et le EAS pour un type particulier d'application, telles que le système d'exploitation ou des bases de données. Lorsque les produits sont conformes à un profil de protection spécifique, il est plus facile pour le consommateur de choisir et de comparer la mieux adaptée à leurs besoins. Mais la conformité aux CC-PP par des produits n'est pas obligatoire, et il y a une critique que les fournisseurs de produits d'exploiter cette flexibilité de la CC scheme, et choisir de ne pas se conformer à toute les profils de protection qui pourraient être appliquées pour leurs produits [41, 161].

En dépit de la large utilisation et le succès économique de critères communs scheme [163, 80] (principalement dictée par la réglementation gouvernementale et les achats gouvernementaux) sa pratique actuelle a bénéficié d'importants critiques. Fondamentalement sur ces aspects suivants : (a) Comparabilité; (b) cértification à un moment donné; (c) Long et coûteux processus de certification; (d) Préoccupations pour la reconnaissance mutuelle.

Nous avons procédé à une analyse pour évaluer si les deux premières cri-

tiques sont vraies. Les points (c) et (d) ont été exclus de cette analyse, car : une analyse sur les coûts et la durée des certifications CC a déjà été discutée dans [132, 180] (c), l'adressage et le problème de la reconnaissance mutuelle (d) ne peuvent pas être analysés en regardant les certificats. À partir de l'analyse, nous avons découvert que:

Comparabilité: *nous avons constaté que, sans l'assistance de l'outil il n'est pas une mince tâche pour effectuer des comparaisons entre les produits basé non seulement sur leurs AE mais aussi leur EFS. À cet égard, les objectifs de sécurité doivent être représentés dans une machine facilement transformable manière qui facilite raisonnement automatisé pour être exécutées sur eux.*

Cértification à un moment donné *le faible nombre de produits sous maintenance soulèvent une importante question sur le cycle de vie du produit, notamment lorsque des vulnérabilités sont trouvées dans le produit qui ont besoin de l'être - peut le fournisseur d'un produit question un correctif et techniquement perdre la certification ou continuez à vendre la version vulnérable du produit pour demander la certification? Il est plutôt évident, que le vendeur du produit choisira pour corriger les problèmes et risque de perdre la certification.*

Nos résultats montrent que, malgré la conclusion de nouvelles vulnérabilités, qui sont parfois inconnue au moment de la certification initiale, et les dispositions prises par le schéma des Critères communs pour appuyer la certification incrémentielle (CCMA), les produits certifiés sont massivement certifié une fois pour toutes. Bien que ceci soit parfaitement valide en soi, il montre que deux certificats devraient être comparés quant à leur date d'émission, ainsi qu'à l'égard de l'information à la disposition du public des bases de données de vulnérabilité (comme NVD). Il illustre également que la description de la cible d'évaluation et EFS devait préciser si de nouvelles vulnérabilités doivent être *considérées, et devrait en particulier bénéficier d'être lisibles en machine pour automatiser cette identification.*

L'assurance de Sécurité des Services

Services La complexité et la sécurité supplémentaires apportent aux préoccupations découlant de la perception d'un manque de transparence, de contrôle et de reddition de comptes. Accords de niveau de service aide à aborder les questions relatives au manque de responsabilisation, mais ne peut pas répondre aux préoccupations sur le manque de transparence et de contrôle. En outre, les environnements de service sont très dynamiques et donc ne peut pas bien évoluer de SLA de tels scénarios. Alors que processus fondé des certifications de sécurité sont utilisés abondamment dans les paysages de service afin de compenser l'absence du service client de contrôle, pourtant, ils ne parviennent pas à s'assurer de la transparence.

Produit à base des certifications de sécurité tels que le CC et le CSPN peut donner l'assurance que s'attaque au manque de transparence dans les environnements de service. Toutefois, CC et CC des régimes, dans leur forme actuelle, ne pas bien s'adapter aux environnements de service et ne peuvent donc pas être utilisées pour obtenir l'assurance de sécurité nécessaires aux consommateurs pour les raisons suivantes : (a) aucune facilitation de raisonnement automatisé; (b) elles sont adaptées pour "déployables" logiciel services plutôt que de "déployés"; et (c) ne peut faire face avec service compositions. La prochaine partie de la thèse propose des solutions qui permettront de surmonter ces limitations.

