Transforming Weakness into Strength: Improving Unreliable Malware Detection Methods

Pavel Novak, and Vaclav Oujezsky

Original scientific article

***Abstract*—This paper proposes a novel malware detection methodology that leverages unreliable Indicators of Compromise to enhance the identification of latent malware. The core contribution lies in introducing a sequence-based detection method that contextualizes unreliable IoCs to improve accuracy and reduce false positives. Unlike traditional methods reliant on predefined signatures or behavior analysis, this approach dynamically assesses system behaviors, focusing on suspicious actions and interaction patterns. Key contributions include a novel combination of unreliable IoCs with sequence alignment methods, an extensive mapping study of detection techniques, and initial experiments on a dataset of over 19,000 malware samples. Results demonstrate the method's ability to cluster and identify malware families based on their behavioral signatures, even in its early developmental stage. This innovative approach shows promise for detecting previously unknown threats, establishing a foundation for advanced research in malware detection.**

***Index Terms*—Behavioral Analysis, Cybersecurity, Malware Detection, Sequence Similarity.**

I. Introduction

As the complexity and frequency of cyber threats continue to rise, traditional malware detection methods, such as signature-based and heuristic approaches, face significant challenges in keeping up with the evolving tactics employed by adversaries. Malware authors are increasingly utilizing sophisticated obfuscation techniques, polymorphism, and evasive strategies to bypass conventional defenses. In this context, the techniques used by defenders must evolve as well.

This paper builds on a proposal for a novel approach to malware detection presented at the SoftCOM 2024 conference [1]. This approach leverages non-reliable Indicator of Compromise (IoC) to improve detection efficacy and enable the identification of latent malware. Our idea is simple. Latent and hiding malware might be hidden in the system for some time. But it still somehow changes the system state [2]. By monitoring tiny suspicious actions and indicators, we can reveal the malware presence. Unlike traditional methods, which rely on predefined signatures or known attack patterns, our method dynamically assesses system behavior and interaction patterns that are deemed suspicious. Since the collected indicators are not reliable on their own and the methods employed may result in a high rate of false positives, our approach seeks to enhance reliability by placing these actions in context and examining similarities and patterns in the sequence of suspicious actions, rather than relying on a single detection hit.

This paper further develops this idea and presents the results of a mapping study of malware detection methods, sequence matching algorithms, and initial experiments with a dataset obtained from public sources. These experiments aim to provide basic verification, revealing whether the method has potential for success. Based on a manual review of various malware families, we identified a set of 19 suspicious events that we aim to collect for each sample in the preprocessing step. After preprocessing the data, we extracted statistics about the suspected events and employed multiple algorithms to cluster and categorize the results to verify the concept.

Even though both the testing dataset and the suspicious event set were not in their final state and we conducted only several experiments on a limited dataset, the results obtained exhibit promising outcomes. We expect further improvements in the algorithm's reliability as our research progresses, with the number of collected events increasing alongside the dataset.

Like any other malware detection method, this one is not bulletproof. Since the descriptions of the suspicious events being collected are publicly available, an attacker may attempt to conceal their presence. However, the general principle of the algorithm is robust, as it allows for custom changes in what is considered suspicious and facilitates the creation of reliable detection based on unreliable indicators.

In our research, and in the context of the proposed algorithm, we face three major questions: What malware detection methods and techniques exist, and how reliable are they? How can we compare the sequence of events, considering not only the sequence itself but also the internal structure of each event? Lastly, is the proposed method more effective in disclosing unknown threats than the methods currently in use?

The main contributions of this work are:

- Novel Malware Detection Approach: The methodology combines unreliable IoCs and dynamic behavioral analysis to create an enhanced detection framework, diverging from traditional static or heuristic methods.
- Sequence-Based Analysis: A new approach contextualizes suspicious actions in sequences, providing a mechanism to improve detection accuracy, particularly for unknown threats.
- Extensive Mapping Study: A comprehensive review of

Manuscript received November 6, 2024; revised December 6, 2024. Date of publication December 16, 2024. Date of current version December 16, 2024. The associate editor prof. Toni Perković has been coordinating the review of this manuscript and approved it for publication.

Authors are with the Department of Computer Systems and Communications, Masaryk University, Brno, Czechia (e-mails: novakpav@mail.muni.cz, oujezsky@fi.muni.cz).

Digital Object Identifier (DOI): 10.24138/jcomss-2024-0098

existing malware detection methods and sequence similarity techniques, forming a foundation for our methodology.
- Experimental Validation: Initial experiments with a dataset of 19,313 malware samples showcase clustering and classification capabilities, highlighting the method's potential in reducing false positives and detecting latent threats.

The following sections elaborate on the underlying principles, implementation, and empirical evaluation of the initial experiments based on the proposed method and the rest of the paper is structured as follows. Section II provides an overview of the malware detection methods and methods used to compare the similarity of the obtained data. Section III briefly summarizes the method and its key aspects. Section IV details the methodology and specific steps taken during the experiments. Section V discusses the performed experiments and evaluates their results. Finally, Sections VI and VII conclude and discuss the paper and summarize the results presented in the previous sections.

II. State of the Art

This section provides a mapping study of the two areas essential for our research. Section II-A is devoted to an overview of detection methods in general and their reliability. Section II-B focuses on methods that may be applicable for comparing suspicious event sequences.

A. Malware Detection Methods

This section maps the currently used malware detection methods and evaluates their focus, implementation difficulty, resource demands, and reliability based on a literature review. Table I presents the main directions in malware detection approaches based on the collected events [3].

TABLE I
Malware Detection Methods Overview

Category	Impl. Difficulty	Resource Demands	Reliability
Traffic Fingerprinting [4], [5], [6], [7], [8]	Medium	Medium	Medium
System Call Sequence [9], [10], [11], [12], [13], [14]	High	Medium	Medium
Assembly Code Sequence [15], [16], [17], [18]	Medium	Low	Medium to High
Byte Sequence [19], [20], [21], [22]	Low	Low	High
Hardware Traces [23], [24]	High	Medium	Medium
Behavioral Analysis [25], [26], [27], [28], [29]	High	High	Medium
System Artifacts [30], [31], [32], [33]	Low to Medium	Low	Low

1) Traffic Fingerprinting: Traffic fingerprinting is a family of methods that can be used for various purposes, including malware detection and traffic monitoring. The core of these methods is to condense the extensive amount of packets flowing through the network into manageable pieces of information that can be further processed. However, this condensation comes at a cost, as we lose information, and detection methods based on this data may have a higher false positive rate than those based on Deep Packet Inspection (DPI) [34], [35]. On the other hand, these methods can detect new and unknown threats and can also handle encrypted communication.

John Althouse et al. work on the fingerprinting of Transport Layer Security (TLS)-encrypted traffic. Their initial method, JA3, utilizes information in the initial unencrypted TLS handshake messages to identify the underlying TLS library used by the communicating program [4]. This method can help identify malware-generated traffic, and there exists a database of JA3 fingerprints mapped to particular malware families [36]. However, this database is not regularly updated.

The same research group developed a more generic set of fingerprinting methods, JA4+, capable of analyzing Hyper Text Transfer Protocol (HTTP), Secure Shell (SSH), and Transmission Control Protocol (TCP) traffic. However, no database of malicious fingerprints exists so far.

Cabaj et al. also employed an HTTP traffic fingerprinting method to detect specific ransomware families based solely on the traffic patterns of HTTP messages observed on the network [6]. Since this method proved successful, it was tested only on two ransomware families.

Liu et al. introduced a novel approach to fingerprinting network traffic in order to detect malware [7]. They fingerprint multiple sessions instead of individual packets, and their method successfully recognized several malware families.

