UNIVERSITÀ DEGLI STUDI DI BERGAMO

Department of Ingegneria Gestionale, dell'Informazione e della Produzione

Master Degree in Ingegneria Informatica

Class LM-32

eBPF for Windows

Advisor

Chiar.mo Prof. Stefano Paraboschi

Master Thesis

Matteo Locatelli

Student number 1059210

ACADEMIC YEAR 2022/2023

Abstract

Berkeley Packet Filter (BPF), an originally Unix-based packet filtering technology, has evolved into a versatile tool with significant impact on network performance and security. This master thesis aims to explore the story of BPF, tracing its development from its inception on Unix-based systems to its adaptation on Windows platforms. Through a comparative analysis, we investigate the challenges, solutions and advancements that have led to the successful integration of BPF in the Windows environment. By studying its history, architecture and programs development, we explore the potential of BPF to revolutionize network engineering on Windows and contribute to the broader understanding of cross-platform technology adoption.

Acknowledgements

Completing this master thesis on the evolution and adaptation of Berkeley Packet Filter on both Linux and Windows platforms has been an enriching journey for me. I am deeply grateful to the individuals whose guidance, encouragement and support have made this research possible. Without their firm belief in my abilities, this effort would not have come to completion.

First and foremost, I extend my heartfelt gratitude to my esteemed advisor, professor Stefano Paraboschi, whose expertise, mentorship and invaluable feedback have been instrumental in shaping this thesis.

My sincere appreciation must be extended to the people in the *Unibg Security Lab* team who actively participated in the development of this thesis: their continuous support throughout the entire research process have motivated me to push my boundaries and aim for excellence. I am grateful for their patience, insightful discussions and profound knowledge in the fields of computer engineering and systems security, which have significantly contributed to the depth and quality of this work: their willingness to share their expertise has been essential in overcoming various challenges faced during this study and in refining the ideas presented in this research. Their support has made this academic pursuit not only a productive venture, but also an enjoyable one. Also, they provided me with the LaTex template that I used to write this thesis.

Speaking of people that gave me something practical that helped me to work on this project, I have to thank Subconscious Compute, an Indian IT company founded in 2020 that works on security for distributed devices and data. Their decision to open-source their GitHub repository, which is under AGPL license, and grant me access to it has been fundamental in enabling me to develop eBPF programs on the Windows platform and to do a better comparison with eBPF on Linux, which was the scope of my master's thesis. The availability of the repository not only provided me with a lot of resources and code examples, but also allowed me to gain insights into best practices and advanced techniques in programming for Windows using eBPF.

Moreover, I would like to express my obligation to the wider academic community of the *Università degli Studi di Bergamo* for providing an environment that encourages learning, curiosity and innovation. I am deeply grateful to everyone who played a part, big or small, in the ending of my academic journey. The education I received has been invaluable and I am fortunate to have had such exceptional guidance throughout my academic period. This thesis stands as a testament to the collective effort and support of those who have been part of my academic journey. The knowledge and experiences I have gained throughout the last five years have been instrumental in shaping my growth as a computer engineering student.

Last but not least, I must express my very profound gratitude to my parents for their love, encouragement and support throughout eighteen years of education. Their belief in my capabilities and constant motivation have been the driving force of my academic achievements, especially during the demanding period of writing this thesis. I owe my successes to them because with their sacrifices they have allowed me to focus on my studies and achieve my academic goals, celebrating every milestone with infinite joy and pride. In conclusion, I am grateful for the life lessons and values they instilled in me, which have shaped me into the person I am today.

Thank you.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

In the ever-evolving landscape of computer science and networking, the demand for efficient, flexible and secure packet filtering technologies has been dominant. The Berkeley Packet Filter (BPF), an innovative technology developed in the Unix environment, has emerged as a powerful tool for network monitoring, traffic analysis and security enforcement. Over the years, BPF has undergone significant advancements, culminating in the birth of Extended Berkeley Packet Filter (eBPF), a groundbreaking extension that has revolutionized network engineering and performance analysis.

1.1 Background

Computer networks establish the backbone of modern communication, enabling the seamless exchange of information across the globe.

The rapid growth of network traffic, the rise of complex cyber threats and the increasing need for real-time monitoring have motivated researchers and engineers to explore innovative solutions to enhance network performance and to build robust security mechanisms. Packet filtering, a fundamental networking technique, serves as a first line of defense in safeguarding networks and optimizing data transmission.

