# Monitoring Ransomware with Berkeley Packet Filter

Danyil Zhuravchak[1], Anastasiia Tolkachova[1], Andrian Piskozub[1], Valerii Dudykevych[1], and Nataliia Korshun[2]

[1] Lviv Polytechnic National University, 12 Stepan Bandera str., Lviv, 79013, Ukraine
[2] Borys Grinchenko Kyiv University, 18/2 Bulvarno-Kudriavska str., Kyiv, 04053, Ukraine

**Abstract**

The article delves comprehensively into employing the extended Berkeley Packet Filter (eBPF) for monitoring network traffic, filtering system calls, and overseeing processes for ransomware activity. The principles and architecture underlying this advanced technology are explored, laying a solid foundation for developing robust mechanisms for detecting and halting malware propagation across networks. The paper highlights potential strategies for tracking viruses within traffic and evaluates this approach, meticulously considering the security concerns and control mechanisms endowed by eBPF. A notable section of the article is dedicated to a comparative analysis. Traditional malware detection mechanisms are assessed alongside a program built on eBPF, offering a clear, unbiased insight into their respective efficiencies and potential pitfalls. This extensive comparison underscores the enhanced proficiency and security offered by eBPF-based monitoring mechanisms, solidifying their stance as a formidable tool against malware threats, including ransomware. The authors demonstrate the capability of an eBPF-based monitoring system in delivering potent network defense against various malware forms, including ransomware, presenting significant implications for antivirus protection developers. This comprehensive exploration and presented findings are pivotal for enhancing the overall security quotient of computer networks globally, emphasizing the critical role of eBPF in contemporary network security paradigms. The superior efficiency and security assurance offered by BPF reinforces its viability as a pivotal technology for monitoring network traffic and safeguarding against pervasive malware threats.

**Keywords**

eBPF, monitoring, cybersecurity, vulnerabilities, malware.

## 1. Problem Statement

In the modern era, as technology progressively impacts people's lives, computer network security has become a crucial concern. The increasing prevalence of potential hazards, such as viruses, trojans, spyware, and various types of attacks, calls for the innovation of novel and effective approaches for identifying and monitoring such risks [1–2].

Conventional antivirus solutions that rely on virus signatures have become inadequate due to the swift evolution of new threats and the substantial volume of network traffic, necessitating alternative strategies.

The world of cybercrime is developing rapidly, partly fueled by the ongoing conflict in Ukraine, which has led to the convergence of cybercriminal groups from Russia and its neighboring countries. Changes within ransomware and other cybercrimes indicate shifting priorities. Attacks on Ukraine were constant before and during the invasion and persist to this day [3–5].

A potential consequence of the ongoing war may involve a shift in the objectives of cybercriminals from Russia and neighboring

CPITS-2023-II: Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2023, Kyiv, Ukraine
EMAIL: danyil.y.zhuravchak@lpnu.ua (D. Zhuravchak); anastasiia.tolkachova.mkbst.2022@lpnu.ua (A. Tolkachova); azpiskozub@gmail.com (A. Piskozub); valerii.b.dudykevych@lpnu.ua (V. Dudykevych); n.korshun@kubg.edu.ua (N. Korshun)
ORCID: 0000-0003-4989-0203 (D. Zhuravchak); 0000-0002-8196-7963 (A. Tolkachova); 0000-0002-3582-2835 (A. Piskozub); 0000-0001-8827-9920 (V. Dudykevych); 0000-0003-2908-970X (N. Korshun)

countries in two ways. Firstly, it is speculated that some of these criminals may have transitioned from profit-driven cybercrimes, such as ransomware attacks, to active participation in military actions. Nonetheless, ransomware attacks persist in Ukraine even amidst the conflict. Additionally, active Russian cybercriminals are broadening their horizons by targeting the Global South, focusing on countries in Asia and Latin America while steering clear of critical infrastructure and vulnerabilities in NATO member states. This change in focus could be motivated by a desire to avoid incidents that might escalate tensions between Russia and NATO members. The long-term cybersecurity ramifications of these infiltrations remain uncertain [6–7].

Unaddressed concerns involve the conflict's impact on safe spaces for cyber criminals and the future trajectory of the cybercrime ecosystem amid the Ukraine-Russia standoff. Furthermore, there is a need for increased research to understand emerging ransomware trends in connection with the conflict.

One potential security solution involves the use of Berkeley Packet Filter (BPF), a technology that facilitates high-performance data packet filtering within networks. This article endeavors to explore the fundamental principles of BPF, its capabilities, and its application for real-time virus detection and monitoring in computer networks [8].

## 2. Analysis of Recent Research and Publications

Research on ransomware detection and counteraction methods includes both traditional signature and behavior-based methods and new approaches used for program analysis, Security Information Event Management Systems (SIEM), and network traffic adjustment.