Appliquant la certification de sécurité pour SOA

Nous avons analysé et identifié les besoins d'un certificat de sécurité dans les environnements de service. Dans un environnement de service, les consommateurs devraient être en mesure de comparer les fonctionnalités de sécurité (certifié) d'un service avec leurs exigences en matière de sécurité et en outre de comparer les fonctionnalités de sécurité (certifié) des offres de service provenant de différents prestataires de services. Mais la sécurité des certificats qui sont représentés dans le langage naturel, rempli de jargon juridique et sont plutôt non structurées, empêcher toute sorte de raisonnement automatisé pour être exécutées sur eux. Afin d'apporter la sécurité à la certification les environnements de service de façon qu'ils peuvent jouer un rôle constructif dans la sélection de service et service processus de consommation, plusieurs modifications de l'état actuel de l'art sont nécessaires [104]. Parmi eux, nous nous sommes concentrés sur les aspects liés à la sécurité et la représentation de la génération du certificat.

Modèle conceptuel de certificat de sécurité numérique

Nous présenterons notre proposition pour un certificat de sécurité numérique de concept (désormais dénommée $\mathcal{CRT}$), qu'à notre avis, aborde les questions de certification de sécurité mentionnées dans soc. En fait, les tubes cathodiques sont conçus pour être lisibles à la machine. Ils sont conçus pour faire face aux modèles d'interaction de service web existantes et des normes et, partant, à faciliter leur adoption et intégration dans soc. En outre, $\mathcal{CRT}$ s sont conçues pour être entièrement descriptive et donc autoriser raisonnement automatisé sur eux. En particulier, $\mathcal{CRT}$s peut fournir des preuves de la présence et de la mise en oeuvre de fonctions de sécurité d'un service, qui sont recueillies *dans le cadre d'une phase d'évaluation des services; cela permet d'approfondir l'analyse client, à l'égard des exigences de sécurité spécifiques. La liaison entre l'implémentation d'un service et son certificat de sécurité alors que dans la forme actuelle de $\mathcal{CRT}$ nous fournir un élément spécifique à contenir de l'information qui aide à assurer la liaison. Enfin, la représentation de certificat de sécurité est conçu pour être assez large pour capturer des informations provenant de différents systèmes de certification et nous introduisons le concept d'un certificat de sécurité numérique de profil qui permet aux autorités de certification et/ ou le service consommateurs pour être en mesure de spécifier les exigences sur un écran $\mathcal{CRT}$ - tant de sémantique et syntaxique.*

Le modèle conceptuel pour le $\mathcal{CRT}$, est conçu pour recueillir de l'information émanant des procédés de certification de sécurité. En particulier, nous avons examiné le CC régime, puisqu'il est le plus plus large régime actuellement. Le CC-ST, qui est la partie descriptive de la CC scheme, sert de base à notre $\mathcal{CRT}$. Toutefois, nous avons étendu ce considérablement, afin de rendre la machine facilement transformable et adapté pour les besoins spécifiques à chaque service. En revanche à CC, et d'autres les programmes existants de certification, le certificat de sécurité numérique est conçu pour être entièrement descriptive [176], et donc il contient la description de l'entité certifiée, les propriétés de sécurité de l'entité certifiée, les détails de l'évaluation qui sous-tendent les propriétés certifiées. Nous avons officialisé ce modèle et réalisé le modèle du certificat numérique $\mathcal{CRT}$ grâce à un langage basé sur XML qui permet la représentation du certificat dans une machine traitables formulaire, qui nous désignons comme une ASSERT.

