A reversed approach to security and software engineering - Introduction

Alexandre Dulaunoy

8th January 2005

Contents

`http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?hacker+ethic`

1 Security and software : a long antinomy

There is a very strange paradox between security and software engineering. It's very complex to build secure software that can resists to attacks or having a stable operation in a computer environment. The issue came from various source. One origin is the complexity surrounding software and its operation. A software is always part of various components done by other programmers and rely on them at different stages.

For example, a HTTP server is a using the underlying operating system and can interact with other external programs. The HTTP server can use system libraries but also built-in libraries. A simple operation like giving a page from a specific HTTP request involves a multitude of software components via different kind of interface and methods.

The analogy of babel tower (Software Architectures from an Escher Perspective - `http://trese.cs.utwente.nl/taosad/escher.htm`). It's an utopia to build secure software but we will take the ride to try to achieve this utopia. Another interesting point when doing software development, implementation, maintenance is the trusting that can you have with the system itself (Reflections on Trusting Trust, Communication of the ACM, August 1984 Volume 27 Number 8).

Another interesting analogy is between security in the physical world compared to the security in non-physical (digital) world. Why security in the physical world seems more easy and secure? What are the differences and the common ground ? Is it possible to gain the same level of security in the non-physical world ?

An interesting point of view is a paper (M. Blaze. "Safe cracking for the Computer Scientist." U. Penn CIS Department Technical Report.) made by Matt Blaze making a comparison between Safe cracking and "Computer Science". You'll see that physical security can be also very very insecure like any complex computing systems.

2 Vulnerability, Attack and attackers

The definition of a vulnerability in "computer science" refers to a weakness in a system but also any opening in a system due to the system itself or any other causes in its environment (e.g. a vulnerability can be created in a system by setting up the wrong permission attribution on a file used by the system).

Often vulnerabilities are closed in result of a security incident but not often by proactive method of auditing during the development process. This is evolving (check the various code checker tools) but it's only at the early stages. Vulnerabilities must be evaluated when designing not when distributing the software/system.

The definition of an attack is often the act to make a malicious act on a system. A system can be composed of softwares, protocols, algorithms, data structures, physical components... in the definition we don't limit ourself to specific system. This is also true to the attack itself, there are various kind of attacks including physical. An attack can use one or more vulnerabilities in order to make a malicious act and reach his goals.

The objectives of an attack are very variables and can be of any sort including using a system to start other attacks, stealing bandwidth of the network,... In order to full fit his/its objectives, an attacker can make different actions/activities on the attacked system (e.g. removing logging entries, installing back-door, ...).

3 Honeynets or how to catch attacks

Theory is very interesting in computer security but it's only theory. Real data, real attacks, real attackers are more interesting than any theoretical security data and examples. But how can we catch attackers and real attacks instead of elaborating theories ? We need to create tools in order to gain visibility in the real world of the attackers. One of the most important tool is the honeynet. The purpose is to learn as much as possible from the attacks but also the define the profile and the goals of the attackers... Honeynets are not perfect but can be an effective research security tool.

3.1 History

We can say the first practical computer honeynet was created by Clifford Stoll when he baited an Intruder by creating fictitious electronic document about secret defense (Stalking The Wily Hacker, Communication of the ACM May 1988 Volume 31 Number 5). The process was also used to some extend by Bill Cheswick and described in a paper called "An Evening with Berferd". Various other works were made on the subject but the first large project of honeynet was founded by various technical guys who want to learn about the attacks and the attackers. The "Honeynet Project" (`http://www.honeynet.org/`) was founded in 1999 in order to define and provide resources to create honeynet. The concept was not very new but they conceptually made the terminology and the new technical approach used in the honeynet technologies. A large number of people participate to the "Honeynet Project" due the approach taken when working in community (Known Your Ennemy, The Honeynet Project, Addison Wesley) : Keep It Small, Make It Fun, Communicate... The first honeynet were very simple including a single computer that act as vulnerable system.

3.2 Definition

There is no single definition of what is a honeynet. The term is covering a bunch of different type of systems. The objective of the system is to be compromised in order to gather as much as possible information from the its unauthorized or illicit use. A honeynet is not limited to a specific technology or method, a lot of methods can be considered honeynet-like (e.g. Including a false door with no trespassing in a building could be considered as a honeynet-like technology). A honeynet/pot has no production value and is generally running false production service.

3.3 Advantages/Disadvantages

What are the main advantages of running honeynets :

- Honeypots are very flexible. Compared to a production system you can customize the system to fit fully its task to collect information. That would be very difficult on existing production system.
- Learning. You can learn a lot from data collected by compromised honeynets. New attacks could be discovered and analyzed in order to make new protection measures against them
- Playground for security definition. Collecting the information from a honeynet, you can reduce for example the false positive from a NIDS.
- Honeynet can works with encrypted data or new network infrastructure. In production environment, you want to keep a high of confidentiality for data crossing the network. In Honeynet, you can implement solutions to intercept encrypted communication.