The MalDiscovery method, introduced by Hong et al., is dedicated to the identification of malware in encrypted traffic based on the similarity of encrypted session features [8].

2) System Call Sequence: Another group of methods that can be applied to detect polymorphic malware is the sequence of system calls. Since almost every program needs to use system calls to interact with the underlying hardware, the sequence of such calls may reveal the true intentions of the software.

The most straightforward method is to use the import table, or its hash [12]. Since most compilers order the import hash based on the order of the first usage of the respective system calls, a simple hash over this table can serve as a signature-based malware detection method.

Chen et al. attempted to use a more sophisticated method that considers not only the bare sequence of system calls but also their parameters, training a deep neural network [13]. The problem with this method is that it requires system call hooking during runtime.

Amer et al. analyzed system call sequence graphs of malicious and benign programs to derive the key features that distinguish malicious graphs from benign ones [14].

Shenderovitz et al. used patterns of system calls to detect and recognize Advanced Persistent Threat (APT) groups for

specific samples [9]. They also collected a dataset of APT samples to test their results.

Prachi et al. applied a genetic algorithm to recognize new and unknown malware families based on system call sequences pruned from generic calls [10].

Maniriho et al. utilized Natural Language Processing (NLP) methods to extract core features from the sequence of system call names [11].

3) Assembly Code Sequence: Analyzing opcode sequences offers a further level of granularity for evaluating software behavior. Previous methods focused only on system calls, as they are essential for the software; many actions can be performed only via system calls as a proxy.

Kakisim et al. address the common problem of all methods using opcode sequences—the extensive length of the sequence [15]. They present a method to create more compact embeddings that are shorter but preserve all important information needed to detect malware. Their experimental results achieve 100% correctness.

Jeon et al. reduce the high dimensionality of the opcode sequence using convolutional autoencoders and a dynamic recurrent neural network classifier to recognize malware samples, achieving an accuracy of 96% [16].

Santos et al. use opcode frequency as a feature and then train several classification models, including decision trees, Support Vector Machine (SVM), Bayesian networks, and K-Nearest Neighbours (KNN), achieving roughly 90% efficacy [17].

Pektas et al. focus on Android malware and extract features from execution graphs by analyzing all possible execution paths [18]. They use Deep Learning (DL) methods to classify apps.

4) Byte Sequence: The last level of granularity we can achieve is analyzing the sequence of raw bytes in the binary. This approach can be tricky, as the same byte sequence might be interpreted differently depending on the context. The most straightforward approach in signature-based methods is to specify the exact combination of byte sequences in the binary that determines its maliciousness. This needs a lot of manual work performed by an expert or a malware analyst. However, more sophisticated methods exist.

Saha et al. propose a novel method inspired by genome alignment techniques to compare DNA sequences [19]. They trained a Machine Learning (ML)-based classifier to detect malware.

Rezaei et al. use byte embedding on the Portable Executable (PE) header to obtain a more compact representation of the PE header features [20]. They then cluster these features using KNN, achieving over 90% malware recognition accuracy. A similar approach was used by Raff et al. [21].

Li et al. employed the technique of grabbing the memory dump of a running process and evaluating its opcode sequence together with a Graph Neural Networks (GNN) model to determine the maliciousness of the running binary, achieving over 95% accuracy [22].

5) Hardware Traces: Collecting information directly from the hardware is more challenging than simply analyzing the malicious binary. However, hiding traces at the hardware level is much more difficult for an attacker.

TABLE II
SEQUENCE SIMILARITY METHODS OVERVIEW

Method Family
Markov Chains [40], [41], [42], [43]
Graph-based Methods [44], [45]
Sequence Alignment Methods [46], [47], [48], [49]
Language Models [50], [51]
Neural Networks [52], [53], [54]

Tian et al. use Intel Processor Trace to collect the control flow of the inspected program and apply DL methods to detect malware [23]. Their method performs well regarding the resources needed.

Panker et al. focus on the detection of malware in the Virtual Machine (VM) environment. They dump memory regions of the running process, collect a set of interesting features, and determine the maliciousness of the running program [24].

6) Behavioral Analysis: Behavioral analysis of software is another method for detecting malware. Behavior-based techniques typically collect events on the system, such as file modification patterns, process creation, or network access. These methods are the closest to our proposed approach.

Jacob et al. provide a general overview of behavior-based detection methods [25]. A significant number of studies focus on behavioral malware detection on mobile devices, particularly Android [26], [27].

Liu et al. present their malware behavior feature classifier based on a set of manually identified actions [29].

7) Suspicious Artifacts: Essentially, any cyber observable can serve as a basis for malware detection, which also motivated our research. We can utilize not only some of the aforementioned detection methods but also many individually suspicious actions. Simple events like creation of new registry key, change of the system file or scheduling new task might be considered suspicious as these actions are often performed by malware to achieve persistence or hiding itself on the system. For the best of our knowledge there is currently no study addressing the most prevalent suspicious actions performed by malware, this is one of the future goals of our research. However, we can draw inspiration from the taxonomy of commonly used techniques published by MITRE [37] and derive information from publicly available Sigma repository [38].

B. Sequence Similarity Methods

The second challenge of our proposed method is to determine the most suitable way to compare the similarity of event chains. This section provides the results of a mapping study of sequence similarity methods [39]. Table II summarizes the studied approaches and the main publications. Because the sequence similarity problem is very common in biology and chemistry, many publications are related to this field.

1) Markov Chains and Markov Models: A Markov chain is a mathematical model that describes a sequence of events where the probability of each event depends only on the state of the previous event. A Markov model, based on this concept, is used to represent systems where future states depend solely

on the current state, rather than the entire history of previous states. Markov models are commonly applied to sequence data, such as text, biological sequences, or time-series data, to model transitions between states and predict future sequences. They can also be utilized to compare sequence data by calculating the likelihood of one sequence being generated by a given model and predicting future states based on current observations. Several studies have employed Markov Chains and Markov Models to compare and predict sequence data.

Paxinou et al. analyzed sequences of actions performed by students during an experiment to evaluate the effectiveness of different learning techniques [40].

Perdikaris et al. used Markov Models to analyze a teacher's behavior based on their actions [42].

Saw et al. applied Markov Chain parameters to compare protein sequences and calculate their similarity [43].

2) Graph-based Methods: In the context of malware detection, representing malicious activities as graphs provides a structured way to capture the sequence of actions or events. For our problem, each graph represents a malware attack, where nodes symbolize suspicious actions observed during the malware execution. The task is to evaluate the similarity between these event chains by incorporating both the structural properties of the graph and the internal characteristics of individual nodes, assessed using a node similarity function $f(\text{node1}, \text{node2})$.

The Graph Edit Distance (GED) is a natural choice for measuring similarity between chain-like graphs. GED computes the minimum number of operations (insertion, deletion, substitution of nodes or edges) required to transform one graph into another. This approach can be adapted to our problem by incorporating the node similarity function f to adjust the cost of node substitution.

Other methods include a simple comparison of adjacency matrices. The DeltaCon method is based on node-pair similarity in the graph [45].

3) Sequence Alignment Methods: Since the graphs primarily represent simple chains of events, they can be treated as sequences of nodes, allowing us to employ techniques from sequence alignment, which are commonly used in bioinformatics [49], [47]. Algorithms like Needleman-Wunsch (for global alignment) or Smith-Waterman (for local alignment) can be applied to align two attack chains, using the node similarity function f as the scoring function for matching nodes [55], [56]. These methods aim to maximize the overall similarity score by allowing gaps (insertions or deletions) and mismatches between nodes, which is analogous to handling differences in the sequence of actions across different malware attacks. Sequence alignment methods are well-suited to linear structures like chains and can efficiently incorporate the similarity of nodes through f. Additionally, these algorithms offer dynamic programming solutions, making them computationally tractable even for longer chains.