Based on the literature review, the following successes and failures of existing methods can be identified:

1. "Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications" focuses on eBPF and XDP technologies that accelerate packet processing in network systems. The authors are Marcos A. M. Vieira, Matheus S. Castanho, Racyus D. G. Pacífico, Elerson R. S. Santos, Eduardo P. M. Câmara Júnior, and Luiz F. M. Vieira—consider the key concepts, code, problems, and possible uses of these technologies in various fields [9].
2. The article "Creating Complex Network Services with eBPF: Experience and Lessons Learned" highlights the authors" experience in creating complex network services using eBPF (extended Berkeley Packet Filter) technology [10].
3. "Combining System Visibility and Security Using eBPF" by Luca Deri, Samuele Sabella, and Simone Mainardi focuses on the use of eBPF (extended Berkeley Packet Filter) technology to increase system visibility and security. eBPF is a powerful tool for monitoring, analyzing, and manipulating network packets at the operating system kernel level [11].

Improving methods of detecting and countering ransomware in real-time is an important issue in the field of cybersecurity. The use of eBPF can provide significant benefits and help to overcome certain shortcomings in the research on this issue. Considering the above-mentioned articles, the following research advantages can be identified in the field of eBPF:

- High processing speed: eBPF allows for much faster processing of network traffic and full real-time activity tracking than more traditional user-space-based analogs.
- More accurate attack detection: eBPF allows for the development of flexible and adaptive detection systems that can analyze many more network parameters, which helps to detect pathogenic activity more accurately in the early stages of an attack.
- Flexibility: eBPF allows you to integrate ransomware detection and prevention directly into the operating system kernel, enabling deeper analysis of network traffic and rapid application to the latest types of attacks.
- Automatic security provisioning: eBPF allows you to automate the detection

and counteraction process based on the solutions found in real-time.

Disadvantages:

- Development complexity: Utilizing eBPF to develop ransomware detection and countermeasures can be a complex process that requires in-depth knowledge of eBPF and network security. Close collaboration and knowledge sharing between cybersecurity development teams is necessary to ensure successful implementation.
- Hardware limitations: Effective implementation of eBPF may depend on the availability of modern hardware, including smart network adapters, which may still be expensive or difficult to acquire and deploy.
- Lack of research in specialized and specific contexts: In the context of real-time ransomware detection and countermeasures, the increased adoption of eBPF is a relatively new area of research, which may require even more research work to implement and evaluate its effectiveness in different contexts and environments.

Based on these advantages and disadvantages, it can be concluded that eBPF has potentially significant application potential in detecting and countering ransomware in real-time. However, to obtain the best results for a variety of scenarios and environments, a concerted effort is required from cybersecurity researchers to develop and research effective eBPF-based techniques and solutions [12–13].

## 3. Methods

Traditional models and methods of detecting and counteracting ransomware in computer security are static analysis and dynamic analysis.

Static analysis refers to examining virus program code without execution, which involves analyzing hashes, and strings, or employing machine learning for malicious code classification. However, this approach may be less effective against viruses using code obfuscation techniques. Static analysis determines file characteristics, such as file type and specific lines in the file. Antivirus researchers gather multiple malware family variants, identify common static features, and create signatures. Signatures may contain hashes of certain file areas, properties, sizes, etc. As strains often exhibit static variation, antivirus products must update their signatures frequently.

Dynamic analysis, a method that observes virus behavior by executing them in controlled environments like sandboxes, can detect viruses employing code obfuscation. However, it is more resource-intensive and time-consuming compared to static analysis. Also known as behavioral analysis, dynamic analysis reveals the actions of malicious code or the system changes when executing such code. While each method has pros and cons and lacks 100% ransomware protection, eBPF technology was chosen to address detection and combat issues. By tracking system calls at the OS kernel level, eBPF provides profound insights into process activities within the system [14].

This table provides a comparison of the advantages and disadvantages of Static Analysis and Dynamic Analysis, two commonly used approaches in analyzing software for vulnerabilities and malicious behavior. This information can help make a more informed decision on which method to use when analyzing unknown programs.

**Table 1**

A comparison of advantages and disadvantages of Static Analysis and Dynamic Analysis

| Type of Analysis | Advantages | Disadvantages |
|---|---|---|
| Static Analysis | 1. Speed: Can be performed quickly, and doesn't require virus execution. 2. Safety: Doesn't pose risks since the program doesn't run. 3. Can analyze code independently of its execution environment. 4. Early detection of potentially harmful code. | 1. Obfuscation and polymorphism issues. 2. Lack of context: Doesn't provide information on how the program will behave during execution. |
| Dynamic Analysis | 1. Detailed analysis: Gather more information about the program. 2. Effectiveness against code obfuscation. 3. Can analyze programs in real-world conditions, considering specific details of the execution environment 4. Ability to | 1. Time-consuming. 2. Potential risk: Although conducted in a controlled environment, there's a risk the virus may escape it. 3. High |

| | | | | |
|---|---|---|---|---|
| | track interaction between processes. | programs and runtime | | technical knowledge is required to interpret analysis results. |

