Institut Eurécom

2229, route des Crêtes

B.P. 193

06904 Sophia Antipolis

FRANCE

Research Report RR-03-081

White Paper:

"Honeypot, Honeynet, Honeytoken: Terminological issues[1]"

September 14, 2003

Fabien Pouget, Marc Dacier

Institut Eurécom

Email: {pouget,dacier@eurecom.fr}

Hervé Debar

France Télécom R&D

Email : herve.debar@francetelecom.com

[1] This research is supported by a research contract with France Telecom R&D, contract N. 425-17044

White Paper: "Honeypot, Honeynet, Honeytoken: Terminological issues[2]"

Fabien Pouget, Marc Dacier
Institut Eurecom
Email: {pouget,dacier@eurecom.fr}

Hervé Debar
France Télécom R&D
Email: herve.debar@francetelecom.com

Institut Eurécom
2229, Route des Crêtes ; BP 193
06904 Sophia Antipolis Cedex ; France

France Télécom R&D
42, rue des Coutures ; BP 6243
14066 Caen Cedex 4 ; France

Abstract

Many different terms, definitions and classifications for honeypots, honeynets and other honeytokens have been proposed by several authors during the last 3 years. In this document, we offer a summary of the various proposals and we discuss their advantages and drawbacks. We also offer our own definition at the end of the paper.

1. Introduction

The concept of "honeypots" has been introduced in computing systems by Clifford Stoll in the late 80's. In the 'Cuckoo's Egg' [Stol88], he describes the monitoring and tracking of an intruder. For this purpose, he had to create a complete but non existent government project, with realistic but false files which intruders spent an extended period of time downloading and analyzing, providing an opportunity for him to trace back their origin. It is only in 2001 that the term "honeypot" has been introduced by Lance Spitzner. Since then, several authors have proposed ad hoc definitions and classifications. This white paper offers a survey of the literature.

[2] This research is supported by a research contract with France Telecom R&D, contract N. 425-17044

The paper is organized as follows. Section 2 reviews the various definitions that have been made. Section 3 focuses on the various classifications that have been proposed. However, since we are not really satisfied by these terminologies, Section 4 presents our definition proposal. Section 5 concludes the paper.

2. Existing terminology

2.1 Honeypot definitions

2.1.1 Introduction

In the 90's, Cheswik implemented and deployed a real "honeypot" [Ches92]. Bellovin discussed the very same year the advantages and problems related to its usage [Bell92]. In 98, Grundschober and Dacier ([GrDa98, Grun98]) introduced the notion of "sniffer detector" (see also [AbKK02]), one of the various forms of what is called today a "honeytoken". As one can see, honeypots, honeytokens and honeynets have been used for some time in computing systems even if this terminology is quite recent.

2.1.2 Lance Spitzner's definition

We present in this section some definitions that are found in articles and security web pages. Lance Spitzner, a senior security architect for Sun Microsystems is the author of *"Honeypots, Tracking hackers"* [Spit02]. In this book, he proposes the following definition:

"A honeypot is security resource whose value lies in being probed, attacked or compromised." [Spit02, page 40]

This is the most common definition and many papers refer to it ([Cole01, Baum02, and Seif02]). However its precise meaning is not so clear. If we take a deeper look at it, we see that the definition can be decomposed as follows:

1. One term: « a security resource »
2. A subordinate description: "its value which lies in being probed, attacked or compromised"

Now, the problem is to determine what a 'security resource' is. Lance Spitzner follows the definition up with this comment:

"This means that whatever we designate as a honeypot, our expectations and goals are to have the system probed, attacked, and potentially exploited. It does not matter what the resource is (a router, scripts running emulated services, a jail, an actual production system). What does matter is that the resource's value lies in its being attacked. If the system is never probed or attacked, then it has a little or no value. This is the exact opposite of most production systems, which you do not want to be probed or attacked." [Spit02, page 40]

The reader will note that honeypots, as defined in the previous comment, can be routers or production systems even though they are not really considered as "security resources". Thus, the comment, which is supposed to illustrate the definition, actually proposes another one. Let us take a simple example to illustrate the problems with this definition. Let us assume that we have a firewall that logs all the failed connections he can see. One can reasonably argue that the system is expected to be probed and attacked. Indeed a firewall is by definition the first line of defense against external attacks. It protects production systems against the *wild network* and is consequently exposed to probes and attacks. The question is: is this firewall a honeypot according to Definition 1? We would say that, since its first value resides in the protection it offers, this is not a honeypot. However, based on the comments, this is far from being evident.

In [Spit02, page 42], the following example of a honeypot deployment is given: an old and unused server in the DMZ is closely watching any traffic to or from it. According to L.Spitzner, the server is here to "*determine if there is any unauthorized activity happening within the DMZ"*. Can we reasonably call this machine a security resource? Furthermore, what happens if it logs nothing? According to lance Spitzner "*if the system is never probed or attacked, then it has a little or no value"*. It is not a honeypot anymore in this case. These contradictions are mentioned by L. Spitzner, on page 41 where he writes:

"Honeypots are a highly flexible tool that can be applied to a variety of different situations. This is why the definition may at first seem vague."

This definition, even vague, is useful though as it offers a good feeling of what honeypots are. However our objective is a complete honeypot presentation. So we need to know exactly what can be called a honeypot. Consequently we present other definitions that can be found in academic as well as commercial papers, even if the previous one is the most widespread.

