Anomaly Detection in TCP/IP Networks

Joanna Kołodziej
Department of Computer Science
Cracow University of Technology
ul. Warszawska 24, 31-155 Kraków, Poland
Email: joanna.kolodziej@pk.edu.pl

Mateusz Krzysztoń
Research and Academic Computer Network
ul. Kolska 12, 01-045 Warsaw, Poland
Email: mateuszkr@nask.pl

Paweł Szynkiewicz
Research and Academic Computer Network
ul. Kolska 12, 01-045 Warsaw, Poland
Email: pawel.szynkiewicz@nask.pl

ABSTRACT

Intrusion Detection Systems (IDS) should be capable of quickly detecting attacks and network traffic anomalies to reduce the damage to the network components. They may efficiently detect threats based on prior knowledge of attack characteristics and the potential threat impact ('known attacks'). However, IDS cannot recognise threats, and attacks ('unknown attacks') usually occur when using brand-new technologies for system damage.

This paper presents two security services – Net Anomaly Detector (NAD) and a signature-based PGA Filter for detecting attacks and anomalies in TCP/IP networks. Both services are modules of the cloud-based GUARD platform developed in the H2020 GUARD project. Such a platform was the main component of the simulation environment in the work presented in this paper. The provided experiments show that both modules achieved satisfactory results in detecting an unknown type of DoS attacks and signatures of DDoS attacks.

KEYWORDS

Anomaly Detection; Machine Learning; Cybersecurity; TCP/IP

I. Introduction

The rapid development of research, technology and information tools for communication and control systems, sensor networks and the processing of enormous data sets has contributed to a real revolution in IT support to manage real-life engineering and smart systems. Despite the benefits of implementing numerous Internet-based models to support engineering systems, such systems must usually cope with the secure processing of streaming time-series data induced by connected real-time data sources. Examples of streaming data sources include sensors in transportation systems, e-health systems, smart infrastructures, intelligent cars, monitoring systems, and many others. The collected data can be processed locally using the resources and IT infrastructure of the institution (customer) directly responsible for local data management. The data, metadata, and initial analysis results are often sent to external systems (e.g., cloud computing) that can analyze the data more closely and send alerts on potential threats and anomalies.

This paper focuses on the attacks, threats and anomaly detection in the TCP/IP networks. We present the main concept of two security-service components of the GUARD platform developed in the H2020 GUARD project [1], namely Net Anomaly Detector (NAD) and PGA-Filter. GUARD platform was used as a core of the simulation environment used in the research presented in the paper. The overall GUARD architecture is a typical structure of the Security Events and Information Management (SIEM) system[2]. It was designed in a service-oriented way and takes into account the presence of heterogeneous technical and administrative domains.

Successful signature-based detection of DDoS attacks requires a source of high-quality, up-to-date network traffic signatures. We developed the Packet Generation Algorithm (PGA) Filter. PGA is deployed as an agent in the GUARD environment at the edge of the protected network, and it translates provided signatures into packet filtering rules. The rules are then applied to the client's network configuration concerning incoming traffic and hardening the network ingress security infrastructure. The signatures can be regularly updated based on the information obtained from the signature generator through the signature share service. We used PGA Filter as a supporting agent in detecting the DDoS signatures in the simulated TCP/IP traffic.

Detection of unknown attacks is challenging due to the lack of exemplary attack vectors. However, unknown attacks are a significant danger for systems due to a lack of tools for protecting systems against them. The most widely used approach for malicious behaviour of the monitored system is detecting anomalies. The developed NAD module records regular traffic in the system and creates a set of models of regular traffic using Machine Learning (ML) algorithms. NAD selects the optimal model from that set based on the given criteria or combines models with one of the proposed strategies (i.e. ensemble model). Such an op-

[1] `https://guard-project.eu/`
[2] `https://www.ibm.com/topics/siem`

Communications of the ECMS, Volume 37, Issue 1,
Proceedings, ©ECMS Enrico Vicario, Romeo Bandinelli,
Virginia Fani, Michele Mastroianni (Editors) 2023
ISBN: 978-3-937436-80-7/978-3-937436-79-1 (CD) ISSN 2522-2414

timal model is then used to detect anomalies in the simulated network traffic. NAD achieved satisfactory results in detecting an unknown type of DoS attack. The experiments were carried out on the CIC-IDS2017 dataset[3].