4) Language Models: Language models, particularly those employed in NLP, can be adapted to compare and predict sequence data by treating sequences as analogous to sentences or phrases. In the context of suspicious malware events, each event can be viewed as a "token" in a sequence, where the order of events and their interdependencies carry meaningful patterns. By training a language model on historical malware event sequences, it can learn the statistical relationships between events, enabling it to predict the likelihood of future sequences, detect anomalies, or classify new event sequences as suspicious or benign. This approach leverages the model's ability to understand temporal dependencies and context, making it well-suited for analyzing and predicting malware behavior patterns.

A similar approach has been used in bioinformatics to compare protein sequences by Ofer et al. and Li et al. [50], [51].

5) Neural Networks: Neural networks can be applied to malware event sequences by treating each sequence as a fixed-dimensional vector, which ignores the sequential nature of the data. In this approach, each malware event sequence is transformed into a feature vector that encapsulates key attributes, without regard for the order in which the events occur. The neural network then processes these vectors, learning patterns or correlations between features that can be utilized for tasks such as classification, anomaly detection, or comparing malware sequences. This method is beneficial when the order of events is less important than the presence or combination of specific features.

However, when the order is crucial, GNN provide a more powerful solution. GNN are designed to operate on graph-structured data, where individual events can be represented as nodes and their interactions, dependencies, or co-occurrences as edges. This representation allows GNN to model non-linear and non-sequential relationships, capturing how different malware events might influence each other within a broader context. By representing suspicious events as a graph, GNN can uncover deeper patterns and interactions that classical neural networks might miss, thereby improving the model's ability to detect subtle and complex malware behaviors. This combination of classical and graph-based approaches enables a more flexible and comprehensive analysis of malware event sequences, accounting for both feature patterns and relational structures.

Bai et al. proposed a novel algorithm called SimGNN for efficient similarity search on the sequences [52]. Xu et al. implemented a prototype similarity search algorithm Gemini capable of efficient similarity search in big data [53].

III. Proposed Method

The novel malware detection method proposed in [1] uses structured Cyber Threat Intelligence (CTI) data combined with advanced ML techniques to detect latent malware. Traditional detection methods often rely on specific IoCs, such as file hashes or Internet Protocol (IP) addresses, which can be easily circumvented by attackers. In contrast, this method aggregates high-level, albeit less reliable, IoCs, including events like file modifications, registry changes, and suspicious network communications, to create a holistic overview of the system state.

By correlating these aggregated indicators, our system should be able to detect subtle patterns in system behavior that

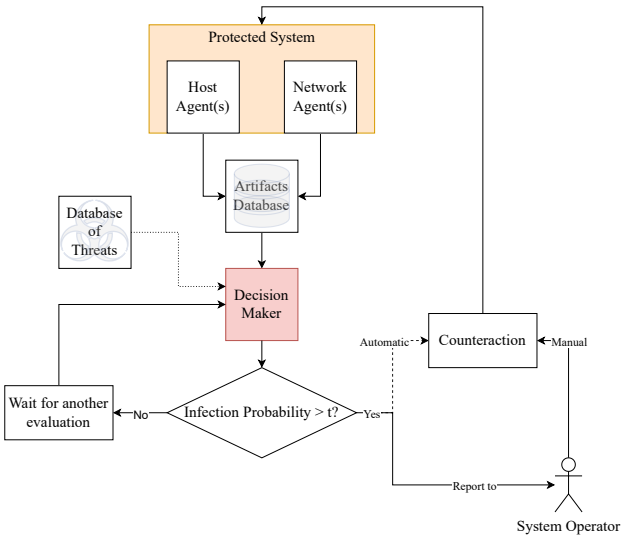

Fig. 1. Detection Pipeline Flow

align with known malware tactics. These patterns are collected and stored using a structured format.

Our method stands out by detecting anomalies even when individual IoCs might not suffice to trigger alarms in traditional systems. The collected suspicious events are transformed into structured reports, and the scoring mechanism provides a probabilistic indication of potential system compromise, facilitating faster response times and reducing false positives.

The detection mechanism depicted in Figure 1 presents a simplified view of the entire pipeline. The protected system is monitored using two agents (components). The host agent component resides directly on the monitored VM but runs only during specified time frames. This component is responsible for collecting host-based artifacts, such as changes in system files, settings, newly created and scheduled tasks, system logs, and other defined potentially suspicious actions. The host agent runs only when a request is sent to limit the resources it consumes.

On the other hand, the network agent is a standalone component completely separated from the monitored system. This design has two advantages: first, it is harder, if not impossible, for an attacker to manipulate this component; second, we can monitor network traffic continuously, which is not feasible for the host agent. Continuous monitoring is crucial since network traffic is transient and cannot be captured once it flows through the network. Both agents produce a stream of events in a standardized format and store them in the central artifact database.

At the time of evaluation, the decision maker component collects all previous events, merges them with new events that occurred since the last evaluation, and compares the sequence of events (if any) with known malware patterns stored in the threat database. If the similarity is above a given threshold or if the sequence has the potential to evolve similarly to a known malware pattern, the system is considered infected and reported. Note that the similarity will likely not be 100%. This is because not only the order of the sequence is evaluated, but also the similarity of individual event values, which may be randomized.

The proposed method differs from traditional malware detection approaches by combining unreliable IoCs as foundational elements to identify complex, evasive malware. It leverages the advantages of both signature-based and behavior-based methods while mitigating their limitations. The method partially employs signature-based detection, which relies on static patterns, but also allows for the inclusion of highly unreliable detection patterns with a high false-positive rate. These observations are then connected through sequence analysis, closely aligning with behavior-based detection systems.

However, behavior-based detection systems typically collect exclusively either network traffic or host-based artifacts. Additionally, behavior-based detection is significantly more resource-intensive, as it generally observes the real-time behavior of every process on the system and collects extensive information. By contrast, our method focuses only on suspicious actions, which significantly reduces its resource demands [57].

In conclusion, this approach provides a significant improvement over conventional malware detection systems by capturing broader and more generic behavioral patterns, thus adding another layer of security to the system.

IV. Methodology

Our approach relies on a sequence of custom unreliable detection methods. To test their efficacy, we need to build a custom testing environment (sandbox) capable of collecting relevant information. However, creating such a sandbox is a time-consuming task. Therefore, we decided to utilize data from public sandbox services and, in conjunction with a data pre-processing step, create a dataset to test and evaluate our approach.

We collected reports from 19,313 malware samples obtained from the public sandbox service Joe Sandbox [58]. These malware reports were parsed and pre-processed to form a dataset containing the selected events used in our detection method. The whole process, shown in Figure 2, consists of the following key steps: 1) Dataset collection, 2) Data pre-processing, 3) Basic statistics collection and 4) Unsupervised learning. The following sections describe the processes of sample collection, data pre-processing, and feature extraction, which lead to the creation of the final dataset.

A. Sample Collection

As with any detection method, the quality of testing data is essential. Since we are using a custom set of events for which no existing dataset is available, we faced the challenge of how to collect the necessary data. While we are in the process of building a specialized sandbox environment capable of monitoring custom events, this is a time-consuming task that is not yet complete. Therefore, we opted to utilize existing malware reports and convert them into the format required for our experiments and initial validation of the proposed detection method.