Dans le modèle conceptuel nous ont présenté les éléments qui doivent être capturés dans un certificat de sécurité numérique et nous avons présenté une langue à travers lequel nous réalisons ce modèle conceptuel dans une machine *facilement transformable manière (`ASSERT`). Toutefois, l'affirmer la langue fournit une structure de données pour représenter les certificats, mais ne donne ni prescrire les données qui doivent être contenues dans les structures de données. C'est un choix intentionnel afin d'avoir une claire séparation entre le modèle conceptuel, la réalisation et le contenu réel des certificats, qui par exemple aurait facilité l'adoption de $\mathcal{CRT}$ avec différents systèmes de certification et d'évaluations.*

Par conséquent, l'assertion des éléments linguistiques devraient faire usage des vocabulaires, qui doivent être définis par les autorités de certification différentes pour leurs régimes respectifs sur la base de la certification/ les processus d'évaluation et les types de produits qui sont certifiés. Les vocabulaires peut apporter de la sécurité existante des ontologies pour décrire différents éléments dans la langue `ASSERT`, permettant ainsi le raisonnement sur eux, en prenant également bénéficier de données couplées [29] modèle, à l'égard d'établir un lien avec d'autres d'ontologies. Un exemple de cette souplesse est représentée par l'utilisation d'une ontologie, appelé USDL-SEC C, pour exprimer les mécanismes de sécurité dans la langue `ASSERT`, tandis que des vocabulaires spécifiques sont également prévues pour l'expression d'autres éléments, comme de assert propriétés de sécurité.

Afin d'apporter plus d'homogénéité et d'uniformité entre les certificats générés par les régimes de certification de sécurité spécifique, nous avons introduit le concept de profil de certificat de sécurité numérique ($\mathcal{CRTP}$). Le principal objectif d'un programme de formation itinérante est de fournir des moyens adéquats pour la création de certificats en assurant l'uniformité des certificats de sémantique une certification particulière capturer toute certification et évaluation des aspects spécifiques, le vocabulaire de produits certifiés, propriétés de sécurité, ou d'autres aspects pertinents à la sémantique du $\mathcal{CRT}$s.

Cycle de vie de la certification de la securité numérique

La représentation d'un certificat de sécurité sous une forme déchiffrable par machine est l'un des nombreux changements nécessaires pour apporter la sécurité des environnements de services certifications. Alors que les questions de convivialité sont abordées à l'aide du concept de certificat de sécurité numériques, l'assurance obtenue des régimes de certification est toujours assez faible, étant donné qu'actuellement la certification de sécurité adopte un "Cértification à un moment donné" approche. Dans de telles approches, une version d'un service peut subir le processus d'évaluation et, par conséquent, être certifié. Cependant, les consommateurs ne peuvent obtenir l'assurance qu'au cours de la consommation, ce qui peut se produire à tout moment après la certification du service, que l'instance de service est la même version qui est certifié et la plate-forme, sur lesquelles le service fonctionne, n'a pas changé depuis le configurations certifiées.

En fait, dans les environnements de service, les prestataires de services mettent à jour leurs offres de services souvent, presque sur une base quotidienne/hebdomadaire, pour ajouter de nouvelles fonctionnalités au service, ou déployer des implémentations de services optimisée ou corriger des bugs ou des vulnérabilités. Le fait que la majorité de la cloud service se fait le développement à l'aide de méthodologies de développement agile contribue à ce cycle de mises à jour fréquentes et ces environnements dynamiques est l'un des avantages fondamentaux que les développeurs obtiennent à partir d'environnements de service. Étant donné que tout changement dans le service se produit de manière transparente à partir de la perspective du consommateur, les développeurs peuvent exploiter cet avantage pour réduire le temps de mise sur le marché en fournissant les fonctionnalités de base et l'ajout de plus de fonctionnalités supplémentaires. Mais la sécurité actuelle de régimes *de certification sont relativement rigides quand il s'agit de changements de services certifiés. Ils considèrent tout changement au service certifié ou son environnement opérationnel sous-jacent (OE), qui comprend à la fois les aspects informatiques (IT-OE) et les aspects non-informatiques (non-IT-OE) ,qui ne fait pas partie de la configurations certifiées pour invalider la certification acquise. Cela constitue un grave obstacle pour le rendre attrayant pour les fournisseurs de services afin d'obtenir leurs services certifiés. Dans ce chapitre, nous présentons une approche pour le certificat de sécurité numérique de maintenance pouvant faire face à de fréquentes mises à jour de maintenance et peuvent surveiller l'IT-OE pour tout changement. Toutefois, dans la certification de sécurité de service, il y a beaucoup de différentes entités participantes et les diverses relations de confiance qui existent entre ces entités ont une implication sur les assurances de sécurité qui est acquise par la certification et, en particulier, ont un impact direct sur les assurances de sécurité sur le IT (et non) environnement opérationnel. Par conséquent, nous présentons d'abord un peu des scénarios clés des relations de confiance qui existent entre les différentes entités dans les environnements de service.*