The rest of the paper is organized as follows. In Sec. II, the backgrounds of known and unknown attacks and anomalies are presented. GUARD project and platform are briefly presented in Sec. III. Sec. IV and Sec. V present the concepts of two GUARD modules: NAD and PGA Filter, which were experimentally evaluated in Sec. VI. The paper is summarized in Sec. VII.

II. Detection of anomalies in ICT systems

These days, it's hard to imagine an intelligent system without IT support. However, the use of such support carries the risk of numerous anomalies. The implementation of IT tools is exposed to hacking attacks, which can result in great difficulties in managing the intelligent system and be dangerous for its users.

Anomalies in ICT systems are monitored by dedicated software and can have various causes. They can result from failures, overloading or ineffective management of the specific infrastructure associated with these systems. The occurrence of anomalies can be the result of external attacks on networks and information systems.

The classification of threats (attacks) is usually based on the classification of the threat techniques, recognition of the attack's type (known - recognized or unknown), and threat impact [13]. This paper focuses on the two attack categories: 'known' and 'unknown' attacks.

A. Signature-based detection methods of 'known' attacks

Most methodologies for detecting known threats are signature-based (SB) techniques. Such methods mainly aim to compare suspicious payloads with signatures of specific known attacks. These signatures can correspond to data types, such as byte sequences in network traffic, known malicious instruction sequences used by malware, etc. Signature schemes assume that patterns can define malware.

Despite the popularity of signature-based methods, there are several significant drawbacks to this approach:

- **Vulnerability to evasion** – signature patterns (bytes) from known attacks are – as the name implies – universally known. The use of obfuscation techniques or polymorphic methods [14], popular in malware, allows known signatures to be dropped. In the case of network attacks or exploits, bugs or vulnerabilities found in software are exploited. Specific application protocols generally limit the scope of such attacks.
- **Zero–day attacks** – signature analysis-based attack detection methods cannot effectively detect polymorphic malware. This means that SD does not provide zero-day protection. Signature-based detectors use different signatures for each malware variant. As a result, the volume of the signature database is generally very large, and when new signature variants are generated – it grows exponentially.

[3]`https://www.unb.ca/cic/datasets/ids-2017.html`

Signatures can be generated manually by experts. In this case, the experts must analyze the attack and identify the invariant fragments in the involved flows, using their knowledge of the attacked application and the exploited vulnerability. They also construct a signature that fully identifies the threat thanks to their detailed knowledge. Such signature generation is a time-consuming process. Several provided experiments indicate that during signature generation, more than 90% vulnerable systems can be infected at that time. In current anomaly detection systems, signatures are generated automatically using dedicated IT tools. These methods search for common features of suspicious flows not seen in regular, benign traffic. Several systems for automatic generation of signatures of zero–day polymorphic worms have been developed: Autograph [6], Polygraph [11], Nebula [20], Hamsa [22], Lisabeth [5]. Most of them use relatively simple (computationally inexpensive) heuristic approaches. Another method is defined in [15], where the generation of multi-set signatures is formulated as an optimization problem. A specialized version of the genetic algorithm (GA) is used to solve it.

B. Detection methods of unknown threats and attacks

Unknown threats and attacks are not recognized by signature-based methods based on the accumulated knowledge of attacks. One possible reason is that the attacker may use new methods or technologies. 'Unknown threats' are referred to as anomalies. The following types of anomalies can indicate malicious system behaviour [21]:

- **Point anomaly** is the simplest form of anomaly and denotes an anomalous single event (outlier). It can be caused by defining a strange (unexpected) login variable or IP address.
- **Contextual anomaly** is an event anomalous in a certain context but may be normal in another. Such an anomaly occurs, for example, when an employee logs into the system outside of working hours. Such an event would not be classified as an anomaly during normal working hours.
- **Collective/frequency anomaly** is usually characterized by the anomalous frequency of single normal events. This can be a database dump in computer networks, which SQL can cause–injection.
- **Sequential anomaly** represents an anomalous sequence of events classified as normal. An anomalous sequence can be caused in a data communications network, for example, by violating the access chain.