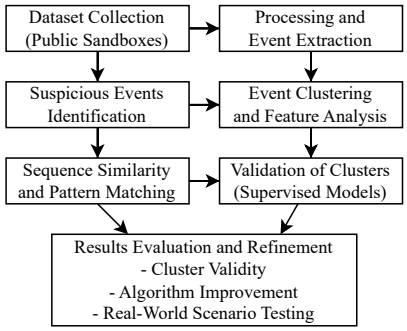

Fig. 2. Block diagram of the proposed malware detection methodology

We used reports from the publicly available and free sandbox service Joe Sandbox for several reasons. The results are openly accessible and provide extensive information about the analyzed samples, including detailed network communication, the complete process tree, and other useful data. This level of detail is particularly beneficial for collecting certain events, such as those derived from comprehensive network traffic captures.

In total, the final dataset consists of 19,313 randomly selected malware samples. Detailed statistics about the dataset are presented in Table III and Table IV.

The samples were selected randomly, with the only requirement being that they were classified as malicious by the automated classification of the sandbox.

B. Pre-Processing

Based on a manual analysis of multiple malware families, we identified an initial set of suspicious events commonly occurring in the studied samples. This selected set is only a subset of a larger set of monitored events planned for future work. The features encompass both network-based and host-based events. The complete set of collected features is presented in Table V.

If a suspicious feature was identified in the report, a new event was created. Since we were unable to obtain timestamps for all events in the reports and reconstruct the chronological order of events, we decided to omit this aspect for now. However, in future work, the temporal sequence of events will play a crucial role in malware identification, particularly when using graph-based similarity search methods or Markov Chains. For the purpose of our experiments, we focused solely on the set of identified events as the dataset.

TABLE III
MALWARE FILETYPES IN THE DATASET

Filetype	Count
exe	11,738
xlsx	1,038
doc	862
xls	480
xlsm	134
dll	128
N/A	4,427

TABLE IV
MALWARE TYPES IN THE DATASET

Identified Malware Type	Count
Agent Tesla	3,426
FormBook	2,207
Lokibot	1,361
Remcos	843
Snake Keylogger	782
RedLine	650
N/A	5,516

The result of the pre-processing step is a collection of identified events for each sample, accompanied by the following metadata:

- **ID**: Universally Unique Identifier (UUID) used to uniquely identify the particular event.
- **Feature Type**: Categorized as either Network or Host.
- **Feature Name**: The specific name of the feature.
- **Value**: The particular value identified as suspicious.
- **Reference**: Additional context that may enhance understanding (e.g., the full URL for the random path event).
- **Datetime**: The date and time of the event occurrence (currently not utilized).

The collected dataset consists of a separate file for each studied sample, populated with the set of observed suspicious events from Table V. The suspicious events were extracted from the reports and evaluated using our custom evaluation logic to determine if they satisfied any of the identified suspicious categories. The source code for this evaluation, along with the specific criteria that the observed values must meet, is available in the project GitLab repository [59].

TABLE V
COLLECTED SUSPICIOUS EVENTS

Feature Type	Name
Network	Direct HTTP connection to IP
Network	Random path in Unified Resource Locator (URL)
Network	Random domain in URL
Network	Suspicious filetype
Network	Suspicious Top Level Domain (TLD)
Network	Double extension
Network	PE header in data
Network	Using geolocation service
Network	Simple Mail Transfer Protocol (SMTP) communication
Network	Communication to mega.io or archive.org
Network	Suspicious JA3 hash
Host	Random filename
Host	Process running from suspicious location
Host	Unusual PE entrypoint section
Host	Unusual PE section names
Host	Manipulation with firewall or Anti-Virus (AV)
Host	Changes in autorun registries
Host	Scheduled tasks

V. Experiments and Results

The initial experiments with a limited dataset aimed to reveal key features of the resulting event sets, guiding our decisions regarding the sequence comparison methods. In total, we tested 19,313 randomly selected malware samples, each of which was reported as malicious.

A. Goals

The primary goal of our experiments was to verify several hypotheses regarding the method, test its potential, and gather insights about the dataset that could inform future design and research decisions.

The main questions addressed by these experiments include: What is the typical length of the event chain? How many different types of events are usually observed for a given malware sample? Do various malware families form groups with similar observations? Are these groups identifiable through visual inspection? Can these groups be identified through unsupervised learning (clustering)? Finally, can these groups be identified through supervised learning (classification)?

The results of these experiments were not intended to measure the efficacy of the detection method. Instead, they focused on exploring the potential of this method and gaining insights into malware behavior. The knowledge acquired from these initial experiments will facilitate future research on this novel approach.

B. Event Chain Statistics

As a starting point, we utilized the set of suspicious events presented in Table V. Table VI explores the frequency of each suspicious event across the samples. The most prevalent suspicious event was the random process name, occurring in more than 96% of samples. This statistic may be biased though, as many samples are submitted with the Message Digest (MD5) or Secure Hash Algorithm (SHA1) as their filenames, which are evaluated as random strings. In real-world scenarios, this should not pose a problem, as benign processes typically do not have random names.

Among the network-based suspicious events, the most common was communication with suspicious TLD. Attackers often utilize non-standard and suspicious TLD because they allow for the registration of names associated with existing companies, and they are cheaper and easier to register [60], [61].

Another interesting statistic concerns the number of unique event types, indicating how many distinct suspicious events were observed for each sample. The results are illustrated in Figure 3. The vast majority of samples exhibit between 3 to 5 unique suspicious event types. Notably, there were 244 samples with 0 suspicious events, which included partially misclassified samples (those that were actually clean) and parsing errors. The maximum number of unique event types observed was 13. The distribution of unique event types conforms to a Poisson distribution with $\lambda = 4.36$.

While the chain of 3 to 5 unique suspicious events may be insufficient for similarity search, considering the entire event chain—potentially including duplicates in event types (e.g., multiple communications to suspicious TLD)–yields more favorable numbers. Figure 4 presents the histogram of event chain lengths for the studied samples. The event chain statistics revealed that the mean length of the event chain is 19.97, with a median of 10 and a mode of 8. The range of the event chain lengths spans up to 883, indicating a wide variation in lengths. Additionally, the standard deviation is calculated as 41.46, reflecting significant variability in the dataset.

The average event chain length is notably high, even with the limited set of only 19 specified suspicious events. This is promising, as the size of the event set is expected to grow in the future. The range of event chain lengths is quite large, with 244 samples exhibiting no observed events, while the maximum event chain length reached 833. However, the

TABLE VI
Collected Suspicious Events

Type	Name	Count	Percentage
Host	Random process name	18,585	96.2%
Host	Process running from suspicious location	17,066	88.3%
Network	Suspicious TLD	9,191	47.6%
Host	Autorun modification	4,757	24.6%
Host	New task scheduled	4,659	24.1%
Host	Unusual section name	4,637	24.0%
Network	Suspicious extension	4,468	23.1%
Network	Direct IP communication	3,439	17.8%
Network	Download of executable file	3,323	17.2%
Network	Randomly generated path	2,893	15.0%
Network	Randomly generated domain	2,757	14.2%
Host	Unusual entrypoint section	2,550	13.2%
Network	Using geolocation service	2,049	10.6%
Network	SMTP communication	1,072	5.5%
Network	Exfiltrates user information	886	4.5%
Network	Double extension	686	3.5%
Network	Suspicious JA3 fingerprint	531	2.7%
Network	Using public storage	384	2.0%
Host	Firewall manipulation	347	1.8%