2.1.3 Reto Baumann's definition

Reto Baumann, a Swiss engineer, discusses in [BauPla02] Lance Spitzner's interpretation. His definition slightly differs from the previous one:

"A honeypot is a resource which pretends to be a real target. A honeypot is expected to be attacked or compromised. The main goals are the distraction of an attacker and the gain of information about an attack and the attacker." [BauPla02]

Thus, a honeypot is here a resource that behaves as a real target but knows that it is not (according to the definition given by the Cambridge Dictionary of the verb *'to pretend'*). Here too we consider two examples to assess the validity of that definition:

- A firewall which logs all traffic is not a honeypot because it does not feign to be a real target. Both Baumann and Spitzner seem to agree on this one.
- In the case of the unused server that passively waits in the DMZ, the question is more delicate. The machine is a real server (Microsoft Exchange) but is not a target. It is not used anymore and there is no reason *a priori* that the machine becomes a target. Furthermore how could it decide to be a target? A target is "*an object fired at during shooting practice*" (Cambridge Dictionary). The server becomes a target if attacked. But the decision does not come from its side. The expression 'to pretend to be real target' is very ambiguous. Does it stand for any object that *can eventually become a hacker target* or an object that *once attacked is not the expected target.*

We are unable to conclude in the second example if it is, or not, a honeypot deployment. The server does nothing but logging connection attempts. It has no pretence. On the contrary we would classify it as a non-honeypot deployment as the definition is not verified. It is surprising to define as non-honeypot the first honeypot illustration given by Lance Spitzner.

Furthermore Reto Baumann introduces two new restrictions in the definition. To qualify as a honeypot, the resource must fulfill the following goals: i) Distraction of the attacker and ii) Gain of information about the attack and the attacker. Is it correct to say that distraction is the first honeypot motivation? It is worth noting that "distraction" is not even mentioned in Lance Spitzner's first chapter called 'History and Definition of Honeypots'.

2.1.4 Definition of SearchWebservices.com

Many documents refer to Lance Spitzner's definition. Some of them adapt it to a more restrictive usage. This is the case of searchWebservices.com, a commercial portal on IT services. They suggest this one [Web03]:

"A honeypot is a computer system on the Internet that is expressly set up to attract and "trap" people who attempt to penetrate other people's computer systems. (This includes the hacker, cracker, and script kiddy.) Maintaining a honey pot is said to require a considerable amount of attention and may offer as its highest value nothing more than a learning experience (that is, you may not catch any hackers)." [Web03]

This definition is quite restrictive. First and foremost honeypots are not reserved to the Internet usage. They can be implemented to reveal internal attacks. In addition some of them are not 'expressively set up to attract'. One simple example consists in putting a basic Honeyd machine (see [PoDac] for more information on Honeyd) into the DMZ. It is likely to be scanned and attacked but it is not "expressly set op to attract people".

2.1.5 Barnett's definitions

The University of Wisconsin-Platteville (http://www.uwplatt.edu/) as well as R.C. Barnett from the sourceforge.net web site ([Sour03]) mentions the following definition:

"An Internet-attached server that acts as a decoy, luring in potential hackers in order to study their activities and monitor how they are able to break into a system. Honeypots are designed to mimic systems that an intruder would like to break into but limit the intruder from having access to an entire network. If a honeypot is successful, the intruder will have no idea that s/he is being tricked and monitored." [Sour03]

Here, other restrictions are expressed, like the fact that a honeypot may not be a real 'server' offering some real service on the net. Indeed, a server is a computer/device which provides information or services to computers on a network. It cannot be a decoy. Again, this differs from Lance Spitzner's view as explained before.

Furthermore, honeypots are not necessarily designed to *mimic systems that an intruder would like to break into*. If we consider the second example that is invoked by Lance Spitzner, an old and unused server that is left on the DMZ to collect some traffic is a honeypot. It does not *mimic* anything.

2.1.6 Definitions of the SANS institute

The last definition we propose is suggested by the SANS Institute [Sans03]. It appears in an article written by Michael Sink in April 2001: "The use of Honeypots and Packet Sniffers for Intrusion Detection" [Sink01]:

"Within the realm of computer security, a honeypot is a computer system designed to capture all traffic and activity directed to the system. While honeypots can be set up to perform simple network services in conjunction with capturing network traffic, most are designed strictly as a "lure" for would-be attackers. Honeypots differ from regular network systems in that considerably greater emphasis is placed on logging all activity to the site, either by the honeypot itself or through the use of a network/packet sniffer. A honeypot is designed to look like something an intruder can attack to gain access to a given system." [Sink01]

The problem with this definition is that it is rather verbose and vague. It characterizes honeypots from "*regular network systems"* as those that place "*considerably greater emphasis on logging all activity*". But many *systems* can be designed to capture traffic and activities. What does '*greater'* exactly mean? A firewall which logs connections or an application that stores history may be considered as honeypot. The definition is very broad as many computer systems are collecting activities and traffic directed to them.

2.1.7 Discussion

We have presented the five most common definitions of a Honeypot. The list is obviously non-exhaustive but it leads to a disappointing conclusion that none of them seems fully satisfactory. They do not really offer a precise, concise and non ambiguous frame to decide what is, or not, a honeypot. We consider, in the next Subsection, the definitions that exist for another term: Honeynet.

