uc3m | Universidad Carlos III de Madrid

University Degree in Computer Science and Engineering
Academic Year 2021-2022

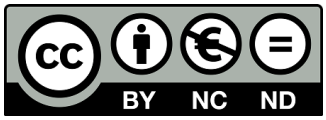
Bachelor Thesis

"An analysis of offensive capabilities of eBPF"

Marcos Sánchez Bajo

Juan Manuel Estévez Tapiador

Leganés, 2022



SUMMARY

Keywords:

DEDICATION

ABSTRACT

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation

As the efforts of the computer security community grow to protect increasingly critical devices and networks from malware infections, so do the techniques used by malicious actors become more sophisticated. Following the incorporation of ever more capable firewalls and Intrusion Detection Systems (IDS), cybercriminals have in turn sought novel attack vectors and exploits in common software, taking advantage of an inevitably larger attack surface that keeps growing due to the continued incorporation of new programs and functionalities into modern computer systems.

In contrast with ransomware incidents, which remained the most significant and common cyber threat faced by organizations on 2021[1], another powerful class of malware called rootkits is found considerably more infrequently, yet it is usually associated to high-profile targeted attacks that lead to greatly impactful consequences.

A rootkit is a piece of computer software characterized for its advanced stealth capabilities. Once it is installed on a system it remains invisible to the host, usually hiding its related processes and files from the user, while at the same time performing the malicious operations for which it was designed. Most common operations include storing keystrokes, sniffing network traffic, exfiltrating sensible data from the user or the system, or actively modifying the data at the infected device. The other characteristic functionality is that rootkits seek to achieve persistence on the infected hosts, with the purpose of being launched again after a system reboot, without further user interaction. The techniques used for achieving both of these functionalities depend on the type of rootkit developed, a classification usually made depending on the level of privileges on which the rootkit operates in the system.

- **User-mode** rootkits run at the same level of privilege as common user applications. They usually work by hijacking legitimate processes on which they may inject code by preloading shared libraries, thus modifying the calls issued to user APIs, on which malicious code is placed by the rootkit. Although easier to build, these rootkits are exposed to detection by common anti-malware programs.

- **Kernel-mode** rootkits run at the same level of privilege as the operating system, thus enjoying unrestricted access to the whole computer. These rootkits usually come as kernel modules or device drivers and, once loaded, they reside in the kernel. This implies that special attention must be taken to avoid programming errors since they could potentially corrupt user or kernel memory, resulting in a fatal kernel panic and a subsequent system reboot, which goes against the original purpose of maintaining stealth.

Common techniques used for the development of their malicious activities include hooking system calls made to the kernel by user applications (on which malicious code is then injected), or modifying data structures in the kernel to change the data of user programs at runtime. Therefore, trusted programs on an infected machine can no longer be trusted to operate securely.

These rootkits are usually the most attractive (and difficult to build) option for a malicious actor, but the installation of a kernel rootkit requires of a complete previous compromise of the system, meaning that administrator or root privileges must have been already achieved by the attacker, commonly by the execution of an exploit or a local installation of a privileged user.

Historically, kernel-mode rootkits have been tightly associated with espionage activities on governments and research institutes by Advanced Persistent Threat (APT) groups[2], state-sponsored or criminal organizations specialized on long-term operations to gather intelligence and gain unauthorized persistent access to computer systems. Although rootkits' functionality is tailored for each specific attack, a common set of techniques and procedures can be identified being used by these organizations. However, during the last years, a new technology called eBPF has been found to be the target of the latest innovation on the development of rootkits.

eBPF is a technology incorporated in the 3.18 version of the Linux kernel[3], which provides the possibility of running code in the kernel without the need of loading a kernel module. Programs are created in a restrictive version of the C language and compiled into eBPF bytecode, which is loaded into the kernel via a new bpf() system call. After a mandatory step of verification by the kernel in which the code is checked to be safe to run, the bytecode is compiled into native machine instructions. These programs can then get access to kernel-exclusive functionalities including network traffic filtering, system calls hooking or tracing.

Although eBPF has built an outstanding environment for the creation of networking and tracing tools, its ability to run kernel programs without the need to load a kernel module has attracted the attention of multiple APTs. In fact, on February 2022, the Chinese security team Pangu Lab reported about a NSA backdoor that remained unnoticed from 2013 that uses eBPF for its networking functionality and that infected telecommunications, scientific and military systems worldwide[4]. More recently, PwC reports about a China-based threat actor that has targeted telecommunications systems with a eBPF-based backdoor[5].

Taking all the previous background into account, and attending to the previous work on this matter by Jeff Dileo from NCC Group at DEFCON 27[6] and by Guillaume Fournier and Sylvain Afchainthe from Datadog at DEFCON 29[7], we can confidently claim that there is a growing interest on researching the capabilities of eBPF in the context of offensive security, in particular given its potential on becoming a common component of modern rootkits. Additionally, there currently exists official efforts to extend the eBPF

technology into Windows[8] and Android systems[9], which extends the mentioned risks to new platforms.

1.2. Objectives

1.3. Regulatory framework

1.3.1. Social and economic environment

1.3.2. Budget

2. STATE OF THE ART

3. METHODS??

4. RESULTS

5. CONCLUSION AND FUTURE WORK

BIBLIOGRAPHY

[1] "Cyber threats 2021: A year in retrospect," PricewaterhouseCoopers. [Online]. Available: https://www.pwc.com/gx/en/issues/cybersecurity/cyber-threat-intelligence/cyber-year-in-retrospect/yir-cyber-threats-report-download.pdf.

[2] "Rootkits: Evolution and detection methods," Positive Technologies, Nov. 3, 2021. [Online]. Available: https://www.ptsecurity.com/ww-en/analytics/rootkits-evolution-and-detection-methods/.

[3] (Dec. 7, 2014), [Online]. Available: https://kernelnewbies.org/Linux_3.18.

[4] "Bvp47 top-tier backdoor of us nsa equation group," Pangu Lab, Feb. 23, 2022. [Online]. Available: https://www.pangulab.cn/files/The_Bvp47_a_top-tier_backdoor_of_us_nsa_equation_group.en.pdf.

[5] "Cyber threats 2021: A year in retrospect," PricewaterhouseCoopers, p. 37. [Online]. Available: https://www.pwc.com/gx/en/issues/cybersecurity/cyber-threat-intelligence/cyber-year-in-retrospect/yir-cyber-threats-report-download.pdf.

[6] Presented at the, Evil eBPF Practical Abuses of an In-Kernel Bytecode Runtime, DEFCON 27. [Online]. Available: https://raw.githubusercontent.com/nccgroup/ebpf/master/talks/Evil_eBPF-DC27-v2.pdf.

[7] Presented at the, Cyber Threats 2021: A year in Retrospect, DEFCON 29. [Online]. Available: https://media.defcon.org/DEF%20CON%2029/DEF%20CON%2029%20presentations/Guillaume%20Fournier%20Sylvain%20Afchain%20Sylvain%20Baubeau%20-%20eBPF%2C%20I%20thought%20we%20were%20friends.pdf.

[8] "Ebpf incorporation in the linux kernel 3.18." (Dec. 7, 2014), [Online]. Available: https://kernelnewbies.org/Linux_3.18.

[9] "Ebpf for windows." (), [Online]. Available: https://source.android.com/devices/architecture/kernel/bpf.

APPENDIX A

APPENDIX B