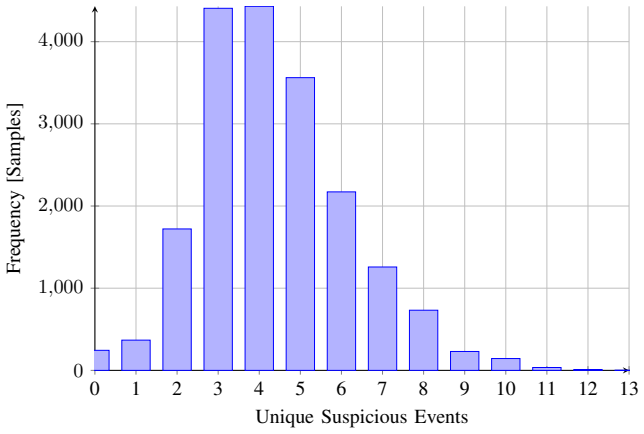

Fig. 3. Number of Unique Suspicious Events per Sample

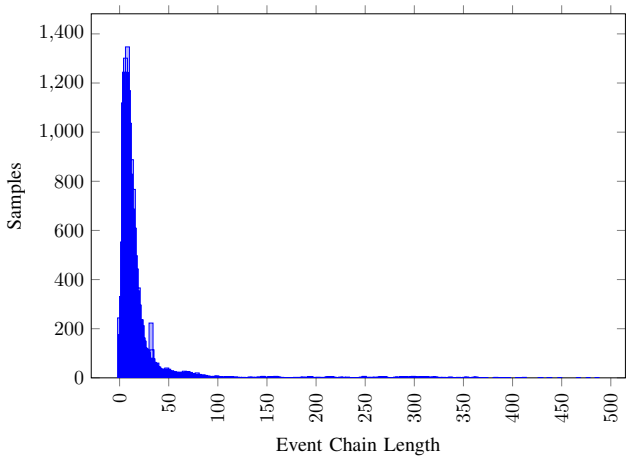

Fig. 4. Event Chain Length

number of samples with high event chain lengths is relatively low, as data points above 500 have been omitted from the histogram.

C. Event Correlation

Another interesting aspect to investigate is whether some features might be duplicates. Figure 5 illustrates the co-observations of unique events. The numbers represent the fraction of samples where feature x was observed together with feature y, relative to all samples where feature x was present. Here, x corresponds to the feature specified in the row, and y corresponds to the feature specified in the column.

The strongest co-observations are noted between the suspicious TLD, random process name, and processes running from suspicious locations. This correlation arises because these three events are among the most generic and cover the widest range of samples, as evidenced in Table VI. Additionally, the values on the diagonal indicate the fraction of samples where the suspicious event was the only one observed for a given sample.

The standard correlation matrix between features is presented in Figure 6. Similar to the co-observation matrix, there are no significant correlations observed among the features, with the exception of the unusual section name events.

D. Malware Clustering

As part of our initial experiments, we tested the clustering capabilities of our proposed method on the dataset. We pre-processed our dataset into 19-tuple vectors, where each position in the vector represented the count of specific events observed for each sample. For clearer visualizations, we limited the following experiments to the top five virus families presented in Table IV.

Despite the pre-processing omitting substantial information, such as specific suspicious values or the event sequence, Figure 7 demonstrates clearly visible clusters for various malware families. This suggests that, even with a limited amount of information compared to the full reports, distinct malware families tend to cluster based on similar behavior.

TABLE VII
CLASSIFICATION PERFORMANCE SUMMARY

Model	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)
Logistic Regression	0.84	0.84	0.77	0.79
Random Forest	0.90	0.91	0.86	0.88
SVM	0.71	0.70	0.48	0.51
KNN	0.87	0.87	0.83	0.85
Gradient Boosting	0.89	0.91	0.84	0.87
Neural Network (Multi Layer Perceptron (MLP))	0.89	0.90	0.83	0.86

Although these clusters are apparent through visual inspection, a critical question arises: Can these clusters be recognized by an automatic clustering algorithm? Due to the testing dataset lacking information about the event sequence—which will be included in the custom dataset—we opted to test traditional clustering algorithms to see if they could successfully identify any clusters related to the virus families shown in Figure 8.

However, none of the tested clustering algorithms were able to effectively cluster the dataset. The most promising results came from the GMM, but these results were still far from satisfactory. This leads us to conclude that although the malware patterns are visually identifiable, they cannot be automatically detected due to the lack of information. As a result, even though the clusters are distinguishable by visual inspection, they remain inseparable by automatic clustering methods.

E. Prediction Models

To verify our hypothesis, we employed several supervised learning algorithms to train a classification model. Given that various malware families form visible clusters, we anticipated that the classification task would yield successful results. We tested six algorithms without any parameter tuning. The results of this classification task are presented in Table VII.

Except for the SVM, all algorithms performed quite well. These findings suggest that malware can be effectively recognized based on our suspicious event set, as it exhibits distinct patterns.

VI. DISCUSSION

The initial part of the research aimed to explore various detection methods that can be included in our method.

The initial results of our proposed method indicates advantages over traditional malware detection techniques, particularly in handling complex and evasive malware. By prioritizing unreliable Indicators of Compromise and analyzing them through structured sequence analysis, our approach offers a balanced solution that combines both signature-based and behavior-based detection strengths, while minimizing their individual weaknesses.

	Direct IP Communication	Suspicious Extension	Randomly generated path	Randomly generated domain	Suspicious TLD	Using Geo API	SMTP Communication	Double extension	Using public storage	Exfiltrates user information	Download of exe file	Suspicious JA3 fingerprint	Random process name	Process from suspicious location	Firewall manipulation	Autorun modification	New task scheduled	Unusual entrypoint section	Unusual section name
Direct IP Communication	0%	57%	44%	9%	50%	5%	3%	2%	3%	9%	52%	3%	98%	84%	2%	15%	21%	16%	28%
Suspicious Extension	44%	0%	37%	13%	64%	20%	3%	15%	4%	1%	63%	5%	97%	87%	2%	21%	24%	13%	22%
Randomly generated path	53%	57%	0%	13%	57%	13%	1%	9%	5%	1%	42%	3%	95%	77%	2%	24%	22%	16%	27%
Randomly generated domain	11%	21%	13%	1%	57%	10%	11%	1%	2%	1%	16%	3%	95%	89%	2%	30%	26%	10%	22%
Suspicious TLD	19%	31%	18%	17%	1%	16%	4%	5%	2%	6%	25%	4%	97%	89%	2%	20%	23%	9%	22%
Using Geo API	9%	44%	19%	13%	73%	0%	3%	29%	2%	0%	15%	2%	98%	96%	1%	33%	28%	10%	24%
SMTP Communication	9%	13%	4%	28%	38%	6%	0%	2%	0%	0%	11%	1%	100%	99%	2%	24%	24%	4%	7%
Double extension	8%	97%	36%	6%	64%	87%	3%	0%	1%	0%	11%	0%	93%	91%	0%	10%	20%	5%	10%
Using public storage	28%	49%	37%	15%	52%	13%	1%	3%	0%	1%	7%	5%	99%	54%	2%	39%	19%	3%	7%
Exfiltrates user information	33%	3%	5%	3%	63%	0%	0%	0%	1%	0%	3%	0%	100%	98%	0%	4%	12%	9%	26%
Download of exe file	54%	85%	36%	13%	70%	9%	4%	2%	1%	1%	0%	6%	99%	94%	2%	16%	24%	14%	24%
Suspicious JA3 fingerprint	17%	42%	19%	14%	69%	6%	2%	0%	4%	0%	40%	2%	91%	58%	0%	12%	16%	1%	3%
Random process name	18%	23%	15%	14%	48%	11%	6%	3%	2%	5%	18%	3%	1%	92%	2%	26%	25%	14%	25%
Process from suspicious location	17%	23%	13%	14%	48%	12%	6%	4%	1%	5%	18%	2%	100%	0%	2%	27%	26%	15%	27%
Firewall manipulation	16%	21%	15%	14%	43%	4%	5%	0%	3%	0%	17%	0%	100%	100%	0%	61%	26%	11%	24%
Autorun modification	11%	19%	14%	17%	38%	14%	5%	1%	3%	1%	11%	1%	100%	96%	4%	0%	27%	18%	29%
New task scheduled	15%	23%	14%	15%	45%	12%	5%	3%	2%	2%	17%	2%	100%	96%	2%	27%	0%	10%	20%
Unusual entrypoint section	21%	22%	18%	11%	32%	8%	2%	1%	0%	3%	19%	0%	100%	98%	1%	33%	18%	0%	95%
Unusual section name	20%	21%	17%	13%	44%	11%	2%	2%	1%	5%	17%	0%	100%	98%	2%	30%	20%	52%	0%