2.2 Honeynet definitions

2.2.1 Introduction

Honeynets are defined as specific instances of honeypots ([Honey1], [Honey2], [honey3]), but Won-Seok Lee offers in his course slides a definition which implies that a honeynet consists in a network of multiple systems:

"Honeynet is nothing more than a high-involvement Honeypot within which risks and vulnerabilities are the same that exist in many organizations today. It is not a single system but a network of multiple systems" [Lee02].

2.2.2 Lance Spitzner's definition

The definition presented by Lance Spitzner is more accurate and it is the one which is generally used while referring to *Honeynets* [Honey1]:

"Honeynets represent the extreme of research honeypots. They are high interaction honeypots, which allow learning a great deal; however they also have the highest level of risk. Their primary value lies in research, gaining information on threats that exist in the Internet community today. A Honeynet is a network of production systems. Unlike many of the honeypots, nothing is emulated. Little or no modifications are made to the honeypots. This gives the attackers a full range of systems, applications, and functionality to attack. From this it can be learnt a great deal, not only their tools and tactics, but their methods of communication, group organization, and motives."

To sum up a Honeynet is actually a network made up of real systems designed to be hacked. Some architectural Honeynet models have been suggested these few months

([GenH03], [Hone02]). Those of Lance Spitzner and the Florida Honeynet Project team have received a lot of attention and many security groups are testing them around the world. A concrete outcome of that work is the creation of the *Honeynet Research Alliance* (http://project.honeynet.org/alliance/) which is a forum dedicated to "*share ideas, experiences and findings, helping to develop Honeynet research*".

However, the definition remains unclear. Solutions called GenI & GenII Honeynets (described in [PoDab]) are based on virtual networks and virtual machines as some honeypots do. Thus, according to the previous definition they can not be called honeynets since the condition "*unlike honeypots, nothing is emulated*" is not fulfilled by any of them.

2.3 Honeytokens definition

2.3.1 Introduction

Another concept appeared during the beginning of 2003, called 'honeytoken'. This word has mainly been used on the *honeynet* mailing list. A summary of their discussion is presented in [Focus03].

In fact, the idea is not new and was already suggested in many papers such as [Stol88]. Their definition is however presented in the following paragraph to get an overview of the basic concept hidden behind this word.

2.3.2 Lance Spitzner's definition

One of the greatest misconceptions of honeypots is that they have to be a computer, or some physical resource for the attacker to interact with. While this is the traditional manifestation of honeypots, as defined by lance Spitzner, it is not the only one. The definition he gives does not state a honeypot has to be a computer; merely that it is a resource that bad guys should interact with.

That is exactly what a honeytoken is: a honeypot which is not a computer. Instead it can be any type of digital entity. A honeytoken can be a credit card number, Excel spreadsheet, PowerPoint presentation, a database entry, or even a bogus login. Honeytokens come in many shapes or sizes but they all share the same concept: a digital

or information system resource whose value lies in the unauthorized use of that resource. Just as a honeypot computer has no authorized value, no honeytoken has any authorized use.

The concept of honeytokens is not new, as Lance Spitzner notices in [Spit03]. This concept is as old as security itself. For example some map-making companies insert bogus cities or roads into their maps to determine if competitors are selling copied versions of their own maps. In Cliff Stoll's book "The Cuckoo's Egg," he deploys digital files to track and monitor a German hacker. While the concept is not new, the term is.

A honeytoken is just like a honeypot. We should even say that a honeytoken *is* a particular honeypot (as honeynets are). Any interaction with a honeytoken most likely represents unauthorized or malicious activity. A classic example of how a honeytoken could work is the "John F. Kennedy" medical records example [Focus03]. U.S. hospitals are required to enforce patient privacy and only certain authorized people have access to patient data (such as doctors, nurses, etc). A honeytoken could be used for such a purpose. A bogus medical record called "John F. Kennedy" is created and loaded into the database. This medical record has no true value because there is no real patient with that name. Instead, the record is a honeytoken, an entity that has no authorized use. If any employee is looking for interesting patient data, this record will definitely stand out. If someone accesses it they have most likely violated the system's usage policy.

3. Existing Classifications

3.1 Lure/Defend/Study Classification

This classification proposes to classify tools based on one of their objectives. Indeed, three are recurrent in literature: to lure, to defend or/and to study.

3.1.1 Lure

In this case, the honeypot is used as a security measure. It aims at getting all hacker’s attention on the honeypot and not on the real system. Damages would be less [BauPla02].

Honeypots can be used as decoys to trick and confuse hackers into finding vital information.

They are many methods to apply this security technique and the honeypot can be used:

- To waste attacker time. Sticky connections, long responses or endless data transfers are solutions to keep the attacker attention focused on the honeypot [Lab03].
- To route/filter incoming requests. If packets are suspicious and dangerous they may be forwarded to the honeypot instead of the production server. The attacker interacts with the honeypot and so he is prevented to compromise the real system [Bait03].
- To consume the resources of the attacker by sending malformed packets or by answering abnormally [THP03].

