Using Virtal Machines to increase honeypot security

Eduardo Fernandes Piva[1], **Paulo Lício de Geus**[1]

[1]Laboratório de Administração e Segurança de Sistemas
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6176 – CEP 13084-971
Campinas – SP – Brazil

Abstract. *This work in progress discuss how virtual machines can be used to implement a data capture system for a honeynet. The main advantage of this approach is that the data capturing is done outside the honeypot kernel, making it practivally impossible to be detected and disabled by the intruder. By achiving this goal, a honeypot can be deployed in a safer environment, mitigating risks involved when using a honeynet.*

1. Introduction

Since the proposal of the concept in the early 90's, the honeypot has become popular among security professionals and is considered a tool that can be used against blackhats[The Honeynet Project 2003a, Spitzner 2002].

Since then, the technology applied to deploy honeypots continued to evolve. Such evolution is necessary to keep up with advances in the techniques developed by the blackhat community, designed to detect and disable a honeypot[Corey 2004b, Dornseif and Klein 2004, Corey 2004a], and so allowing a protection of the honeynet.

Currently, the state of the art in honeypot technology is based on the second-generation honeynets[The Honeynet Project 2003b] (also known as GenII honeynet). The GenII honeynet uses a tool that is capable of capturing data (such as keystrokes and downloaded files) inside the honeypot, even if the intruder is using an encrypted connection. This is feasible because the data is captured inside the Kernel of the operating system, after the data is decrypted. This tool is called Sebek[The Honeynet Project 2003c].

The GenII honeynet architecture has worked well since it was developed, but more and more the blackhat community is developing knowledge on how to detect, disable and even use a GenII honeynet as a starting point of an attack.

This work in progress describes a technique that is being implemented, through the use of virtual machines, that can decouple the data capturing system from the OS, preventing a blackhat from disabling and taking over the data capture system.

The data capture system used in this work is HECK, a tool developed by Martim Carbone[Carbone and de Geus 2004], which captures a large group of syscalls then Sebek does, and is able to capture all data that is modified in the honeypot filesystem by the intruder.

The next section of this paper summarizes how a GenII honeynet works and how it can be detected and disabled. Section 3. describes how this ongoing work uses HECK and virtual machines to achive its goal. Section 4. concludes and points to further developments.

2. GenII Honeynets

As defined by the Honeynet project, a GenII honeynet is made from a Data Capture system, a Data Control system and an Alert system. The Data Capture part is responsible for capturing data inside the honeypot, acting in the Operating System (OS) Kernel. The Data Capture System of a traditional GenII honeynet is implemented by Sebek.

Sebek works by intercepting some syscalls in the OS. During this interception, captured data is sent through the network to a storage server and then the original syscall is called. Figure 1 illustrates this functionality.

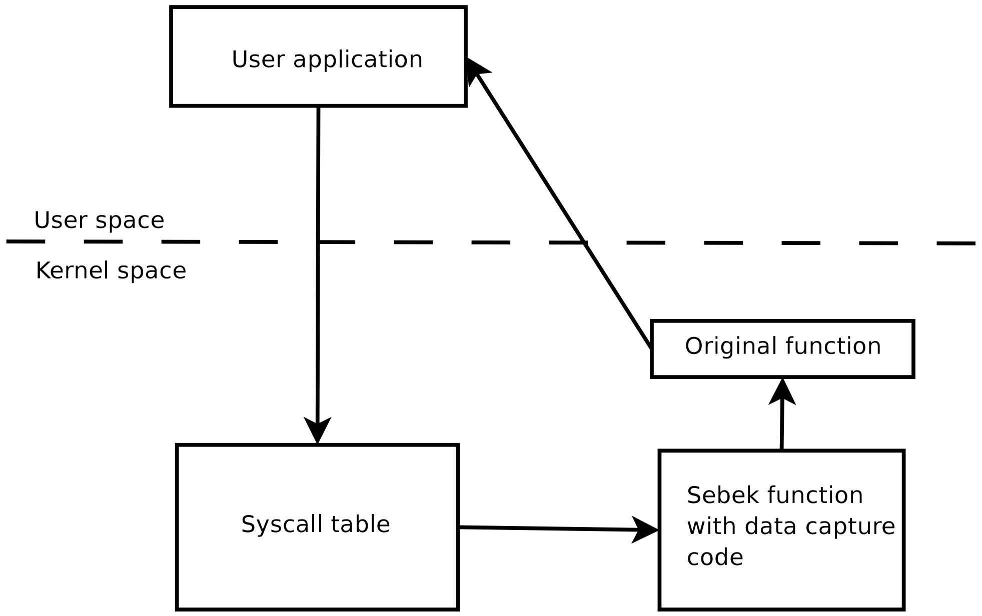

Figure 1. Diagram that shows how Sebek intercept a syscall.

Thanks to a modification done in the network driver through Sebek, data sent is invisible to sniffer programs. This is possible because data is filtered before the iptables hooks, which are used by sniffer programs, like those based on libpcap[1].

Although Sebek works, it can be detected and disabled. This is true because the intruder of the honeypot is supposed to become root of the machine, hence it will be able to control and inspect everything. A few examples are given throughout this section to elucidate how this can be done.