Fig. 5. Feature Co-Observation

Higher false-positive rates of particular methods can result from unreliable indicators, yet by linking these indicators in a sequence, our approach verifies the context of suspicious activity. This helps mitigate false positives that are common in both signature-based methods (due to reliance on patterns not exclusive to malware) and behavior-based systems.

The results also highlight our method's potential in detecting polymorphic and stealthy malware, which can bypass static signature detection. By possibility of using less reliable and more generic static patterns and focusing on their sequence, our approach is positioned to recognize latent, evasive behaviors that are detectable only when examined within a structured sequence, enhancing its robustness against stealth tactics.

An attacker could try to avoid detection using our method by using various evasion tactics. One common strategy is to use obfuscation and polymorphism, where malware constantly

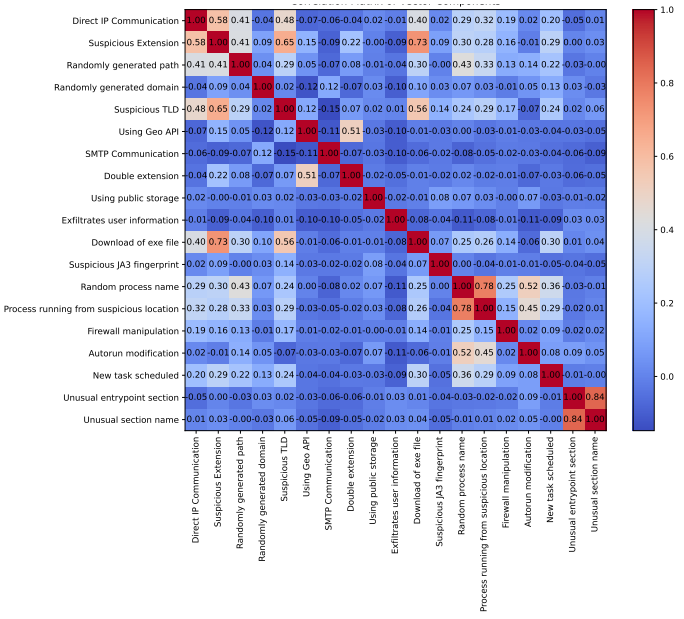

Fig. 6. Feature Correlation Matrix

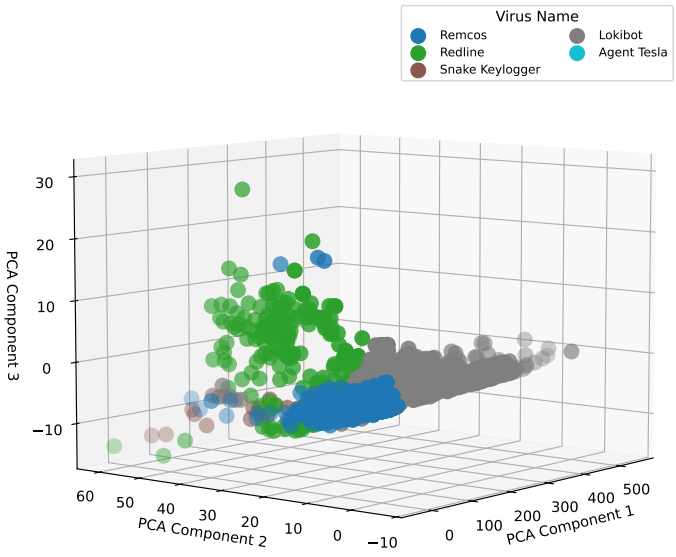

Fig. 7. Principal Component Analysis (PCA) Based on Virus Family

changes its structure or encrypts its payload to avoid behavior-based detection. Additionally, attackers can delay the execution of malicious actions or stagger them over time, making the sequence of suspicious events less obvious. Another method is mimicking legitimate behavior, where malware performs actions that resemble normal system processes and mixes in benign activities. Additionally, attackers could target specific aspects of our method by suppressing or avoiding certain suspicious events, such as not interacting with specific network addresses or registry keys flagged as indicators of compromise. Finally, they can manipulate data or use anti-analytical techniques, such as virtual environment detection, to bypass detection during analysis.

To mitigate these issues, our method emphasizes flexibility by incorporating dynamic behavior analysis and a sequence-based approach, rather than relying solely on predefined patterns. By evaluating not only individual events, but also the relationships and patterns between them, our system is more resistant to attempts to imitate legitimate behavior or delay actions. In addition, continuous updates to the event set and improvements to the similarity algorithms ensure that the detection process adapts to new malware tactics over time, making it more difficult for attackers to consistently evade detection.

A. Future Research Direction

The results presented in this paper summarize our ongoing efforts; however, much work remains to fully implement and refine our method.

1) Sandbox Environment: We are currently preparing a custom sandbox environment and developing the agent modules. This sandbox environment will enable us to create a dataset by running live malware samples and observing their behaviors.

2) Suspicious Event Set: To build a set of suspicious events, we analyze the MITRE framework, the Sigma repository, and malware samples. Our goal is to gather as comprehensive a collection of suspicious events as possible, while avoiding duplication and minimizing extensive event correlation to reduce resource demands.

3) Experiments and Tests: Once we obtain a collection of events from our environment, we will conduct tests and experiments using various sequence alignment methods.

4) Comparison with Existing Detection Methods: Evaluating and comparing our method with existing detection mechanisms is crucial. We will assess the efficacy and detection speed of our approach based on the execution of novel malware samples.

5) Dynamics of Event Chains: Understanding the sequence and evolution of events may allow us to predict the future behavior of malware and automatically generate preventive and counter actions. This approach could enable automatic remediation of infected systems without human intervention.