3.1.2 Defend

Honeypots may be implemented as defensive tools. There are two prevalent methods: *Deception* and *Intimidation* [Coh01]. In [Coh88], examples are presented of the use of deception in military campaigns dating back thousand of years. With the same approach Fred Cohen presents in [Coh01] a classification of defense for information systems, in which one of those defenses is deception:

"Typical deceptions include concealment, camouflage, false and planted information, reuses, displays, demonstrations, feints, lies, and insight (Dunnigan, 1995). Examples include facades used to misdirect attackers as to the content of a system, false claims that a facility or system is watched by law enforcement authorities, and Trojan horses planted in software that is downloaded from a site. Deceptions are one of the most interesting areas of information protection (...)."[Coh88]

Based on this military defense Fred Cohen suggests in [Coh88] to transform honeypots into *deception tools*. We notice in this definition that *intimidation* is already mentioned in the *deception* concept (*"false claims that a facility or system is watched by law enforcement authorities"*). To deceive, a honeypot must provide realistic responses to requests so that an attacker does not suspect it is a trap. To intimidate, a honeypot might advertise on one unauthorized deception port a default banner such as "Honeypot in Use"

which will increase the intruder's fear of being caught in a manner similar to posting a security alarm sign in the physical world.

Other defensive methods based on honeypot technologies are emerging. For instance honeypots can be set up next to firewalls. All packets coming into such a honeypot are by definition suspicious. The honeypot can be used to automatically update the firewall blacklist. Such an automatic countermeasure must be carefully designed to prevent misusers from using it to launch denial-of-service attacks.

3.1.3 Study

One major motivation for building honeypots is to learn hacker's techniques and tools. That way system administrators and agents can improve their forensics and defense techniques. The honeypot is deployed as a sensor. Its mission is to gather information about attack trends, attacker tools and network activities. It can provide vital attack signature information to other security tools, especially intrusion detection systems and firewalls. The results of the analysis of its data can be used to tune the sensors and, hence, decrease the number of false positives and/or increase the alert level of some alarms.

3.1.4 Discussion

This classification is interesting. Many honeypot applications are suggested. However, since these three objectives are not mutually exclusive, they can not be used as a basis to build taxonomy for honeypots.

3.2 Production/Research classification

Lance Spitzner explains in [Spit02] that honeypots can be classified according to their use. They add value to security and reduce the organization's overall risk but in different ways. Following this, Martin Roesch (creator of the Snort Intrusion Detection System) defines in [Honey2] two categories of honeypots:

Production Honeypots *are systems that help mitigate risk in the organization or environment. They provide specific value to securing systems and networks by preventing (deception/deterrence to have attackers waste time and resources for instance), detecting*

(Intrusion Detection Systems help by reducing false positives, false negatives and data aggregation) and responding (honeypots can easily be taken offline for further analysis). Honeypots can be a powerful tool to complement the reaction capabilities of a network administrator by capturing precise details on how the attacker got in and what they did.

Research Honeypots *give a platform to study cyber threats and fill the lack of information on the enemy. This becomes an educational tool.*

This classification is simpler than the previous one. We are tempted to make an analogy between these two:

2. Production Honeypots: equivalent to honeypots design to catch or to defend
3. Research Honeypots: equivalent to honeypots design to study

But there is still no way to classify honeypot solutions with it. Moreover many administrators who are in charge of the security of *production* systems use honeypots to *study* the network activity. This simple example shows that a honeypot can easily belong to both categories.

3.3 Level of interaction classification

Lance Spitzner introduces another honeypot classification by level of interaction [Spit02]. How to architect a honeypot depends on the objectives it has to fulfill. A complex honeypot can be built to give the attacker a complete operating system with which to interact. However, for detecting any unauthorized activity such as scanning, a simpler honeypot which merely emulates a variety of services in operation can be built. And when capturing the latest worm for analysis is the main requirement, then a customized honeypot with the intelligence to interact with the worm and capture the worm activity should be more appropriate.

A honeypot can offer many different functionalities and the level of interaction they offer to attackers is vital. It is supposed to give a granular scale with which to measure and compare honeypots. The more a honeypot can do and the more an attacker can do to a honeypot, the greater the information that can be derived from it. However, by the same token, the more an attacker can do to the honeypot, the more potential damage an attacker can do. Each level has its advantages and drawbacks which are briefly exposed in more details in the next three paragraphs.

3.3.1 Low-Interaction Honeypots

A low-interaction honeypot provides certain fake services [Baum02]. In a basic form, these services can only be implemented by having somebody listening on a specific port as illustrated in figure 1. Services are limited to listening ports. For example a simple Unix command like: "*netcat –l –p 80 > /log/honeypot/port_80.log"* could be used to listen on port 80 (HTTP) and log all incoming traffic to a log file. In such a way all incoming traffic can easily be recognized and stored. However, with such a simple solution it is not possible to catch communication of complex protocols. The honeypot cannot trace TCP connections for instance as it logs only the first connection requests without answering.

On a low-interaction protocol there is no real operating system target that an attacker can operate on. This will minimize the risk significantly because the complexity of an operating system is eliminated. On the other hand this is also a disadvantage. It is not possible to watch an attacker interacting with the operating system. A low-interaction honeypot is like a one-way connection as the honeypot is only listening [Baum02]. Its role is very passive and it does not alter any traffic. It is used to generate logs or alerts when incoming packets match their patterns.

We can classify as low-interaction honeypots tcpdump and tiny Honeypots for instance. They are described with more details in [PoDab].