Nous avons proposé un moniteur de certification de sécurité numérique (MCSN) qui traite de l'entretien de la $\mathcal{CRT}$. Le MCSN Framework expose plusieurs interfaces pouvant être consultée par diverses entités. Toutes les opérations sur les certificats est effectuée dans un composant appelé registre de certificats de couche d'abstraction qui à son tour interagit avec le certificat référentiel pour stocker et récupérer des certificats.

Semblable à une PKI [119] ou pour le schéma des Critères communs [48], les autorités de certification permet-cadre MCSN, pour enregistrer les certificats qui sont délivrés ou de révoquer toute certification en cas de violations. L'une des composantes clés de l'MCSN est le service cadre Moniteur de certifi-

cat (SMC). La SMC effectue des contrôles périodiques sur le service déployé et la configuration de la plate-forme. La fréquence de ces vérifications peuvent être ajustés selon les exigences de la plate-forme et des autorités de certification de fournisseur. En outre, le SMC peut être invoqué explicitement par un consommateur de service à travers le MCSN cadre. Lorsqu'un consommateur prie le MCSN cadre permettant de vérifier un certificat de sécurité du service, les MCSN, vérifie la validité des signatures, ainsi que la justesse sémantique et syntaxique du certificat. Il demande ensuite le module de SMC correspondant à la plate-forme sur laquelle le service est déployé pour vérifier l'instance de service pour s'assurer que l'instance actuelle du service répond encore aux propriétés structurel certifié.

Le MCSN cadre avec le nous avons proposé module de SMC contribue à augmenter la qualité de l'assurance que les consommateurs profitent de l'Certificats de sécurité numériques. Notre cadre améliore l'aspect de maintenance de la cycle de vie des certificats de sécurité. Notre proposition ne répondent pas seulement l'aspect de fournir une assurance sur l'environnement opérationnel, mais accroît également la transparence concernant l'environnement opérationnel d'un service au cours de la consommation de services.

En introduisant la notion de certificats repérés, nous permettons à l'assurance de se dégrader au fil du temps, si et quand il y a des changements au service ou son environnement opérationnel que les impacts propriétés certifiées. Cela est très important, car il permet aux consommateurs d'effectuer leur propre analyse de risques pendant la consommation de service s'ils souhaitent consommer un service avec un certificat de sécurité signalées. Ainsi, l'utilisateur est habilité avec des informations à jour concernant l'instance de service, plutôt que de dépendre d'un certificat de sécurité qui a été publié à un moment donné dans le passé.

Le MCSN Cadre est également capable de faire face à des mises à jour fréquentes des services et ainsi permettre aux prestataires de services d'obtenir le core produit certifié et ajouter plus de fonctions d'une manière qui n'affecte pas les propriétés de sécurité certifié. Cela incite aux prestataires de services qui ont leurs services en phase de certification, et confère à l'ensemble de l'écosystème de service certifié un énorme élan.

En outre, l'Accord MCSN et SMC module, nous proposés sont de nature générique et peut être adapté à de multiples régimes de certification de sécurité ainsi que les fournisseurs de plates-formes multiples. En cours de certification scheme agnostique, encore une fois, permet la thèse des contributions d'avoir une plus grande incidence sur le paysage de certification de la sécurité.

