WG Attack Taxonomy

Marc Dacier, Herve Debar, Thorsten Holz (note taker), Engin Kirda, Jan Kohlrausch, Christopher Kruegel (chair), Konrad Rieck, James Sterbenz

Introduction

The starting point of this working group was the question about the kinds of attacks that can be detected by inspecting in network traffic. In general, we identified four major problems that network-based intrusion detection systems are facing:

1. Encrypted network traffic
2. Application-level attacks
3. Performance
4. Evasion attack.

An obvious problem in this area is payload inspection of encrypted traffic: since the network-based intrusion detection system (NIDS) commonly has no access to the encryption keys, it can not decrypt the captured data and, therefore, no analysis is possible. From a network perspective it is thus hard to deal with encrypted traffic. However, more and more traffic within networks uses some form of encryption (e.g., IPSec or SSL) and, thus, we need to develop approaches to also deal with this kind of network traffic in the future. Traditional attack venues such as buffer overruns or exploits of input validation errors have been known for a long time and are widely understood. As a result, a large number of defense mechanisms have been devised [16, 18]. For client-side attacks, however, only a few viable defense solutions have emerged so far. These techniques often focus on one particular problem area only and fail to address the larger and more general problem of unauthorized information flow attacks. A distinctive feature of client-side attacks is that security problems often cannot be traced to a particular vulnerability that can be easily fixed. In fact, the danger is precisely that the client's security policy is not obviously and immediately violated. In case of a cross-site scripting attack, the malicious script is truly sent by the trusted server and thus, has to be granted the privilege to access the session tokens. Similarly, when a user enters sensitive data into a web form on a phishing site or installs spyware, agreeing to the license, one could argue that there is no problem because a deliberate action is taken and information is voluntarily disclosed. Such a point of view, however, neglects the fact that there is an implicit security requirement of users who do not want to disclose their sensitive data. Thus, even when the same-origin policy is not violated by a cross-site scripting attack, there is an implicit policy that dictates that no sensitive user data should be disclosed to unauthorized parties. Furthermore, sending of code from server to client becomes more and more common (e.g., AJAX sends JavaScript over the network) and this new interaction model poses further challenges since a NIDS would need to inspect and verify the code. By moni-

Dagstuhl Seminar Proceedings 08102
Perspectives Workshop: Network Attack Detection and Defense
http://drops.dagstuhl.de/opus/volltexte/2008/1495

toring network traffic, such attacks are not easy to identify as they occur at the application-level: the NIDS would need to understand the context of requests and also keep track of the application state. That is, one needs to understand the application logic and try to detect attacks, which is hard even given the current network speed. Given the fact that networks are getting faster at a higher pace then processing power increasing, this is clearly a problem. Furthermore, so called traffic blending attacks and similar evasion attacks pose several challenges for NIDS [2, 3].

State of the Art

Detecting scan activity in network traffic has attracted a lot of interest in the research community over the last few years. As a result, there are several systems and algorithms that can be used to detect either port scans or to identify worm propagation [7, 8, 10, 15, 19]. Since nowadays attackers commonly use bots to have control over an infected machine [20], the research community has developed some systems to also detect this kind of attack-related traffic [1, 4,5, 6]. Another area of active research is Distributed Denial of Service (DDoS) analysis [11, 12, 13, 17]. A common approach to deal with this kind of attacks DDoS attacks. Nevertheless, it is still an open problem how to differentiate between a flash crowd and an actual DDoS attack. Current traffic monitoring techniques are useful to detect effects of attacks in order to identify hosts that have been compromised. For example, a bot that has been installed by a user because of a social engineering attack or with the help of a successful exploit can be detected by monitoring the network for suspicious behavior: such hosts commonly generate either lots of scan traffic, are suspicious due to a large amount of mails sent via these hosts, or generate lots of DNS queries. All such effects can be easily detected with different traffic monitoring strategies [5, 6]. Furthermore, current monitoring techniques allow us to detect artifacts of attacks. For example, we can detect common attack tools, ready-made exploits, or worms based on specific signatures in the network traffic [9, 21].

Challenges

To improve the detection capabilities from a network point of view and to cope with future challenges in this area, we developed some recommendations for future work in this area. An application should support a NIDS such that it becomes easier to check for ongoing attacks. This could for example be achieved by developing protocols in such a way that the NIDS can verify – without too much overhead – whether or not a given packet is legitimate. Furthermore, additional meta-information which is sent together with the actual application data could help a NIDS to detect attacks. One example of such meta-information is proof-carrying code [14], another example is signed code (with the drawback of requiring a public key infrastructure). Presumably such changes also need to address the interaction model such that the communication is more regular and not too much state needs to be kept by a NIDS. Another option

would be to develop application specific filters that can be deployed in front of servers to protect them.