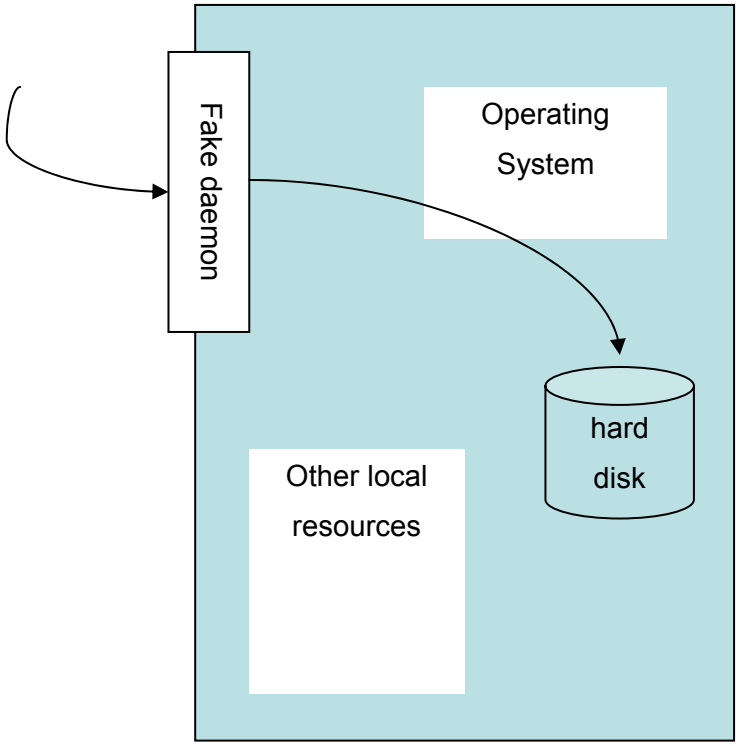

Figure 1: Low-Interaction honeypot (picture taken from [BauPla02])

3.3.2 Mid-Interaction Honeypots

A Mid-Interaction honeypot offers richer interaction capabilities but does not provide any real underlying operating system target as shown in figure 2. The fake daemons are more sophisticated and have deeper knowledge about the specific services they provide. Generally speaking the attacker gets a better illusion that a real operating system exists and he has more possibilities to interact and probe the system. Special care has to be taken for security checks as all developed fake daemons need to be as secure as possible (buffer overflow risk, etc). Furthermore the knowledge for developing such a system is very high as each protocol and service has to be understood in some depth. In existing implementations though, fake services are often limited to simple scripts. Honeyd, Specter and LaBrea are honeypot solutions that can be classified as mid-interaction honeypots (see [PoDa03b] for more information on these tools).

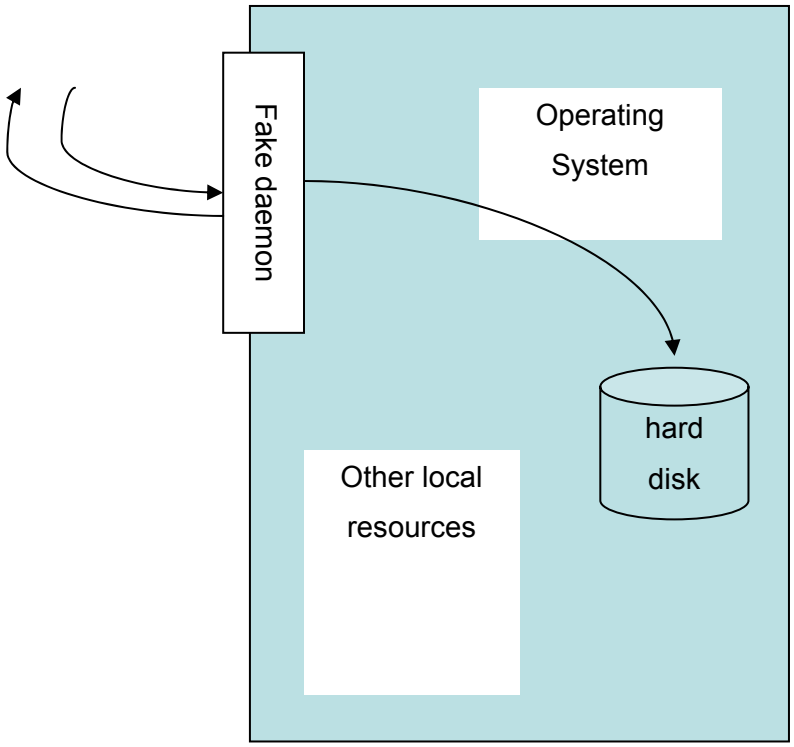

Figure 2: Mid-Interaction honeypot (picture taken from BauPla02])

3.3.3 High-Interaction Honeypots

A High-Interaction honeypot has a real underlying operating system to offer to the attacker as illustrated in figure 3. This leads to much higher risk as the complexity increases. On the other hand the possibilities to gather information, the possible attacks and the attractiveness increase a lot. The goal of a hacker will most likely be to get as many privileges as possible on the target machine. By providing a full operating system to the attacker we offer him the possibility to upload and install new services/applications. This implies that the system must be under surveillance all the time. All actions can, and must, be recorded and analyzed to gather more information about the blackhat community. R. Baumann and C. Plattner write that it is the main goal of a high-interaction honeypot and it legitimates the higher risk [BauPla02].

Lance Spitzner gives in the SecurityFocus forum ([Secu03]) more information of his classification and explains that: "*(his) perception of low interaction vs. high interaction is intent. With low interaction, we intend on limiting the attacker to only emulated services. With high interaction honeypots, we intend on giving attacker access to the full operating system. Both deployments require a real operating system. With low interaction, the emulated services do run on a real operating system, as in Tiny Honeypot, Specter, and Honeyd. However, the goal is to limit the attacker to interacting with just the emulated services and not give them access to the operating system.*"

We observe that Honeyd and Specter are classified as mid to high interaction honeypots. The reason is that these tools are highly configurable and their interaction level depends on their configuration. For instance, Honeyd services are emulated by small scripts. The more complex scripts are designed, the higher interaction the honeypot should manifest.