What are the main disadvantages of running honeynets :

- Associated risks of running honeynets. A honeynet/honeypot system is also a complex system. They are vulnerabilities for the honeynet itself that extend the risk of the honeynet to be reuse to make for example new attacks.
- Honeynets/honeypots view is very limited. They see only the attacks they got. It's not a global view but only a view of vulnerabilities accessible for the honeynets. The view can be extended by distributed honeynets. The limited view permits also to focus on the attack itself and limit the interference generated by the environment.

3.4 Use of Honeynets/Honeypots

Two major usages exist for honeynets/honeypots one is to use it as a companion to an existing production network (e.g. attackers can waste their times by attacking honeynets instead of the real production system).Production honeynets can protect a production network in order to detect earlier attacks... but the risk can be important if you plan to implement a high interaction honeynet in a corporate network. A lot of network administrator got no time for managing their networks including honeynets can generate new problems and more works. Great care must be taken for implementing honeynets on real environment.

The second major usage is the research purpose. This purpose is to fill the gap of information missing by security professional on real attacks and attackers. The research honeynet is often easier to manage due to its dedicated nature without any production constraints.

3.5 Type of Honeynets/Honeypots

There are two big categories of honeypots. The two categories are built based on the level interaction that the attacker can have the honeypot.

3.5.1 low interaction

Low interaction honeypot often used an emulated approach to software. It's not the real software running but a small software making an emulation of the possible interaction between the service and the attacker.

Some example of low interaction honeypot software :

- NFR BackOfficer Friendly. It's a simple service for WIN32 preconfigured to emulate well-known services (like FTP, Telnet and alike). The software is very limited but can be used a starting point.
- Specter. It's very similar NFR BackOfficer Friendly but more option are available and can emulate false TCP/IP stack to act like a specific Operating System.
- KFSensor. A WIN32 only proprietary
- Honeyd. It's a free software (`http://www.honeyd.org/`) written by Niels Provos can be easily modified and extend to support other Operating System and network services. An interesting feature is the basic support for emulating IP routing, you can build easily virtual network for evaluation.

A sample configuration for Honeyd :

```
create default
set default personality "Linux 2.2.14"
set default default tcp action block
add default udp port 53 "./scripts/dnstool.py"
```

3.5.2 high-interaction

There is a major difference between low interaction honeypot and high-interaction as they provide a complete application including operating system. High-interaction are not emulating software but are running the software that could be vulnerable to attacks. The advantage is clear, the attack can be a classical attack and we can collect all the interaction. The interaction can be the keystroke entered by the attackers, the communication done with external parties, compromised software installed,... Like that you can learn the maximum from the attackers and making deduction based on the data collected. The major issue with high-interaction honeypot is the major level of complexity to manage and the various possible risks to mitigate. Building high-interaction honeypot/honeynet is difficult and time consuming. Some proprietary product exist for building high-interaction honeynet like the Decoy Server (a product from Symantec). But high-interaction honeynets are often customized honeynets infrastructure with specific and dedicated tools.

3.6 Evolution of Honeynets

Here is the evolution of design of the honeynets from the "Honeynet Project" but also some other project using honeynet-like technologies. The projects are all using different method for data capture in order to log all the activities of the attackers in the honeynet.

3.6.1 Generation I

3.6.2 Generation II

3.6.3 Virtual Honeynets

3.6.4 Distributed Honeynets

3.6.5 Hardware Honeynets

3.7 Legal aspects

There are a lot of legal incertitude around Honeynets and their usage.When building and installing Honeynets, you must take great care of the various security implication of running such kind of systems.

- The honeynet can aid of an attack. Various countries have laws

- punishing non-protected computer infrastructure in protection.
- The honeynet can be used to attack third parties.
- The question of privacy.

4 Forensic analysis

Warning : The images given for this course include malicious code on it (including viruses) and maybe running malicious processes. Great care must be taken to ensure that you do not infect your own system and that the malicious software does not make any attempt to contact external networks. It is recommended that you perform analysis on a dedicated network with proper security.

Data and forensic analysis is the major important part of learning attacks from crime scene.A honeypot is a crime scene where any digital crime can be done. Capturing data of the attacks is one small part of Honeynets but the analysis part is very important to get the maximum from your data collected in the Honeynets.

4.1 Best practices and golden rules for forensic analysis

- Don't work on the original data. Always make a working copy (without altering) of the data and keep the original in the safe place.
- Make documentation of all action made during the analysis. Investigation records must be kept in secure place and can be reuse to recover modified data during the investigation.
- Don't trust the system/data. Always (e.g. use external binaries) work with "trusted" components.
- Correlate various data source. In order to confirm the findings and enhance the various proofs.

4.2 Tools for forensic analysis

Various tools exist to help the coroner to dig into the scene of the crime.

4.2.1 Network tools

- Ethereal/tethereal (wiretap).
- tcpdump (libpcap).
- tcpreplay.
- Snort.

4.2.2 "File System" tools

- TCT (The Coroner's Toolkit)
- TASK
- Autopsy
- File
- Strings
- dd

4.2.3 Disassembler tools

- gdb
- objdump
- gdb
- (*)trace
- Fenris
- IDA pro (proprietary)