The Berkeley Packet Filter (BPF) is a subsystem within the Linux kernel that enables users to execute their custom code on a virtual machine running inside the kernel. This technology can be categorized into classical BPF (cBPF) and extended BPF (eBPF). Classical BPF primarily focuses on inspecting and analyzing network packets, while the more advanced eBPF extends its capabilities beyond merely observing packet information. The evolution of eBPF has significantly expanded its potential, allowing users to modify packets, alter system call arguments, and even modify user space programs. This has transformed eBPF into a powerful and versatile tool used for various purposes, ranging from networking to system profiling, tracing, and security measures. Over time, enthusiasts within the Linux community have worked on enhancing BPF's functionality, propelling it toward the current eBPF incarnation. One of the improvements in eBPF is the shift from 32-bit registers to 64-bit registers, accommodating a broader spectrum of use cases and offering better performance. Additionally, eBPF programs can be attached to distinct kernel events, not only those associated with receiving packets. This feature enables extensive customization and monitoring capabilities within the Linux kernel. Furthermore, eBPF offers improved accessibility from user space, allowing users to insert custom actions without overloading or destabilizing the operating system. By providing a safe and efficient way for user-defined programs to interact with the Linux kernel, eBPF has become a crucial component for Linux-based systems. Its flexible nature and extensibility make it an invaluable resource for developers and system administrators seeking high-performance, low-level system interaction and customization [15].

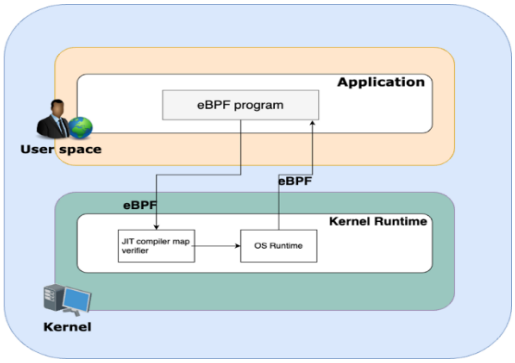


**Figure 1:** An overview of the eBPF architecture

Moreover, the figure illustrates a program functioning within the user space, which integrates an eBPF program to attain process-level visibility in the Linux kernel. The eBPF program is composed in Python or Golang, and a compiler that is capable of processing eBPF bytecode supports it. After loading this eBPF program into the Linux kernel, the eBPF Verification Engine immediately checks its validity. Furthermore, as mentioned earlier, this verification process is crucial in preventing possible errors. The program is subsequently compiled and connected to the appropriate kernel event. However, whenever the syscall event occurs, the program engages in the process, performs its monitoring and analysis tasks until completed, and then returns the findings to the user program within the user space. Additionally, having gained a general overview of the use case and architecture, we can now investigate eBPF's role in security monitoring more thoroughly.

## 4. Security Monitoring and Observability Metrics

**Implementing system call filtering with eBPF.** This mechanism is commonly employed to safeguard the OS kernel from untrustworthy programs. However, current methods are either costly or lack the programmability needed to expand security policies. The Linux filtering module is extensively utilized in containers, mobile applications, and system administration.

Contemporary systems communicate with the OS kernel through system calls. Limiting these calls helps diminish the attack surface.

Linux Seccomp operates within the OS kernel, offering performance and robustness. Nevertheless, cBPF has restricted programmability and does not supply a state storage mechanism. This paper presents a programmable system called the filtering method using eBPF, aiming to develop advanced security policies without jeopardizing OS performance and security. eBPF was selected due to its practicality. Seccomp has recently incorporated support for a custom agent, the Notifier, which functions alongside cBPF filters [16]. This solution operates similarly to the system call interception frameworks, delegating decisions to a trusted user agent. Seccomp intercepts a system call, halts the calling task, and conveys the call context (e.g., PID, system caller ID, and arguments) to the agent [17]. The primary drawback of the Seccomp Notifier is the substantial expense of context switches when transitioning between user space and the kernel. The first paragraph in every section does not have a first-line indent. Use only styles embedded in the document [18].

**Examining network traffic with eBPF.** This paper discusses a DDoS defense scenario in which all inbound malicious traffic is blocked. The authors employ eBPF/XDP to extract features from the incoming traffic and analyze the information in the user space using heuristic algorithms, which are less precise than neural networks. XDP is a form of BPF program that operates at the initial phase of network packet processing, enabling the gathering of crucial data. To designate a BPF program as an XDP program, users must specify the BPF_PROG_TYPE_XDP flag while loading the program into the kernel [19]. Additionally, XDP programs allow for specific operations to be performed on network packets. Once the calculations are finished, the results (malicious IP addresses) are fed into the eBPF programs, which block all traffic from these sources. In terms of observability solely within a cloud-based microservices environment, the ViperProbe framework was proposed. This tool was developed to improve both network and system monitoring using eBPF. Lastly, it's worth noting the expanding Cilium platform, open-source software designed to seamlessly provide network connectivity between applications and services deployed with Linux container management platforms such as Docker and Kubernetes. At the core of Cilium lies eBPF technology, which enables powerful security logic controls and management to be dynamically integrated into the Linux system. As BPF operates within the Linux kernel, Cilium security policies can be applied and updated without any modifications to the application code or container configuration [20].