Originally conceived in the 1990s, the Berkeley Packet Filter (BPF) was designed as a mechanism to filter packets at the kernel level for the Berkeley Software Distribution (BSD) operating system (a discontinued operating system based on the early versions of the Unix operating system). However, its potential, consisting of

its lightweightand versatile design, far exceeded its initial purpose and it evolved into a versatile technology with applications across various networking domains.

Over the years, BPF has undergone significant developments and adaptations, until it resulted in the advent of eBPF: with the introduction of a new virtual machine and bytecode, eBPF allowed for the dynamic execution of custom programs within the kernel context, extending its applicability beyond traditional packet filtering to areas such as network monitoring, tracing and deep packet inspection.

1.2 Motivation

Despite the extensive use of eBPF in Unix-based systems, its incorporation into Windows environments has remained a challenge. As Windows continues to be a prominent operating system in both personal and enterprise computing, unlocking the potential of eBPF on this platform becomes crucial for achieving cross-platform network engineering and security solutions.

This thesis will focus on the historical progression of BPF and its adaptation on the Windows platform. In addition to that, we will explore the advancements introduced to eBPF on both operative systems and study the current state of art of eBPF on Windows to show its differences with the Linux environment.

1.3 Objectives

This master's thesis aims to provide an in-depth analysis of eBPF's architecture, installation and functionalities in both operating systems, while showing the history, development and impact of eBPF in the world of computer science and network engineering.

The primary objectives of this research are as follows:

- Tell the history of eBPF: by understanding the origins of BPF, we gain insights into the motivations that led to the creation of eBPF and we can identify the key challenges faced during its integration into Windows and the innovative

solutions designed to overcome them. A look into the historical context provides a solid foundation for exploring eBPF's potential, from a simple packet filtering mechanism to a versatile technology with broader network real-world applications;

- Installation and integration of eBPF on Linux and Windows: we will investigate the process of installing eBPF into both Linux and Windows operating systems. By understanding the differences in installation procedure and requirements on these platforms, we are enabled to leverage the cross-platform capabilities of this technology;

- Development of eBPF programs on Linux and Windows: this thesis will cover the development process of eBPF programs on both Linux and Windows platforms. We will explore the process of creating, loading and executing eBPF programs. Furthermore, by studying the eBPF API, we will:

  - Demonstrate the creation of custom programs to achieve specific networking tasks;
  - Show how far they have come in the development of the technology in the two operating systems;
  - Examine the methods used to safely load eBPF programs into the kernel.

1.4 Organization of the Thesis

BETTER ORGANIZATION -> TEXT OR LIST

The subsequent chapters of this thesis will be organized as follows:

- Chapter 2: the history of eBPF (with real-world examples);
- Chapter 3: eBPF on Linux (installation and programs development);
- Chapter 4: eBPF on Windows (installation and programs development);
- Chapter 5: Future prospects of eBPF on Windows;
- Chapter 6: Conclusion.

Through this master's thesis, we hope to offer a comprehensive understanding of eBPF's significance, capabilities and potential in modern networking environments. We also have the ambition to contribute to the field of computer engineering by closing the gap between Unix and Windows-based network technologies and security measures. By exploring the installation and development processes on both Linux and Windows, we present a comparative analysis of eBPF's cross-platform capabilities.

Chapter 2

Technologies used

Since we already announced that we are going to work on both Linux and Windows, before diving into the installation process of eBPF on both Linux and Windows, it is important to describe the technologies that allowed us to develop programs using eBPF.

2.1 The host environment

The project started with a single Windows 11 PC serving as the host environment for all research and development activities.

The computer has a 64 bit operating system with a processor based on x64, a 16 GB RAM and a Solid-State Drive (SSD) with a capacity of 1TB as for storage. Windows 11, with its user-friendly interface and vast software ecosystem, combined with the power given by the four cores of the Intel Core i7 processor, provided an efficient platform for general computing requirements.

Given the fact that other operating systems were required for this project, the integration of virtualization was crucial to create isolated environments alongside the Windows host.

2.2 Virtual machine for Linux development

For installing and developing programs with eBPF on Linux, a virtual machine running Ubuntu 22.04 was set up within VirtualBox (the version of the Ubuntu operating system is not important).