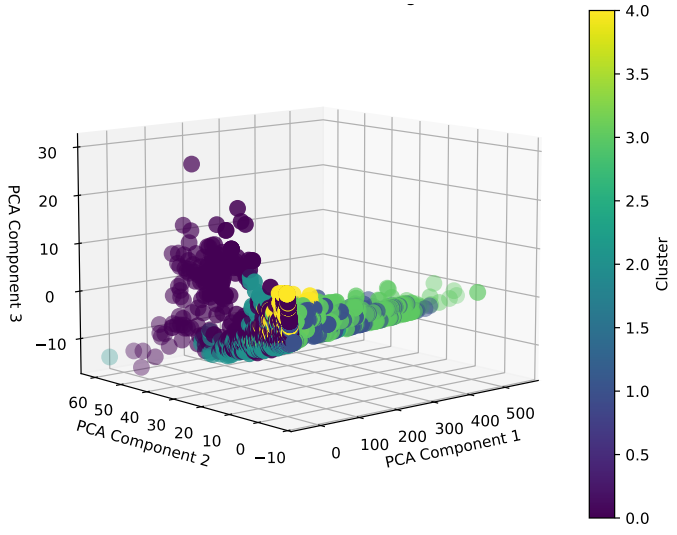

Fig. 8. PCA Based on Gaussian Mixture Model (GMM) Clustering

VII. Conclusion

In this study, we explored the potential of a novel approach to malware detection through the analysis of suspicious event chains derived from malware samples. By focusing on a carefully curated set of 19 suspicious events, we aimed to uncover insights into the behavior and clustering of different malware families. Our initial experiments revealed that the majority of samples exhibited between 3 to 5 unique suspicious event types, with a significant portion demonstrating a high event chain length. This indicates a rich behavioral signature, even within a limited feature set.

The co-observations and correlation analyses highlighted the relationships between various suspicious events, particularly among the most prevalent features. Despite the absence of temporal data, which may enhance future analyses, our findings suggest that distinct patterns emerge, enabling the possibility of clustering and classification.

Although visual inspections indicated the presence of clusters among different malware families, our attempts to employ traditional clustering algorithms were largely unsuccessful in automatically discerning these patterns. Conversely, our classification models demonstrated promising results, with most algorithms achieving high accuracy. This reinforces the hypothesis that malware behavior can be effectively distinguished through the identified suspicious events.

In conclusion, this research underscores the viability of using suspicious event analysis for malware detection and classification. Future work will involve expanding the event set, incorporating temporal data, and exploring more advanced machine learning techniques to enhance the detection capabilities further.

Acknowledgment

This research was funded by the Ministry of the Interior of the Czech Republic, Open challenges in security research, VK01030030, Data backup and storage system with integrated active protection against cyber threats.

REFERENCES

[1] P. Novak and V. Oujezsky, "Heuristic malware detection method based on structured cti data: A research study and proposal," in *2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2024. doi: 10.23919/SoftCOM62040.2024.10721992 pp. 1–6.

[2] P. T. S. Team, "Results of cybersecurity incident investigations in 2021 2023," https://www.ptsecurity.com/ww-en/analytics/results-of-cybersecurity-incident-investigations-in-2021-2023/, 12 2023, (Accessed 2024-05-02).

[3] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on windows malware detection: Techniques, research issues, and future directions," *Journal of Systems and Software*, vol. 209, p. 111921, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223003163

[4] J. Althouse. (2021, 5) Tls fingerprinting with ja3 and ja3s. Online. Available from: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967. (Accessed 2024-07-20).

[5] J. Althouse. (2023, 7) Ja4+ network fingerprinting. Online. Available from: https://blog.foxio.io/ja4+-network-fingerprinting. (Accessed 2024-07-20).

[6] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using http traffic characteristics," *Computers & Electrical Engineering*, vol. 66, pp. 353–368, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790617333542

[7] J. Liu, Q. Xiao, L. Xin, Q. Wang, Y. Yao, and Z. Jiang, "M3f: A novel multi-session and multi-protocol based malware traffic fingerprinting," *Computer Networks*, vol. 227, p. 109723, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623001688

[8] Y. Hong, Q. Li, Y. Yang, and M. Shen, "Graph based encrypted malicious traffic detection with hybrid analysis of multi-view features," *Information Sciences*, vol. 644, p. 119229, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025523008149

[9] G. Shenderovitz and N. Nissim, "Bon-apt: Detection, attribution, and explainability of apt malware using temporal segmentation of api calls," *Computers & Security*, vol. 142, p. 103862, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824001639

[10] Prachi., N. Dabas, and P. Sharma, "Malanalyser: An effective and efficient windows malware detection method based on api call sequences," *Expert Systems with Applications*, vol. 230, p. 120756, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423012587

[11] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "Api-maldetect: Automated malware detection framework for windows based on api calls and deep learning techniques," *Journal of Network and Computer Applications*, vol. 218, p. 103704, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804523001236

[12] (2014) Tracking malware with import hashing. Online. Available from: https://cloud.google.com/blog/topics/threat-intelligence/tracking-malware-import-hashing/. (Accessed 2024-07-23).

[13] T. Chen, H. Zeng, M. Lv, and T. Zhu, "Ctimd: Cyber threat intelligence enhanced malware detection using api call sequences with parameters," *Computers & Security*, vol. 136, p. 103518, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823004285

[14] E. Amer, I. Zelinka, and S. El-Sappagh, "A multi-perspective malware detection approach through behavioral fusion of api call sequence," *Computers & Security*, vol. 110, p. 102449, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740482100273X

[15] A. G. Kakisim, S. Gulmez, and I. Sogukpinar, "Sequential opcode embedding-based malware detection method," *Computers & Electrical Engineering*, vol. 98, p. 107703, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622000210

[16] S. Jeon and J. Moon, "Malware-detection method with a convolutional recurrent neural network using opcode sequences," *Information Sciences*, vol. 535, pp. 1–15, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520304217

[17] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Information Sciences*, vol. 231, pp. 64–82, 2013, data Mining for Information Security. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025511004336

[18] A. Pektaş and T. Acarman, "Learning to detect android malware via opcode sequences," *Neurocomputing*, vol. 396, pp. 599–608, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219304850

[19] S. Saha, S. Afroz, and A. H. Rahman, "Malign: Explainable static raw-byte based malware family classification using sequence alignment," *Computers & Security*, vol. 139, p. 103714, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824000154

[20] T. Rezaei, F. Manavi, and A. Hamzeh, "A pe header-based method for malware detection using clustering and deep embedding techniques," *Journal of Information Security and Applications*, vol. 60, p. 102876, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212621001046

[21] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware detection by eating a whole exe," in *AAAI Workshops*, 10 2017. doi: 10.48550/arXiv.1710.09435

[22] Q. Li, B. Zhang, D. Tian, X. Jia, and C. Hu, "Mdgraph: A novel malware detection method based on memory dump and graph neural network," *Expert Systems with Applications*, vol. 255, p. 124776, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424016439

[23] D. Tian, Q. Ying, X. Jia, R. Ma, C. Hu, and W. Liu, "Mdchd: A novel malware detection method in cloud using hardware trace and deep learning," *Computer Networks*, vol. 198, p. 108394, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621003728

[24] T. Panker and N. Nissim, "Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in linux cloud environments," *Knowledge-Based Systems*, vol. 226, p. 107095, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121003580

[25] G. Jacob, H. Debar, and E. Filiol, "Behavioral detection of malware: From a survey towards an established taxonomy," *Journal in Computer Virology*, vol. 4, pp. 251–266, 08 2008.

[26] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, and B. A. Saleh Al-rimy, "Toward an ensemble behavioral-based early evasive malware detection framework," in *2021 International Conference on Data Science and Its Applications (ICoDSA)*, 2021. doi: 10.1109/ICoDSA53588.2021.9617489 pp. 181–186.

[27] A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral detection of malware on mobile handsets," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '08. New York, NY, USA: Association for Computing Machinery, 2008. doi: 10.1145/1378600.1378626. ISBN 9781605581392 p. 225–238. [Online]. Available: https://doi.org/10.1145/1378600.1378626

[28] M. R. Amin, M. Zaman, M. S. Hossain, and M. Atiquzzaman, "Behavioral malware detection approaches for android," in *2016 IEEE International Conference on Communications (ICC)*, 2016. doi: 10.1109/ICC.2016.7511573 pp. 1–6.