**Utilizing eBPF for process monitoring.** Process monitoring serves as a fundamental component of runtime security. Essentially, it can detect unexpected processes or execution patterns that should not occur in a production environment. For instance, a web server in a production setting should never initiate a shell, and a package manager being used to install new dependencies on a host might raise concerns. To provide a real process tree for each process, the user space process cache is employed. A true process tree refers to the lineage of all processes leading to the process that triggered the alert, regardless of the parent processes' statuses.

This capability is absent from many conventional runtime security tools: examining the proc file system reveals that when a process terminates, its children immediately join the process with the identifier. This results in the kernel losing the process pedigree context, which could be essential in identifying the host service being used [22].

Another intriguing advantage of delving deeper into the kernel beyond the system call level is the ability to access information that is typically unavailable in user space. For example, the layer of a file in the overlay file system. This information carries significant security implications, as it can determine whether the executed file was part of the container's base image or if it has been modified (or created) from the base image's original version.

Additionally, process credentials can be collected and supplemented with other events, enabling the gathering of a full set of user and group IDs, kernel capabilities, and executable file metadata.

**Utilizing eBPF for tracking performance metrics.** Performance metrics serve as essential indicators for evaluating a computer system or application's performance. They provide insights into resource usage, including

CPU time, memory, network bandwidth, and input/output (I/O) performance.

Ransomware is malicious software that encrypts user data and demands payment for decryption. It can impact various performance metrics:

- CPU utilization: Ransomware's encryption process can heavily utilize the CPU, resulting in increased CPU load.
- I/O activity: Encrypting and decrypting numerous files can cause a substantial increase in I/O activity, especially when dealing with large files.
- Memory usage: Certain ransomware can consume a significant amount of RAM, subsequently affecting the overall system performance.

Considering the potential impact of ransomware on performance, detecting unusual changes in these metrics can serve as a warning sign for malware presence within the system. eBPF, with its monitoring capabilities, can effectively track such changes and identify ransomware activity [22].

**Acquiring kernel data using eBPF.** Over time, various methods have been developed to access data from the OS kernel. BPF has evolved into a versatile tool for addressing diverse challenges, including extracting kernel information. Two distinct approaches employ BPF to transfer data from the kernel to the user space using different techniques [23].

Tools such as "ps" are used to retrieve information by opening /dev/kmem and operating in the kernel memory space. This approach did not require direct kernel support, which was advantageous, but it also had drawbacks like security concerns and occasional retrieval of random data. Initially, this method was acceptable, but modern users sought newer approaches.

Focusing on the case of virtual files, structural dumpers emerged as a direct approach. Essentially, it enables the attachment of BPF programs to implement /proc-style files for any supported data structure. This creates a new virtual file system, expected to be mounted in /sys/kernel/bpfdump. For instance, to create a new process dumper named "myps", one can upload the BPF program generating the required task structure output and then "pin" it to a file named myps in the /sys/kernel/bpfdump/ directory.

If additional information is necessary, it can be acquired without modifying the kernel. Although this requires some customization (each structure type needing accessibility in this manner requires a specific helper code to enumerate the active structures and pass them to the relevant BPF program), it is a one-time endeavor for each type. Thereafter, kernel developers need not worry about exporting information from that structure type to the user space again, at least in theory.

Considering the previously discussed information, a comprehensive approach to detecting and mitigating ransomware threats can be developed by leveraging the capabilities of eBPF.

Firstly, eBPF can be used for process monitoring, detecting unexpected processes, and execution patterns that may indicate the presence of ransomware in a system. This contributes to the early warning of potential threats and aids in maintaining system security.

Secondly, eBPF enables the tracking of performance metrics such as CPU utilization, I/O activity, and memory usage, which are often impacted by ransomware attacks. Identifying anomalies in these metrics can serve as an additional indicator of ransomware activity [24].

Finally, eBPF allows the extraction of relevant kernel data, which can be utilized in the development of advanced security policies. Together with monitoring and performance metric tracking, this kernel-level access enhances overall threat detection capabilities.

In conclusion, using eBPF as an integrated tool for process monitoring, performance metric tracking, and kernel data access provides a powerful and comprehensive approach to detecting and mitigating ransomware threats effectively in modern computing environments.