VirtualBox is a type 2 or hosted hypervisor suitable for individual use and small-scale virtualization scenarios. It is a software application that runs on top of an existing operating system (called host OS) and provides the capability to create and manage virtual machines. Figure 2.1 shows a schematic representation of the architecture just described. VirtualBox allows you to test, develop and run multiple guest operating systems within your host operating system simultaneously, providing a good level of isolation between the host and guest operating systems. As a type 2 hypervisor, VirtualBox relies on the host operating system's kernel to manage hardware resources: it uses device drivers and services from the host OS to interact with the physical hardware, which can introduce some overhead and may affect performance compared to a type 1 hypervisor.

Even though VirtalBox relies on the host OS for certain operations, which can lead to performance differences and potential resource conflicts, it was chosen over a type 1 hypervisor for its user-friendly virtualization solution.

The installation process involved creating a virtual disk, configuring memory and CPU allocation and selecting the Ubuntu 22.04 ISO file previously downloaded for installation [5]. The virtual machine provided a native Linux platform for eBPF program development, compilation and testing.

2.3 Virtual machine for Windows development

Since the main focus is the analysis of eBPF state of art on Windows, the project also demanded the capability to develop eBPF programs specific to the Windows platform. For this purpose, the Hyper-V Console Manager, a native Windows feature, was used to create a separate Windows 11 virtual machine.

Hyper-V is a type 1 or bare-metal virtualization software, also known as a Virtual Machine Monitor (VMM), which runs directly on the physical hardware without

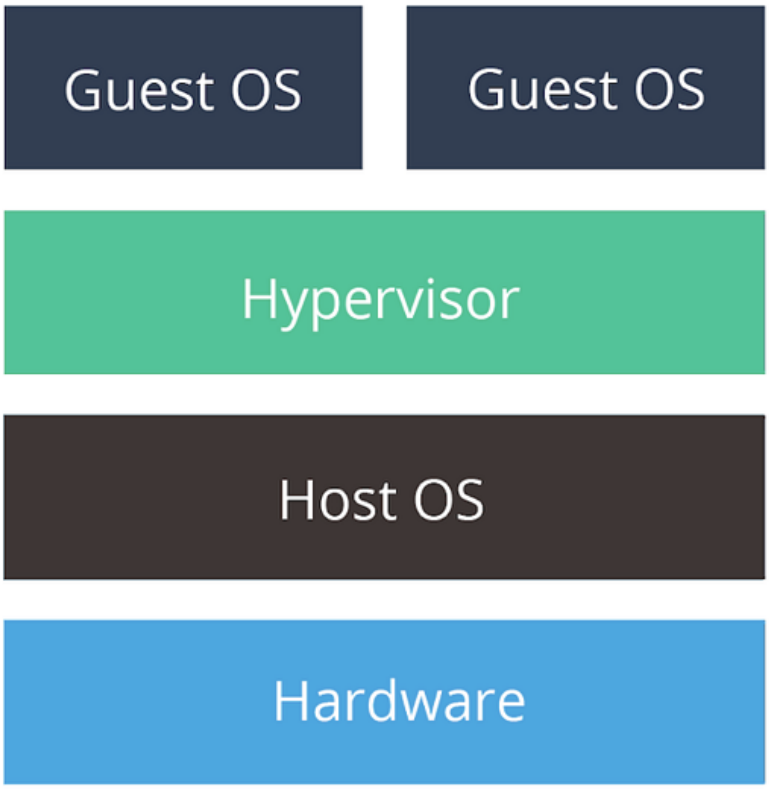

Figure 2.1: Type 2 (or hosted) hypervisor architecture [2].

the need for an underlying host operating system. The illustrative representation of the architecture just described is depicted in Figure 2.2. As the core software responsible for managing virtual machines and allocating hardware resources to each VM, Hyper-V ensures better security and resource utilization by isolating each VM from others and the host OS. With direct access to the physical hardware, it efficiently allocates resources, resulting in improved performance, isolation and scalability compared to type 2 hypervisors like VirtualBox.

Even though hypervisors are favored for their robustness and scalability, enabling the efficient virtualization of large-scale applications and services, the choice of creating a virtual machine using the Hyper-V Console Manager was dictated by the setup instructions described on the *ebpf-for-windows* GitHub repository [6].