Conclusion et orientations futures

Assurance de Sécurité est un aspect important à aborder dans les environnements de service afin de faciliter un plus large adoption de solutions basées sur des services, en particulier dans les domaines critiques d'entreprise. Dans cette thèse, nous avons étudié les différents moyens d'assurance de sécurité de gain dans les modèles de provisionnement traditionnel, et analysé leur applicabilité dans les environnements de service. Nous avons montré que la certification de sécurité est une solution efficace et évolutive pour acquérir l'assurance sécurité dans les environnements de service.

La première partie de cette thèse, présente une analyse exhaustive de la certification de sécurité de paysage. Nous classons les régimes de certification de la sécurité dans le processus et le produit basé les régimes de certification et d'expliquer les assurances qui peuvent être acquises dans le cadre de ces systèmes de certification. Nous nous concentrons sur régime de certification de sécurité des Critères communs, puisqu'il est le plus largement utilisé, reconnue *et largement applicable. Nous avons expliqué en détail sur le schéma de certification CC - le processus de certification, les documents de certification qui résultent du processus d'accréditation, les méthodologies d'évaluation parmi d'autres.*

L'une des principales contributions de la thèse est l'analyse quantitative de la certification de sécurité des Critères communs la pratique que nous avons effectuées afin de comprendre si le CC pratique reste fidèle à l'esprit de la CC scheme. Nous avons présenté les résultats de l'analyse qui a mis en évidence plusieurs limitations de la CC pratique qui suscite des craintes au sujet de l'assurance les consommateurs acquièrent à travers CC certification.

Dans la deuxième partie de la thèse, nous avons présenté le modèle conceptuel d'un certificat de sécurité numérique. Fondé sur le modèle conceptuel, nous avons mis en oeuvre un langage basé sur XML qui permet de service à des certificats de sécurité pour représentées dans un format lisible par machine. Nous avons en outre présenté la flexibilité de notre langue pour capturer des informations provenant de différents programmes de certification. Cela facilite une adoption plus large de la notion de certificat de sécurité numérique par différents régimes de certification de la sécurité. En outre, nous avons proposé le concept d'un profil de certificat de sécurité numérique qui peut être utilisée par des autorités de certification à prescrire certaines contraintes sur les certificats de sécurité sont délivrés. Les profils de certificat de sécurité numérique peut également être utilisée par des consommateurs pour leurs besoins de sécurité de l'état et vérifiez, à l'aide de l'outil de soutien, si un certificat de sécurité est conforme à leur profil. La thèse présentée une approche qui facilite des autorités de certification afin de surveiller les services certifiés et son environnement d'exploitation pour s'assurer que les propriétés certifiées sont toujours satisfaits. Notre approche prend également en compte les mises à jour *fréquentes qui sont faites aux services et a donc un mécanisme qui en déduit si une mise à jour de l'implémentation d'un service viole toute propriétés certifiées.*

Les contributions de cette thèse ont le potentiel de permettre à service domaines critiques dans les affaires d'adoption tels que la santé, la défense, des finances parmi d'autres. Elle permet également la normalisation de la représentation de certificat de sécurité entre les divers régimes.

Il y a plusieurs questions qui doivent être étudiées davantage comme les critères utilisés dans différents processus de certification pour un logiciel d'évaluation n'est pas normalisée et encore certains de ces éléments doivent être testés ou prouvé manuellement. Par conséquent, proroger le délai pour obtenir la certification qui peut s'avérer critique dans les environnements de service où Temps de mise sur le marché est extrêmement crucial. Par conséquent, nous prévoyons enquêter davantage sur les différentes approches que peut évaluer automatiquement les services pour certaines propriétés de sécurité. Une autre question qui mérite d'être étudié plus loin, c'est la composition du certificat de sécurité. Raisonnement Il faut mettre au point des mécanismes qui peuvent déterminer au cours de l'exécution si un service certifié peut se lier avec un autre service certifié de manière que les propriétés de sécurité de son premier service ne sont pas violés.