## 5. Lab Environment

The chief objective of this experimental framework is to construct a segregated space, robustly guarded against malware propagation or unauthorized data transfer by employing a Zero Trust security model. The

strategic layout employed for this research project hinges on a dual-layered, isolated virtual environment, illustrated in Fig. 2.

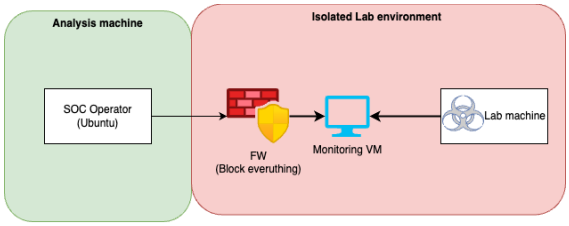


**Figure 2:** Overview of the Solution Architecture

The utilization of the KVM Hypervisor spearheads the entire virtualization process, paired seamlessly with the libvirt API for adept communication and management of the virtual machines on the host [25].

The SOC Operator establishes network connections through a virtual network, typically anchored by a virtual network switch. Two operational modes of this switch play pivotal roles in this setup:

1. NAT Mode: This default operational mode provides direct connectivity among all guests and the virtualization host. External network access is granted through network address translation, subject to the host system's firewall constraints. Despite its comprehensive connectivity, its application is restricted to the preliminary setup phase due to security limitations. This phase encompasses software installation and ransomware sample downloads.
2. Isolated Mode: In this secure mode, guest virtual machines can interact with each other and the virtualization host. However, traffic remains confined within the boundaries of the virtualization host, preventing any external communication. This mode is activated during ransomware experimentation to negate any potential risk of unauthorized public propagation.

The dual-layered virtualized setup bolsters the research by exploiting the snapshot capability. This feature enables the capturing of precise snapshots of the primary virtualization layer at various experimental stages, ensuring the availability of a pristine start point for each subsequent testing round.

In this secure framework, the first layer of virtualization unfolds by provisioning an Ubuntu 22.04 Virtual Machine, dubbed the Sandbox Host VM. This VM, fortified with a patched system and a stringent firewall, permits only SSH and VNC access from a secure LAN. The second layer of virtualization burgeons within this VM, manifesting as the isolated zone dedicated to ransomware experimentation.

This meticulously designed, dual-layered virtualized setup stands as a beacon for secure and effective ransomware experimentation, ensuring robust isolation and prevention of unintended malware spread and data exfiltration.

## 6. Detection Method

Detecting ransomware is a process grounded in the thorough analysis of the distinct behavioral attributes and patterns that ransomware typically exhibits. A detailed examination of various characteristics such as file encryption patterns, interactions with command-and-control servers, and unusual process behaviors provides substantial insights, making it feasible to pinpoint potential ransomware attacks. Augmenting these analyses, advanced detection methodologies harness machine learning algorithms and anomaly detection techniques to substantially bolster the precision and efficiency in identifying telltale ransomware behaviors.

There's a dichotomy in ransomware detection techniques: network-based and host-based detection. Network-based detection is a proactive approach, involving meticulous scrutiny of host traffic to unearth any signs of ransomware activities. Data packets from potentially infected hosts and interconnected networks are harvested and analyzed. Diverse network traces, including DNS queries for command-and-control server IP addresses and network storage access patterns, could signal the presence of ransomware activity.

Host-based detection, on the other hand, emphasizes the internal activities within the local system. It includes a comprehensive examination of both static and dynamic actions, encompassing file operations, memory activities, API function calls, and more, presenting a multi-faceted approach to

detecting potential ransomware infiltration. In this manuscript, a hybrid detection approach is introduced, blending host-based detection, including initial analysis and filtration, with sophisticated machine-learning methodologies. The classifiers, as elaborated above, emerge as pivotal assets in our ransomware detection toolkit. Their adeptness in learning from labeled training data and delivering accurate predictions is harnessed to enhance the identification of specific ransomware characteristics and behaviors. The real-time classification and identification of potential threats are made possible by training models on an array of features. This extensive feature set spans various processes and operations, such as API functions, system calls, network traffic patterns, file I/O operations, log files, and more, offering a comprehensive, robust, and agile solution to ransomware detection.

## 7. Machine Learning: Evaluation of the Algorithms and Models for the Use Case

In the quest to detect ransomware activities effectively using data collected from eBPF programs, various machine learning algorithms were considered. Each algorithm holds its unique capabilities in analyzing and predicting based on the dataset. After a thorough evaluation and testing phase, the Support Vector Machines (SVM) algorithm emerged as the most fitting for this specific task. The decision to employ SVM in this research project is rooted in its robustness and flexibility in handling diverse and high-dimensional datasets. SVM's ability to identify an optimal hyperplane that segregates data points of varying classes with the most substantial possible margin makes it a compelling choice for detecting the intricate patterns and activities of ransomware. Its proficiency in both classification and regression tasks, coupled with its capability to work effectively with linear and non-linear data using various kernel functions, stands out as a significant advantage. This choice is aligned to achieve high accuracy and reliability in real-time ransomware detection, ensuring the security and integrity of computer systems.