The anomaly detection problem in distributed ICT systems can be solved through the analysis of data and information flow monitoring results in these systems. Anomaly detection methods must adapt to system architecture and configuration changes and analyze large amounts of data and information transmitted and generated by devices integrated with the computer system.

Machine Learning (ML) techniques successfully detect unknown threats, attacks and anomalies. The following popular ML and statistical methods are commonly used in intelligent anomaly detection systems: Artificial Neural Networks (ANN) [10], Bayesian Networks [7], Decision Trees [16], Hidden Markov Models (HMM) [12] and Support Vector Machines (SVM) [9].

III. GUARD platform

The main goal of the GUARD project was to develop and implement a programmable platform that could mediate between monitoring and inspection tasks in digital services and algorithms for detecting anomalies in those systems.

The GUARD framework is conceived as a new paradigm for implementing detection and analytics processes for digital service chains[4]. Such chain can be composed of the following components:

- a platform that orchestrates security capabilities by discovering, configuring and connecting them into security analytic pipelines (SAPs);
- a set of digital services implemented using multi-agent systems;
- detection and analytics services for discovering attacks and anomalies throughout the service chain.

Fig. 1 presents the GUARD software architectural model. We used this platform as a core of the simulation environment in the research presented in this paper. We developed two security service modules, Net Anomaly Detector and PGA Filter. The prototypes of those modules are presented as Algo1 and Security Agent1 components in the GUARD model presented in Fig. 1.

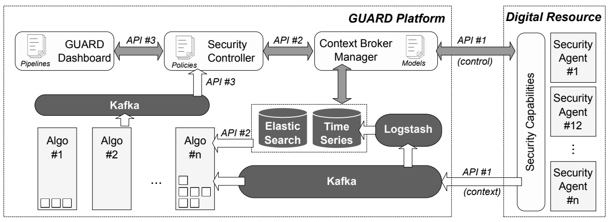

Fig. 1. GUARD software architecture model.

IV. Net Anomaly Detector (NAD)

Net Anomaly Detector (NAD) was designed as a highly modular security service component of the GUARD platform. The architectural model of NAD is presented in Fig. 2.

The core component of NAD is the *Model Generator (MG)*. The input data for MG are feature vectors of observed network traffic represented numeric vectors. Examples of such samples are the flow in TCP/IP or characteristic of the traffic recorded in a time window. The MG module automatically generates the *anomalies detector*, based on the observed, benign traffic and