However, as George Bakos so eloquently put it, "*when dealing with blackhat communities, what is intended and what is gotten may be two different things.*" [THP03] (George Bakos is the Tiny Honeypot designer).

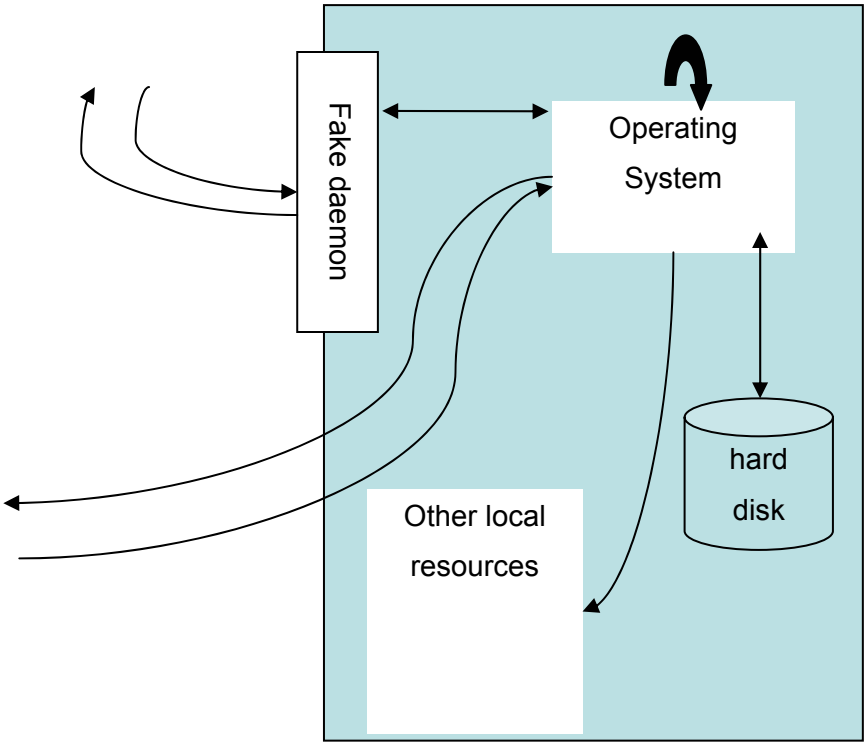

Figure 3: High-Interaction honeypot (figure taken from [BauPla02]

3.3.4 Summary

Each level of interaction has its advantages and disadvantages. The following table 1 extracted from [Spit02] summarizes what has been said so far. In the column entitled '*Work to Deploy and Maintain*', one finds the time required to run and maintain the honeypot. In the '*Knowledge to develop*' one, one sees the amount of pre-required knowledge to build a honeypot environment. The column '*Level of Risk*' column is an indicator of the risk run when implementing a honeypot into one system.

Table 1: Level Interaction and Honeypots

Level of Interaction	**Information gathering**	**Work to Deploy and Maintain**	**Compromise wished**	**Knowledge to develop**	**Level of risk**
Low	Connection Attempts	Easy	-	Low	Low
Medium	Requests	Involved	-	Low	Medium
High	All	Difficult	Yes	High	High

3.4 Discussion

We have presented in this paragraph three classifications. We have tried to use these definitions and classifications rigorously on a large number of 'honeypot' tools available today. The experience highlighted the non-usability of what had proposed so far. Many tools had to be defined as belonging to many categories. For instance, if we consider the 'interaction' classification, Honeyd or Specter may join the low or medium group depending on their configuration. They can simply listen to ports or directly interact with the intruder. Simply classifying honeypots into the class of *research-based* or *production-based* systems does not enable us to draw precise conclusions neither about the information a honeypot is able to gather nor about the reactions that it might have.

The terminology approximation is confirmed by many details and mailing list questions. Lance Spitzner modifies its classification in [Secu03] and drops the 'medium-interaction'

category. As he explains, both honeypot technologies and our understanding of them have dramatically changed. However classifications are not supposed to change each year. The scientific community must share a precise vocabulary in order to communicate efficiently and in a non ambiguous way.

4. Our Honeypot definition

4.1 Concept

In order to provide a more rigorous definition of what honeypots are, we propose to take advantage of well-defined concepts used by the dependability community. To that end, we reuse the definitions proposed b several contributors within the European MAFTIA project. [PoSt03, page 32] introduces the notion of attack, vulnerability and intrusions as follows:

Attack – a malicious interaction *fault,* through which an attacker aims to deliberately violate one or more security properties; an *intrusion* attempt.
Vulnerability – a fault created during development of the system, or during operation, that could be exploited to create an *intrusion*.
Intrusion – a *malicious*, externally-induced *fault* resulting from an *attack* that has been successful in exploiting *vulnerability.*

Vulnerabilities are the primordial faults existing inside the components, in particular design or configuration faults (e.g., coding faults allowing program stack overflow, files with root setuid in Unix, naïve passwords, unprotected TCP/IP ports). Note, however, that a successful attacker might purposely introduce vulnerability (in the form of malicious logic such as T*rojan horses, trapdoors, virus, worms* and so on) as a step in his overall plan of attack.