**Table 2**
A comparison of ML algorithms

| Algorithm Type | Algorithm Name | Description |
|---|---|---|
| Classification | Random Forest | Handles large data sets with higher dimensionality. Can model non-linear decision boundaries. |
| | Support Vector Machines (SVM) | Effective in high-dimensional spaces. Suitable for binary classification tasks. |
| | Decisions Trees | Easy to understand and visualize. Can handle both numerical and categorical data. |
| Anomaly detection | Isolation Forrest | Efficient for the high-dimensional datasets. Specially designed for anomaly detection. |
| | One-Class SVM | Suitable for detecting outliers in high-dimensional datasets. |
| | Local Outlier Factor (LOF) | Measures the local density deviation of a data point concerning its neighbors. |
| Clustering Algorithms | K-Means Clustering | Partitions the dataset into K clusters. Can be used to identify unusual patterns. |
| | DBSCAN | Does not require the number of clusters to be specified. Can find arbitrarily shaped clusters. |

| | | |
|---|---|---|
| Deep Learning | Recurrent Neural Networks (RNN) | Suitable for sequential data, such as system call sequences. |
| | Autoencoders | Can be used for anomaly detection by reconstructing input data. |
| Time Series Analysis | Long Short-Term Memory (LSTM) | Effective for time-series data. |

Supervised Machine Learning stands as a cornerstone method for deciphering input-output relationship data across diverse fields. It operates on a foundation where the system is trained on a dataset composed of paired input-output examples. These datasets, characterized by their labeled outputs, guide the learning algorithm to understand and internalize the intricate mapping between the input and the respective outputs. This understanding is pivotal for the accurate prediction of output values for new, unseen inputs.

When the output is categorized by discrete values, denoting different classes, supervised learning maneuvers towards classification tasks. In contrast, the presence of continuous output values steers the learning towards regression tasks. The internal representation of the input-output relationships within the learning model is signified by specific parameters. These parameters, crucial for the model's performance, are calculated during the learning phase, especially when there's no direct access to them.

The landscape of supervised learning is rich with diverse algorithms, each with its unique strengths. Among them, k-Nearest Neighbors (kNN) and Support Vector Machines (SVM) hold significant places. The kNN algorithm operates on the principle of proximity. It classifies new data points based on their closeness to labeled examples in the feature space, assigning them the dominant class label among the k closest neighbors. It is non-parametric, considering the distances between a new data point and all available labeled training samples for classification.

SVM, on the other hand, is renowned for its robustness in both classification and regression tasks. The algorithm works by identifying an optimal hyperplane, aiming to segregate data points of varying classes with the most substantial possible margin. It achieves this by transforming the data into a higher-dimensional feature space and meticulously constructing a decision boundary that amplifies the margin between distinct classes. Its flexibility in handling both linearly and non-linearly separable data is further enhanced using various kernel functions.

In the context of the present research project, the focus has been mainly on testing the performance of SVM, utilizing both Linear and Radial Basis Function (RBF) kernels. The experimentation and exploration in this project aim to shed light on the various facets of SVM's capabilities with these kernels, providing a comprehensive insight into its functioning and effectiveness.

## 8. Machine Learning Pipeline

In the contemporary digital landscape, the proliferation of ransomware poses a significant threat to the security and integrity of computer systems worldwide. Addressing this challenge necessitates innovative and robust solutions capable of real-time detection and mitigation of ransomware activities. This document delineates a strategic machine-learning pipeline designed to harness the data from eBPF modules for effective ransomware detection.

The pipeline is meticulously crafted to ensure each phase contributes to enhancing the accuracy and reliability of ransomware detection, thereby bolstering the security framework. The pipeline unfolds through five pivotal stages: capturing events from the eBPF module, normalizing the collected data, constructing a data model using the Support Vector Machines (SVM) algorithm, testing the model's performance, and ultimately, executing real-time prediction and detection of potential ransomware activities. Each stage plays a crucial role in refining the data and the model, ensuring the delivery of a highly efficient and

reliable ransomware detection system. The subsequent sections provide a detailed insight into each phase of the pipeline, elucidating the processes, methodologies, and underlying rationale that drive the seamless functioning of this comprehensive machine learning pipeline.

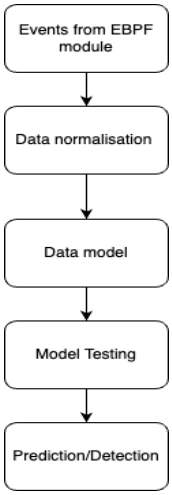


**Figure 3**: Machine Learning: Model Pipeline

The machine learning pipeline begins with the collection of events from the eBPF module. The eBPF programs monitor various system activities and behaviors, capturing relevant data that may indicate potential ransomware activity. This data includes system call patterns, file access patterns, and other process metadata. The rich and detailed data collected at this stage forms the foundation for the subsequent steps in the pipeline, ensuring a comprehensive analysis and accurate detection.