As a first example, suppose an attacker take over a honeypot, but does not manage to become root. Sebek may be detected by exploring the fact that all data is transmitted through the network. If the intruder forces the transmission of an unusual amount of data, for example by copying large files, a network instability will be created and will probably be detected by the intruder[Dornseif and Klein 2004].

If the data is being sent over a dedicated network, it is still feasible to detect the Sebek or any modification in the kernel. By inspecting the device `/dev/kmem` or even inspecting the system kernel through a kernel module, a honeypot system may be detected[Corey 2004b].

[1]http://libpcap.sf.net/

As previously noted by some authors, a modified syscall can be replaced by a legitimate syscall to disable the honeypot system[Dornseif and Klein 2004]. It is also possible to disable the system and attack the data collection system, without the honeynet administrator knowing this, because the data capture system will be disabled[Corey 2004a].

3. Virtual Machines and honeypots

Since an intruder can detect and disable the data capture system, mainly because it is running in the same machine that was compromised, we propose a technique to implement a data capture system using virtual machines.

This data capture system will work similarly Sebek, intercepting syscalls, with the difference that it will act like a hardware interceptor, becoming even more difficult to inspect and detect that the system is monitoring the honeypot. The system will implement the functionality presented in HECK, introduced shortly in the next paragraph.

HECK[Carbone and de Geus 2004] is a tool based on Sebek, acting inside the Linux Kernel. While Sebek intercepts syscalls to implement a logging system, with some additional features like viewing downloaded data into the honeypot, HECK intercepts syscalls that modifies the filesystem. With this approach, it is possible to create a copy of the filesystem at any given timestamp of the attack. This feature, together with Sebek, enables the simulation of the events of an attack, such as filesystem modifications after the execution of malicious code.

With the use of virtual machines, a syscall can be monitored as if it were a hardware monitor, introducing a new layer of abstraction if compared to Sebek and HECK. For this work in progress, User Mode Linux[2](UML) is used as the virtual machine implementation to be adapted as a honeypot virtual machine. This implementation is used mainly because it is open source, which can be freely modified for our purpose and also because of previous experience of the team.

The approach used by UML to implement a virtual machine is to modify the Kernel of the virtual machine so that it can understand how to interact with the host operating system. All the interaction is done mainly over a `ptrace` process that intercepts all the interruptions generated by the virtual machine process. The `ptrace` process is the father of all the virtual machine's processes, and since the interception of syscalls are all done in this layer, no code is added inside the address space of the virtual machine.

When a process in the virtual machine tries to execute a syscall, it will generate an interruption. This interruption will cause the execution to flow to the Kernel space of the host operating system, where it will follow the execution to the `ptrace` process, since the virtual machine is a child process of the `ptrace` process. The `ptrace` process can inspect all data in the virtual machine, and after all data was inspected and correctly logged, the execution is passed to the virtual machine process.

Since inside the UML it is impossible to access any host operatng system process, the data capture system is invisible and can't be disabled by an intruder. This can be guaranteed with the help of SKAS[Carella et al. 2004] on the host operating system.

With this approach, we can capture all information that HECK already could, but

[2]http://user-mode-linux.sf.net

with the following advantages:

- No data is sent through the network.
- The data capture system works outside the honeypot kernel
- The data capture system is implemented as a user process. This makes it much simpler to implement than a kernel module and also allows the use of features like database connections and thread support.

4. Conclusion and Future Work

With the proposed implementation, we will be able to separate the data capture implementation from the honeypot system, creating a more secure environment.

Further work includes a study of the impact the migration of data capture systems can cause to virtual machines. Such work will say a lot about feasiblity of implementing the proposed technique on the variety of honeynet system available.

As the implementation is all done in user space, in the `ptrace` process, another possible work would be the study of the impact of the modularization of this implementation. That is, a system that can install or remove honeypot modules on-the-fly in a virtual machine.

References

Carbone, M. d. P. d. A. and de Geus, P. L. (2004). A Mechanism for Automatic Digital Evidence Collection on High-Interaction Honeypots. In *Proceedings from the 5th IEEE SMC Information Assurance Workshop*, pages 1–8, West Point, NY, USA.

Carella, C., Dike, J., Fox, N., and Ryan, M. (2004). Uml extensions for honeypots in the ists distributed honeynet project. In *Proceedings from the 5th IEEE SMC Information Assurance Workshop*, pages 130–137, West Point, NY, USA. IEEE Computer Society Press.

Corey, J. (2004a). Advanced honeypot identification. *Phrack Inc. (edição falsa)*, 11(63).

Corey, J. (2004b). Local honeypot identification. *Phrack Inc. (edição falsa)*, 11(62).

Dornseif, M. and Klein, T. H. C. N. (2004). NoSEBrEaK–Attacking Honeynets. In *Proceedings from the 5th IEEE SMC Information Assurance Workshop*, pages 123–129, West Point, NY, USA.

Spitzner, L. (2002). *Honeypots: Tracking Hackers*. Addison-Wesley, Boston, MA, USA.

The Honeynet Project (2003a). Know your enemy: Defining virtual honeynets. Disponível em World Wide Web (Agosto de 2004): <http://www.honeynet.org/papers/honeynet/index.html>.

The Honeynet Project (2003b). Know your enemy: Genii honeynets. Disponível em World Wide Web (Setembro de 2004): <http://www.honeynet.org/papers/gen2/index.html>.

The Honeynet Project (2003c). Know your enemy: Sebek.