Typical examples of intrusions interpreted in terms of vulnerabilities and attacks are:

1. An outsider penetrating a system by guessing a user password: the vulnerability lies in the configuration of the system, with a poor choice of password (too short, or susceptible to a dictionary attack).
2. An insider abusing his privilege (i.e., a misfeasance): the vulnerability lies in the specification or the design of the (socio-technical) system (violation of the principle of least privilege, inadequate vetting of key personnel).
3. An outsider using "social engineering", e.g., bribery, to cause an insider to carry out a misfeasance on his behalf: the vulnerability is the presence of a bribable insider, which in turn is due to inadequate design of the (socio-technical) system (inadequate vetting of key personnel).

An intrusion is a security policy violation. It results from a vulnerability and an attack. But attacks are not directly observable and some conditions must be satisfied to observe an intrusion. More information is available in [PoSt03].

Based on this strong terminology, it is easier to build an intuitive and rigorous honeypot definition:

A honeypot consists in an environment where vulnerabilities have been deliberately introduced in order to observe intrusions.

A system can be attacked (either from the outside or the inside) without any degree of success. In this case, the attack exists, but the mechanisms that protect the system or resource targeted by the attack are effective enough to prevent any intrusion. An attack is thus an *intrusion attempt* and an intrusion results from an attack that has been (at least partially) successful. So a honeypot cannot guarantee that intrusions will be observed. It helps observing intrusions if any.
Why are intrusions usually difficult to observe?

In a classic production system all is done to prevent intrusions. That is the traditional approach of security tools such as firewalls, access policies, antivirus, and so on. They

protect the system from specific attacks and they reduce (in theory) the system vulnerabilities.

A firewall is considered a first line of defense in protecting private information for instance. Security administrators protect a production network from intrusion and degradation, and honeypots operate on the opposite. We want to observe intrusions. So the idea consists in injecting vulnerabilities in the system. These *sleeping vulnerabilities* lead to intrusion if correctly exploited by attacks.
Two examples can be:

- An open port on the server where all connection attempts are logged (a port that is closed in theory, the *vulnerability*).
- A firewall which logs refused accesses. Vulnerabilities are these (theoretically) forbidden accesses.

What makes the new solutions presented in [PoDab] very attractive is their relative independency. They are not correlated to the existing system and they do not interfere with the production components. Generally speaking they offer a larger observation field and an all-in-one way to observe, collect and analyze data.

4.2 Structure

The honeypot consists typically in four major activities:

- *Building the environment that (eases) the observation*

  Honeypots create environments which offer the same conditions and behaviors than real working systems. They can be distinguished by their *OS emulation methods,* their *services* and *network simulation methods* as well as their methods to provide *resources and data* ([Spit03], [Honey4], [Honey5]). These simulation methods are not mandatory however to build an observation environment. A honeypot is a solution that should come beside the existing system and that should have limited impact on it.

- *Collecting observation data*

The purpose of Data Collection is to log as much data on the attacker's activity as possible (or wanted). The key is collecting information at many layers. The Honeynet Project has identified three critical layers which are *firewall logs, network traffic* and *system activity* [Spit02.]. It is not mandatory however to implement these three data collection layers.

- *Analyzing information*

Honeypots are collecting some information either by themselves or by complementary tools. The information analysis follows the same principle: some honeypots directly integrate methods in their architecture while others require additional tools.

- *Taking appropriate decisions*

This component is optional and few honeypots are currently using one.

Behavior on detection, or responses, are actions taken by the environment as a result of a generated event. We distinguish two kinds of responses: reactions which concern the honeypot itself and reactions that tend to modify the external system. Reactions on the system may be considered as *active* or *passive*. There are active reactions when the environment is able to notify the existing security system that errors have been observed on the honeypot platform. Reactions on the honeypot environment are classified into two categories: *forward recovery* and *backward recovery* (see [Pow95] for more details on these core dependability concepts).

Building the environment is not easy and many tools aim at simplifying this task. Each month new ones are found on the internet. A comparative survey is given in [PoDa03b] to describe some of them and to explain their main features.

5. Conclusion

This existing terminology is obviously not clear enough, as it is shown in section 2. Some groups have ‘adopted’ these terms but are unable to provide any clear definition. In order to provide a more rigorous definition of what honeypots are, we propose to take advantage of well defined concepts used by the dependability community. This new terminology is briefly explained in section 4 and will be presented with more details in another technical report.

Many definitions are proposed in different mailing lists. We hope this work will help converging on a unanimous concept. Research is built from rigorous definitions and the honeypot research future depends on them too.

5. Bibliography

[AbKK02] H. AbdelallahElhadj,, H. M. Khelalfa and H. M. Kortebi, "An experimental sniffer detector: SnifferWall", *SEcurité des Communications sur Internet Worshop (SECI'02)*, Tunisia, Sept. 2002:.http://www.lsv.ens-cachan.fr/~goubault/SECI-02/Final/actes-seci02/pdf/008-Abdelallahelhadj.pdf

[Bait03] *Bait N Switch Honeypot* from Team Violating site: http://violating.us/projects/baitnswitch/

[Baum02] R.Baumann. "*White Paper: Honeypots*". February 2002. Available on line: http://security.rbaumann.net/download/whitepaper.pdf

[BauPla02] R. Baumann, C. Plattern. *Honeypots, diploma thesis*. Feb. 2002

[Bell92] S. Bellovin, "There Be Dragons", *Proc. of the Third Usenix Security Symphosium*, Baltimore MD. Sept. 1992. Available on line: http://www.research.att.com/~smb/papers/dragon.pdf

[Bell93] S. M. Bellovin, "Packets Found on an Internet", *Computer Communications Review* 23:3, pp. 26-31, July 1993. Available on line: http://www.research.att.com/~smb/papers/packets.pdf

[Ches92] B. Cheswick, "An evening with Berferd in which a cracker is lured, endured and studied", *Proc Winter USENIX Conference*, San Francisco, Jan 20, 1992.