We require a deeper analysis and metrics to define the complexity of an attack. Here, we understand the complexity of an attack as the difficulty to see this attack at the network level. Clearly, certain attacks are directly visible on the network, for example, as malformed packets (e.g., ARP attacks), as deviations in the number of packets that are sent (e.g., denial of service), or as unexpected target of packets (e.g., hijacked DNS traffic is sent to a malicious DNS server). However, other attacks are not immediately detectable. In particular, attacks that target application-level flaws could be perfectly legitimate from a network perspective, but might arrive at an unexpected point in time or in an unintended order. This would require more complex analysis, requiring the NIDS to keep state or to understand application-level semantics. These facets should be captured by the proposed complexity metric. When such a metric is available, it could guide the designers of NIDS to focus on certain classes of attacks that security officer (who has to deploy these solutions) to determine those classes of threats for which additional levels of protection are required.

An additional area of future work is behavior-based detection of attacks: if we can understand the current, normal configuration of a system, then we can detect deviations from this profile as an attack. To achieve this goal, we need to develop algorithms to understand what operations are normal based on the current configuration (e.g., information about the network configuration, the running services, the clients that make use of these services, ..). With the help of this information, we can then develop technique to achieve behavior-based rather than knowledge-based detection of attacks. However, it is then still challenging to detect covert channels or stealing of information at the network level, thus additional techniques need to be developed to counter these threats. Another open problem that needs to be addressed is the question how much deep packet inspection is needed to detect attacks. With the steady increase of network speed, less time can be spent analyzing each single packet. We need to develop metrics and scenarios that are useful to answer the question in what circumstances netflow data is enough and in what cases more information is needed. A flexible infrastructure in which the network measurements can be adjusted during runtime would be useful to help network-based detection of attacks.

References

[1] James R. Binkley and Suresh Singh. An algorithm for anomaly-based botnet detection. In Proceedings of the 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet, pages 7–7, 2006.

[2] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In Proceedings of the 13th ACM conference on Computer and Communications Security, pages 59–68, 2006.

[3] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In Proceedings of the 15th USENIX Security Symposium, pages 17–17, 2006.

[4] Jan Goebel and Thorsten Holz. Rishi: identify bot contaminated hosts by irc nickname evaluation. In Proceedings of the First Workshop on Hot Topics in Understanding Botnets, pages 8–8, 2007.

[5] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. BotHunter: Detecting malware infection through ids-driven dialog correlation. In Proceedings of the 16th USENIX Security Symposium (Security' 07), August 2007.
[6] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08), February 2008.
[7] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In IEEE Symposium on Security and Privacy 2004, Oakland, CA, May 2004.
[8] Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In Proceedings of the 13th USENIX Security Symposium, pages 19–19, 2004.
[9] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Polymorphic Worm Detection Using Structural Information of Executables. In Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), volume 3858 of LNCS, pages 207–226, 2005.
[10] Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao, and Brian Chavez. Hamsa: Fast signature generation for zero-day polymorphicworms with provable attack resilience. In Proceedings of the 2006 IEEE Symposium on Security and Privacy, pages 32–47, 2006.
[11] Jelena Mirkovic, Gregory Prier, and Peter L. Reiher. Attacking ddos at the source. In Proceedings of the 10th IEEE International Conference on Network Protocols, pages 312–321, 2002.
[12] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. SIGCOMM Comput. Commun. Rev., 34(2):39–53, 2004.
[13] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. ACM Trans. Comput. Syst., 24(2):115–139, 2006.
[14] George C. Necula. Proof-carrying code. In POPL, pages 106–119, 1997.
[15] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In Proceedings of the 2005 IEEE Symposium on Security and Privacy, pages 226–241, 2005.
[16] Vern Paxson. Bro: a system for detecting network intruders in real-time. Computer Networks, 31(23-24):2435 -2463, 1999.
[17] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. SIGCOMM Comput. Commun. Rev., 31(3):38–47, 2001.
[18] Martin Roesch. Snort - lightweight intrusion detection for networks. In Proceedings of the 13th USENIX conference on System administration, pages 229–238, 1999.
[19] Stuart Staniford, James A. Hoagland, and Joseph M. McAlerney. Practical automated detection of stealthy portscans. Journal of Computer Security, 10(1-2):105–136, 2002.
[20] The Honeynet Project. Know Your Enemy: Tracking Botnets, March 2005. http://www.honeynet.org/papers/bots/.
[21] Nicholas Weaver, Stuart Staniford, and Vern Paxson. Very fast containment of scanning worms. In Proceedings of the 13th USENIX Security Symposium, pages 3–3, 2004.