[4]https://jitcomputing.wordpress.com/digital-service-chains/

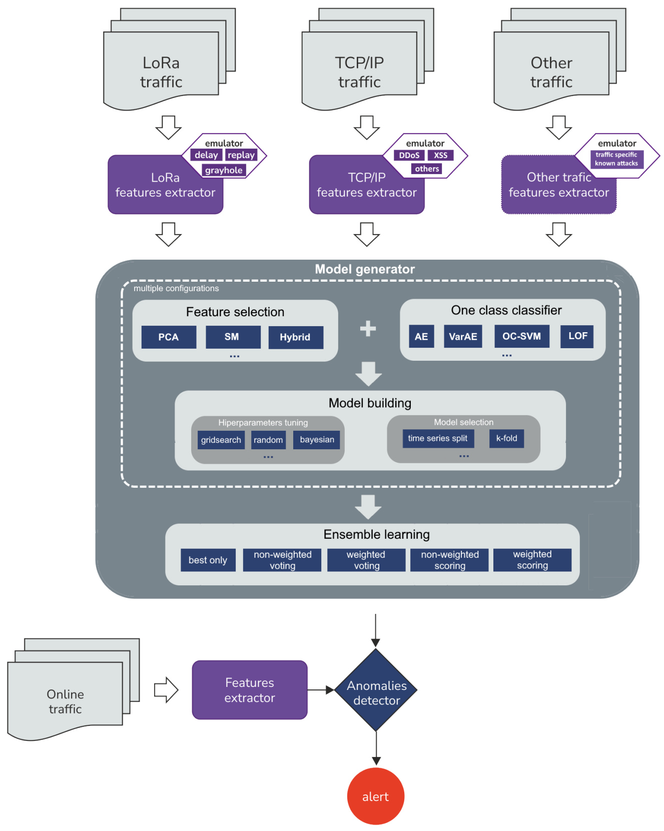

Fig. 2. The NAD modular architecture. Two types of networks: LoRa [4] and TCP/IP are specified, but the approach is easily extendable for other types of traffic.

emulation techniques. In the MG a library, the *Feature Selection (FS)* and *One-Class Classification (OCC)* algorithms are available. For each possible combination of FS and OCC, an optimal model for anomaly detection is built with automatic hyperparameters optimization (e.g. using Tree-structured Parzen Estimator strategy [2]) and model selection techniques. The result is several anomaly detectors. Finally, the ensemble strategies submodule combines multiple models into one to increase detection reliability.

The optimal model generated by the MG module analyses traffic in the monitored network after processing the traffic data by the *Feature Extractor*, responsible for generating features describing network traffic. Feature extractor in NAD is based on versatile method [23]. The main idea of that method is to generate as many features as possible using simple statistical measures as aggregation functions. The following aggregation functions are implemented in NAD:

- mean, minimal and maximal values,
- range (difference between the maximal and minimal values),
- sum of squared values (mean power),
- standard deviation,
- skewness,
- kurtosis,
- the 5th central moment,
- maximal difference between two consecutive mea-

surements,
- autocorrelation,
- count of the given value (e.g. TCP in the case protocol field).

Each type of network can describe each message by several attributes. For example, in the LoRa network, each message has attributes such as RSSI (Received Signal Strength Indication), SNR (Signal-to-Noise Ratio) or payload length. The values of these attributes can be aggregated with the above functions for all messages within a given time window to create a feature vector related to that window.

The easiest verification method of the effectiveness of NAD is to run it on normal traffic data and data infected by the *Attack emulator* embedded in NAD. Such an emulator contains a library of attacks specific to the given type of network. Each attack with different parameters (configurations) is saved in the library as its new element. In this way, the library can be constantly enriched with new data and expand the capabilities of the emulator. Extending the attack list increases the genericity of the verification data set. Thus the quality of the detection model in the unknown attack detection task.

V. PGA Filter Module

PGA Filter, developed as one of GUARD's security modules, is a self-contained intrusion detection system (IDS) that employs botnet fingerprinting techniques [1] to target distributed denial-of-service attacks (DDoS). It is a novel approach to signature-based detection of botnet-originating cyberattacks based on automated packet spoofing with packet generation algorithms (PGA) [8].

Our contribution is a complete ecosystem, which includes the definition of a new signature paradigm, the development of a signature generation process, and the implementation of software able to translate signatures into packet filtering rules to apply them to network traffic in a scalable manner.

PGA signatures describe patterns observed in the headers of packets generated by automated tools or scripts that are a part of botnet software. The procedure behind packet generation, also known as the packet generation algorithm (PGA), has distinct characteristics that can be observed in the packet header. Because PGAs are attack-specific and typically botnet-specific, they can be used to identify attacks and attackers (fingerprinting). As such, botnet attacks generated with PGA usually include fixed patterns in their malicious packets. Patterns can be identified by analysing single or multiple bytes of particular protocol fields in the packet header (e.g. TCP sequence number or IP destination address) that are deterministically dependent. A simple example of a pattern is the equality of the destination IP address and TCP sequence number. The pattern is often observed during a port scanning attack, where the value of the IP destination address is reused to speed up the packet creation process.