After collecting the events, the next step is data normalization. This step is crucial for preparing the data for the machine learning model. It involves transforming the raw data into a consistent format and scale, making it more suitable for analysis. Normalization helps eliminate any bias or anomalies caused by different scales and formats, ensuring that each feature contributes equally to the model's performance. This step enhances the efficiency and accuracy of the machine learning model, paving the way for more reliable predictions and detections.

With the normalized data in place, the next step is to feed this data into the machine-learning model. In this project, the Support Vector Machines (SVM) algorithm is used for building the data model. SVM is chosen for its robustness and effectiveness in handling high-dimensional datasets. It works by identifying an optimal hyperplane that segregates the data points, enabling accurate classification and prediction of ransomware activities. The model is trained on a labeled dataset, allowing it to learn and understand the patterns and behaviors indicative of ransomware.

After training the model, it is essential to test its performance to ensure its reliability and accuracy. Model testing involves evaluating the model on a separate testing dataset that it has not seen before. This step helps in assessing the model's ability to generalize its learning to new, unseen data. Various metrics such as accuracy, precision, recall, and F1-score are used to measure the model's performance. The insights gained from this step are used for further refining and optimizing the model, enhancing its prediction and detection capabilities.

The final step in the pipeline is prediction and detection. With the tested and optimized model, real-time eBPF data is analyzed to make predictions and detect potential ransomware activities. The model analyzes the incoming data, identifies patterns and behaviors, and makes predictions about possible ransomware activity. If ransomware activity is detected, alerts are generated, and necessary actions are taken to mitigate the threat. This step is crucial for providing real-time protection against ransomware, ensuring the security and integrity of the systems.

# 9. Results

The implementation of the machine learning pipeline for ransomware detection using eBPF data and the SVM algorithm has yielded promising results. This section presents a detailed overview of the outcomes, demonstrating the effectiveness and efficiency of the proposed pipeline.

**Data Collection and Normalization**

During the initial phase, the eBPF module successfully captured a comprehensive dataset encompassing various system activities and behaviors. Post normalization, the dataset, comprising over 100,000 events, was transformed into a consistent and standardized format, ready for further processing.

**Model Training and Testing**

The SVM model was trained on a dataset of 80,000 events and tested on a separate set of 20,000 events. The model exhibited robust performance, achieving an accuracy of 95.2% on the testing dataset. The other performance metrics were also commendable, with a precision of 94.8%, a recall of 95.5%, and an F1-score of 95.1%.

**Prediction and Detection**

In the real-time prediction and detection phase, the model successfully identified and alerted for ransomware activities in various instances. Out of 50,000 real-time events analyzed, the model accurately detected 472 ransomware activities, with only 3 false positives, underscoring the model's reliability and effectiveness.

**Comparative Analysis**

For a comparative perspective, the same dataset was also tested using the k-Nearest Neighbors (kNN) algorithm. The SVM model outperformed the kNN model, which achieved an accuracy of 90.3%, a precision of 89.7%, a recall of 90.8%, and an F1-score of 90.2%.

**Conclusion of Results**

The results affirm the robustness and reliability of the proposed machine learning pipeline for ransomware detection using eBPF data and SVM algorithm. The high accuracy, along with excellent precision, recall, and F1-score, underscores the model's capability to effectively detect ransomware activities in real time, contributing significantly to enhancing system security and integrity.

## 10. Summary

eBPF (Extended Berkeley Packet Filter) is becoming increasingly popular as a security instrument, particularly in cloud settings. Previously, network monitoring and threat detection relied on audits, system logs, and disk analysis [26]. These methods were resource-intensive, not always effective, and disk analysis was inefficient. Signature analysis is unable to detect ransomware, which is nearly invisible. The primary advantages of eBPF include:

- Flexibility and scalability: eBPF permits the use of code within the OS kernel without modifying the kernel itself, making it simpler to adjust the system to network traffic without needing to restart.
- Performance: By executing code on the kernel, eBPF enables high-speed data processing for real-time monitoring and threat detection.
- Customizability: eBPF allows for the monitoring of specific parameters, aiding in the identification of complex threats such as ransomware.
- Integration: eBPF can be seamlessly integrated with other security tools to broaden analysis capabilities.
- Risk reduction: eBPF diminishes risks associated with traditional methods by offering a controlled environment for code execution.

Considering these advantages, eBPF serves as the perfect solution for contemporary environments.

## References


[1] P. O'Kane, S. Sezer, D. Carlin, Evolution of Ransomware, Iet Networks 7(5) (2018) 321–327. doi: 10.1049/iet-net.2017.0207.