We are going to talk about the eBPF installation on Windows later: for now, besides the fact that that he virtual machine was configured with adequate resources to support development tasks effectively, the only thing worth noting is that during the quick creation of the virtual machine the option of "Windows 11 dev environment"

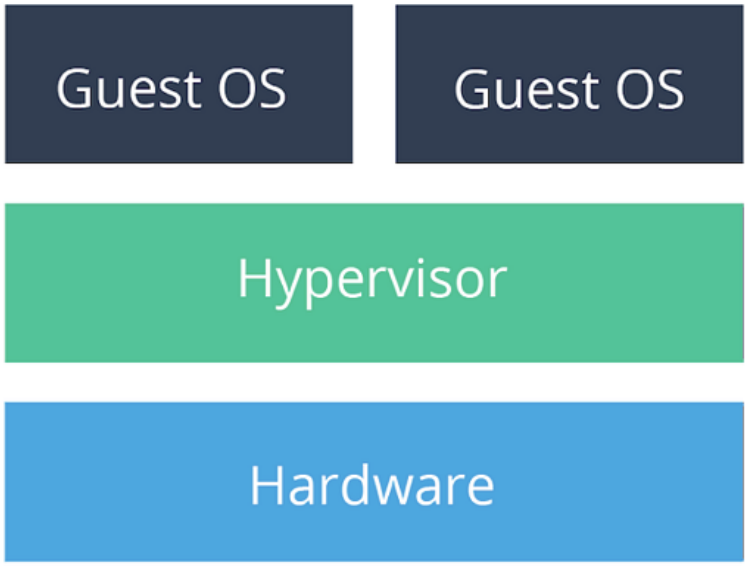

Figure 2.2: Type 1 (or bare metal) hypervisor architecture [2].

must be selected (the tutorial tells to choose the "Windows 10 dev environment", but Windows 11 works as well).

The so created isolated Windows 11 development environment provided a controlled space for testing and optimizing eBPF programs on the Windows platform.

2.4 Repository of the project

GitHub is a platform and cloud-based service for software development and collaborative version control using Git, a distributed version control system that tracks changes in any set of computer files, allowing developers to store and manage their code, owned by the company GitHub Inc., whose logo is displayed in Figure 2.3. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration and wikis for every project. It is commonly used to host open source software development projects.

GitHub has been an invaluable platform for sharing the progress of my work with my co-advisors and facilitating collaboration during the entire development process. Its version control system allowed me to keep track of changes, maintain a detailed history of my project and collaborate consistently with my co-advisors, ensuring a smooth and efficient development workflow. By regularly pushing updates to

Figure 2.3: GitHub *Invertocat* logo [1].

the repository [3], my co-advisors were able to monitor the evolution of my work, review code changes, provide timely feedback and offer valuable suggestions for improvement.

Beyond the immediate scope of my thesis, the public visibility of the GitHub repository opens up the possibility of sharing my work with the broader community. By making the repository public, we hope that others can benefit from the knowledge and insights gained during the project, encouraging collaboration and contributions from future researchers and developers.

CITARE TEXSTUDIO E LATEX

Chapter 3

The history of eBPF

Bibliography

[1] *GitHub Logo*. URL: https://allvectorlogo.com/github-logo/ (visited on 07/2023).

[2] *Hypervisors architectures images*. URL: https://medium.com/teamresellerclub/type-1-and-type-2-hypervisors-what-makes-them-different-6a1755d6ae2c (visited on 07/2023).

[3] *Master thesis repository*. Mar. 2023. URL: https://github.com/Matteo-Locatelli/master_thesis.

[4] Steven McCanne and Van Jacobson. "The BSD Packet Filter: A New Architecture for User-level Packet Capture". In: (Dec. 1992). URL: https://www.tcpdump.org/papers/bpf-usenix93.pdf.

[5] *Ubuntu ISO image download page*. URL: https://www.ubuntu-it.org/download (visited on 04/2023).

[6] *Virtual machine installation instructions*. URL: https://github.com/microsoft/ebpf-for-windows/blob/main/docs/vm-setup.md (visited on 05/2023).

[7] *Windows eBPF Starter*. URL: https://github.com/SubconsciousCompute/windows-ebpf-starter (visited on 06/2023).