[Coh88] F. Cohen, "*Deception and Perception management in Cyber-Terrorism*". 1988. Paper available at: http://www.securityfocus.com/library/1278/scoreit

[Coh99] F. Cohen, "A Mathematical Structure of Simple Defensive Network Deceptions", 1999.

[Coh01] F. Cohen, "*A Framework for Deception*", IFIP-TC11, 'Computers and Security', 2001.

[Coh02] F. Cohen, "method and Apparatus Providing Deception and/or Altered Operation in Information Systems", 2002.

[Cole01] E. Cole. *Hackers Beware*. New Riders Publishing 2001.

[DDW99] M. Dacier, H. Debar, A. Wespi, "*Towards a Taxonomy of Intrusion-Detection Systems*", Computer Networks, 31 (8), pp. 805-22, 1999.

[Focus03] Lance Spizner, "*Honeytokens: the Other Honeypot*", Security Focus information, July 2003.

[GenH03] "*Know Your Enemy: GenII Honeynets Easier to deploy, harder to detect, safer to maintain*", by the honeynet Project members, June 2003. Available on line: http://project.honeynet.org/papers/gen2/

[GrDa98] S. Grundschober, M. Dacier, "Design and Implementation of a Sniffer Detector", *Recent Advances on Intrusion Detection Workshop (RAID98)*, 1998.www.raid-symposium.org/raid98/

[Grun98] S. Grundschober, "*Sniffer Detector Report*", Master Thesis, Eurecom Institute , June 1998, 50 pages, ref. Eurecom : CE-98/IBM/GRUN - Document number: 1914. Available on line: http://www.eurecom.fr/~nsteam/Papers/grundschober98.ps

[Hone02] "*Know Your Enemy: Learning with User-Mode Linux Building Virtual Honeynets using UML*", Honeynet Project, December 2002. Available on line: http://www.honeynet.org/papers/uml/

[Honey1] Honeynet Project, "*Know Your Enemy: Defining Virtual Honeynets*". Sep. 2002. Available on line at: http://project.honeynet.org/papers/index.html

[Honey2] Honeynet Project, "*Know Your Enemy: Part I*". 2001. Available on line at: http://project.honeynet.org/papers/index.html

[Honey3] Honeynet Project, "*Know Your Enemy: Part II*". 2001. Available on line at: http://project.honeynet.org/papers/index.html

[Honey4] Honeynet Project, "*Know Your Enemy: Motives*". 2002. Available on line at: http://project.honeynet.org/papers/index.html

[Honey5] Honeynet Project, "*Know Your Enemy: A Forensic Analysis*". 2002. Available on line at: http://project.honeynet.org/papers/index.html

[Lab03] LaBrea-The Tarpit home page: http://hackbusters.net/LaBrea.May2002

[Lee02] Won-Seok Lee, "*Honeypots*". Ajou University Info. Comm. & Security lab. Available on line at: http://cesec.ajou.ac.kr/board/include/2001security/ files/phpQ2G4Q6/Honeypot.ppt

[PoDab] F. Pouget, M. Dacier, *"Honeypot, Honeynet: A comparative survey"*. Eurecom Research Report RR-03-082. August 2003.

[PoDac] F. Pouget, M. Dacier, *"Honeypot platform: Experimental Report"*. Eurecom Research Report RR-03-083. September 2003.

[PoSt03] "*Conceptual Model and Architecture of MAFTIA"*, D. Powell et R. Stroud (Editors), MAFTIA Project (IST-1999-11583), Deliverable D21, January 2003; available on line at www.maftia.org.

[Pow95] D. Powell, "*failure Mode Assumptions and Assumption Coverage: A Revised Version*", LAAS-CNRS, France. Research Report 91462.

[Sans03] *SANS Institute* home page: http://ww.sans.org

[Secu03] SecurityFocus mailing lists Archives available on line : http://www.securityfocus.com/archive

[Seif02] K. Seifried, "*Honeypotting with VMware – basics*". www.seifried.org/security/ids/20020107-honeypot-vmware-basics.html

[Sink01] M. Sink, « *The Use of Honeypots and packet Sniffers for Intrusion Detection*", Indiana University of Pennsylvania, April 2001. Available on line: http://www.lib.iup.edu/comscisec/SANSpapers/msink.htm

[Sour03] Sourceforge home page: http://honeypots.sourceforge.net/

[Spit02] L. Spitzner, "*Honeypots: Tracking Hackers*". Addislon-Wesley, ISBN from-321-10895-7, 2002.

[Spit03] L. Spitzner, "*Honeytokens: The Other Honeypot*", 2003. Available on line at: www.securityfocus.com/infocus/1713

[Stol88] C. Stoll, *"Stalking the Wiley Hacker"*, Communications of the ACM, Vol. 31 No5. May 1988.

[THP03] Tiny Honeypot home page: http://www.alpinista.org/thp/

[Web03] Webservices commercial IT portal http://www.webservices.org/