[29] W. Liu, P. Ren, K. Liu, and H.-x. Duan, "Behavior-based malware analysis and detection," in *2011 First International Workshop on Complexity and Data Mining*, 2011. doi: 10.1109/IWCDM.2011.17 pp. 39–42.

[30] Y. Yang, Y. Lin, Z. Li, L. Zhao, M. Yao, Y. Lai, and P. Li, "Goosebt: A programmable malware detection framework based on process, file, registry, and com monitoring," *Computer Communications*, vol. 204, pp. 24–32, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366423000907

[31] O. Pluskal, "Behavioural malware detection using efficient svm implementation," in *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems*, ser. RACS '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2811411.2811516. ISBN 9781450337380 p. 296–301. [Online]. Available: https://doi.org/10.1145/2811411.2811516

[32] J. Stiborek, T. Pevný, and M. Rehák, "Multiple instance learning for malware classification," *Expert Systems with Applications*, vol. 93, pp. 346–357, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417307170

[33] J. Singh and J. Singh, "Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms," *Information and Software Technology*, vol. 121, p. 106273, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920300239

[34] A. Boukhtouta, N.-E. Lakhdari, S. A. Mokhov, and M. Debbabi, "Towards fingerprinting malicious traffic," *Procedia Computer Science*, vol. 19, pp. 548–555, 2013, the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050913006819

[35] L. Deri and F. Fusco, "Using deep packet inspection in cybertraffic analysis," in *2021 IEEE International Conference on Cyber Security*

and Resilience (CSR), 2021. doi: 10.1109/CSR51186.2021.9527976 pp. 89–94.
[36] (2022) Ja3 fingerprints database. Online. Available from: https://sslbl.abuse.ch/ja3-fingerprints/. (Accessed 2024-07-20).
[37] Mitre att&ck. Online. Available from: https://attack.mitre.org/techniques/enterprise/. (Accessed 2024-07-23).
[38] Sigma - generic signature format for siem systems. Online. Available from: https://github.com/SigmaHQ/sigma. (Accessed 2024-10-26).
[39] S. Chan, A. K. Wong, and D. K. Chiu, "A survey of multiple sequence comparison methods," *Bulletin of mathematical biology*, vol. 54, pp. 563–598, 1992.
[40] E. Paxinou, D. Kalles, C. T. Panagiotakopoulos, and V. S. Verykios, "Analyzing sequence data with markov chain models in scientific experiments," *SN Computer Science*, vol. 2, no. 5, p. 385, 2021.
[41] X. Liu and T. Cheng, "Video-based face recognition using adaptive hidden markov models," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1. IEEE, 2003, pp. I–I.
[42] S. Perdikaris, "A markov chain model in teachers' decision making," *International Journal of Mathematical Education in Science and Technology*, vol. 23, no. 3, pp. 473–477, 1992.
[43] A. K. Saw, B. C. Tripathy, and S. Nandi, "Alignment-free similarity analysis for protein sequences based on fuzzy integral," *Scientific reports*, vol. 9, no. 1, p. 2775, 2019.
[44] K. Riesen, "Structural pattern recognition with graph edit distance," *Advances in computer vision and pattern recognition*, pp. 1–164, 2015.
[45] M. Tantardini, F. Ieva, L. Tajoli, and C. Piccardi, "Comparing methods for comparing networks," *Scientific reports*, vol. 9, no. 1, p. 17557, 2019.
[46] M. K. Gupta, G. Gouda, N. Rajesh, R. Donde, S. Sabarinathan, P. Pati, S. K. Rathore, R. Vadde, and L. Behera, "Sequence alignment," *Bioinformatics in Rice Research: Theories and Techniques*, pp. 129–162, 2021.
[47] M. Chatzou, C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, "Multiple sequence alignment modeling: methods and applications," *Briefings in Bioinformatics*, vol. 17, no. 6, pp. 1009–1023, 11 2015. [Online]. Available: https://doi.org/10.1093/bib/bbv099
[48] K. Alotaibi, R. Rob, D. Nour, and D. Zamzami, "Detecting subspace malicious vectors attack against smart grid using sequence-alignment method," in *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2023. doi: 10.1109/CSR57506.2023.10224936 pp. 367–372.
[49] M. S. Rosenberg, *Sequence Alignment: Methods, Models, Concepts, and Strategies*, 1st ed., M. S. Rosenberg, Ed. University of California Press, 2009. ISBN 9780520256972. [Online]. Available: http://www.jstor.org/stable/10.1525/j.ctt1pps7t (Accessed 2024-10-23).
[50] D. Ofer, N. Brandes, and M. Linial, "The language of proteins: Nlp, machine learning & protein sequences," *Computational and Structural Biotechnology Journal*, vol. 19, pp. 1750–1758, 2021.
[51] H. Li and B. Liu, "Bioseq-diabolo: biological sequence similarity analysis using diabolo," *PLoS computational biology*, vol. 19, no. 6, p. e1011214, 2023.
[52] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3289600.3290967. ISBN 9781450359405 p. 384–392. [Online]. Available: https://doi.org/10.1145/3289600.3290967
[53] X. Xu, C. Liu, Q. Feng, H. Yin, L. Song, and D. Song, "Neural network-based graph embedding for cross-platform binary code similarity detection," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3133956.3134018. ISBN 9781450349468 p. 363–376. [Online]. Available: https://doi.org/10.1145/3133956.3134018
[54] A. Nair, A. Roy, and K. Meinke, "funcgnn: A graph neural network approach to program similarity," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ser. ESEM '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3382494.3410675. ISBN 9781450375801. [Online]. Available: https://doi.org/10.1145/3382494.3410675
[55] V. Likic, "The needleman-wunsch algorithm for sequence alignment," *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, pp. 1–46, 2008.
[56] F. Zhang, X.-Z. Qiao, and Z.-Y. Liu, "A parallel smith-waterman algorithm based on divide and conquer," in *Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings.* IEEE, 2002, pp. 162–169.
[57] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-Rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware detection issues, challenges, and future directions: A survey," *Applied Sciences*, vol. 12, no. 17, p. 8482, 2022.
[58] "Joe sandbox," Online, available from: joesandbox.com.
[59] Gitlab fi muni - vsafe-experiments. Online. Available from: https://gitlab.fi.muni.cz/xnovak7/vsafe-experiments/-/tree/main. (Accessed 2024-10-26).
[60] Why is there's so much spam coming from .xyz and other new top-level domains? Online. Available from: https://blog.f-secure.com/why-is-theres-so-much-spam-coming-from-xyz-and-other-new-top-level-domains/. (Accessed 2024-07-23).
[61] A peek into top-level domains and cybercrime. Online. Available from: https://unit42.paloaltonetworks.com/top-level-domains-cybercrime/. (Accessed 2024-07-23).

Pavel Novak finished his master's studies focusing on cybersecurity in 2022 at the Faculty of Informatics of Masaryk University in Brno. He is currently a Ph.D. candidate at Masaryk University focusing on early stage behavioral malware detection. He contributes to various research projects, leads student theses, and serves as a seminar tutor. He is passionate about cyber threats and threat hunting. He also participates in public relations contributions regarding new web-based cyber threats.

Vaclav Oujezsky studied teleinformatics at the Brno University of Technology. Between 2013 and 2017, he completed a Ph.D. in the same field. He served as a Researcher in the Department of Telecommunications at the Brno University of Technology. Since 2021, he has held the position of Associate Professor at Masaryk University. Over the years, he has earned several certifications, contributed to numerous grant projects, and co-authored a wide range of scientific articles and conference papers. He is recognized as an expert in the field of modern information systems.