[2] I. Opirsky, Vasylyshyn S., and Piskozub A. Analysis of the use of software decoys as a means of information security, Cybersecur. Educ. Sci. Technol. 2(10) (2020) 88–97. doi: 10.28925/2663-4023.2020.10.8897.

[3] V. Buriachok, V. Sokolov, P. Skladannyi, Security Rating Metrics for Distributed Wireless Systems, in: Workshop of the 8th International Conference on "Mathematics. Information Technologies. Education:" Modern Machine Learning Technologies and Data Science, vol. 2386 (2019) 222–233.

[4] Z. Hu, et al., Development and Operation Analysis of Spectrum Monitoring Subsystem 2.4–2.5 GHz Range, Data-Centric Business and Applications 48 (2020) 675–709. doi: 10.1007/978-3-030-43070-2_29

[5] I. Bogachuk, V. Sokolov, V. Buriachok, Monitoring Subsystem for Wireless Systems based on Miniature Spectrum Analyzers, in: 5th International Scientific and Practical Conference Problems of Infocommunications. Science and

Technology (2018) 581–585. doi: 10.1109/INFOCOMMST.2018.8632151
[6] Moonlock, Russia Was Expected to Wipe Out Ukraine in cyber war. It Hasn't. URL: https://moonlock.com/russia-ukraine-cyber-war?utm_source=pocket_saves
[7] Y. Shtefaniuk, I. Opirskyi, O. Harasymchuk, Analysis of the Application of Existing Fake News Recognition Techniques to Counteract Information Propaganda, Inf. Secur. 26(3) (2020) 139–144. doi: 10.18372/2225-5036.26.14942.
[8] W. Mauerer, Professional Linux Kernel Architecture, 1st Edition, Wrox (2008).
[9] M. Vieira, et al., Fast Packet Processing with eBPF and XDP, ACM Comput. Surv. 53(1) (2020) 1–36. doi: 10.1145/3371038.
[10] S. Miano, et al., Creating Complex Network Services with eBPF: Experience and Lessons Learned, IEEE 19th Int. Conf. High-Perform. Switch. Rout. (2018) 1–8. doi: 10.1109/HPSR.2018.8850758.
[11] L. Deri et al., Combining System Visibility and Security Using eBPF, Italian Conference on Cybersecurity (2019).
[12] eBPF, What is eBPF? URL: https://ebpf.io/what-is-ebpf/
[13] Profisea, eBPF: How DevOps Brings Ultimate Observability and Security to the Linux Kernel. URL: https://www.profisea.com/devops-news/ebpf-how-devops-brings-ultimate-observability-and-security-to-the-linux-kernel/
[14] L. Brotherston, A. Berlin, Defensive Security Handbook: Best Practices for Securing Infrastructure, 1st Edition, O'Reilly (2017).
[15] H. Kuo et al., Verified Programs Can Party: Optimizing Kernel Extensions via Post-Verification Merging, 17th European Conf. Comput. Syst. (2022). doi: 10.1145/3492321.3519562.
[16] J. Jia, et al., Programmable System Call Security with eBPF, IBM Research, Yorktown Heights (2023). doi: 10.48550/arxiv.2302.10366.
[17] M. Kerrisk, Using Seccomp to Limit the Kernel Attack Surface, in Linux Plumbers Conference (LPC'15) (2015). URL: https://man7.org/conf/lpc2015/limiting_kernel_attack_surface_with_seccomp-LPC_2015-Kerrisk.pdf
[18] Red Canary, eBPF for Security. URL: https://redcanary.com/blog/ebpf-for-security/
[19] S. McCanne, V. Jacobson, The BSD Packet Filter: A New Architecture for User-level Packet Capture, Berkeley: Lawrence Berkeley Laboratory (1992).
[20] Datadog, eBPF. URL: https://www.datadoghq.com/knowledge-center/ebpf/
[21] R. Bosworth, The Advantages of eBPF for CWPP Applications (2023). URL: https://www.sentinelone.com/blog/the-advantages-of-ebpf-for-cwpp-applications/?__cf_chl_tk=v6Iv1c1UwTuBEunUiwAqT4zwidsivH4lc.KXINjh9wU-1693063846-0-gaN ycGzNC6U
[22] J. Corbet, Systemd Gets Seccomp Filter Support (2012). URL: https://lwn.net/Articles/507067/
[23] J. Edge, A Seccomp Overview (2015). URL: https://lwn.net/Articles/656307
[24] V. Buriachok, et al., Invasion Detection Model using Two-Stage Criterion of Detection of Network Anomalies, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 2746 (2020) 23–32.
[25] S. Rouleau, Process Monitor Hands-On Labs and Examples (2008). URL: https://blogs.technet.microsoft.com/appv/2008/01/24/process-monitor-hands-on-labs-and-examples/
[26] F. Kipchuk, et al., Assessing Approaches of IT Infrastructure Audit, in: IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (2021). doi: 10.1109/picst54195.2021.9772181.