Extracting PGA signatures involves reverse engineering of the packet generation algorithm, which requires many suspicious traffic data. In our case, the data is provided by NASK's Network Telescope[5] (also a black hole, Internet sink, darkspace, darknet), which is an unused space of IP addresses used exclusively for passive monitoring [3]. Unused IP addresses should receive no legitimate network traffic. Therefore, all arriving unsolicited and anomalous traffic is, by definition, classified as suspicious. Network Telescope provides a view of a wide range of events taking place in a global network, including backscatter from denial-of-service attacks. Backscatter is an accumulation of victims' responses to DoS packets with a spoofed source IP address, which falls within the Network Telescope address space. Closely examining backscatter packets can reveal certain characteristics and similarities between the header values.

To obtain PGA signatures, packets sharing the same source and close arrival times are grouped together. Next, the possible fields of the original DoS attack package are partially recreated. This can involve multiple potential scenarios, depending on the attack type assumed. In case of a TCP SYN Flood attack the source and destination values for IP address and TCP port are swapped, and the TCP sequence number is set to the decremented TCP acknowledgement number of a backscatter packet. Finally, after all the previous steps, the most challenging aspect of this process is determining whether recreated packets share dependencies between protocol field values that follow the same pattern. Indeed, if applying a set of specific bitwise operations to packet headers reveals sequences of repeating bits, it gives the premise that packets were created with the same packet generation algorithm (PGA). More detailed descriptions of the process can be found in publications [19, 18]. An example PGA signature in a descriptive format is presented below.

"The first two bytes (0, 1) of the Source IP Address are equal to the first two bytes of TCP Sequence Number and the last two bytes (2, 3) of Destination IP Address are equal to the last two bytes of TCP Sequence Number."

Alternative representation in a less verbose, proposed PGA syntax:

```
ip-src:0:1 is tcp-seq:0:1 and
ip-dst:2:3 is tcp-seq:2:3
```

Implementing mechanisms to interpret PGA signatures and translate them into system-compatible rules that could be deployed in clients' network security infrastructure proved challenging. Well-known signature IDS/IPS solutions (Snort[6], Suricata[7], etc.) focus on analyzing patterns in the payload data rather than the dependencies between values in protocol headers. Hence a custom solution was required. A standard application that analyzes network traffic in user

[5]https://sissden.eu/blog/darknet-report
[6]https://www.snort.org/
[7]https://suricata.io/

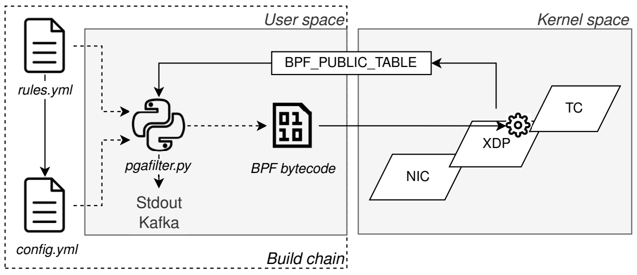

Fig. 3. Architecture of the PGA Filter module.

space would not meet the performance requirements. In this regard, the proposed implementation of PGA Filter leverages eBPF[8] (extended Berkeley Packet Filter). This virtual machine-like construct extends the standard kernel in Unix-like systems with custom functionality. Furthermore, using the XDP (eXpress Data Path) framework, which allows for high-speed packet processing within BPF applications, PGA signatures are applied even before kernel network stack allocation.

The architecture of the PGA Filter is presented in Fig. 3. The implementation resides in `pgafilrer.py` Python script responsible for parsing configuration, generating and loading BPF code, and gathering and sending results. The management is split into two files: `rules.yml` and `config.yml`. The first contains the declaration of available PGA signatures and general runtime configuration. PGA signatures are translated into valid BPF code and loaded directly into the kernel. BPF application inspects every packet arriving at the chosen network interface. Matched packets can be blocked, redirected or allowed through, depending on the configuration. The signature set can be changed anytime. Thus a new BPF code is generated and swapped in one atomic operation. The inspection of network packets is performed continuously and without delay throughout the process.

VI. Experimental evaluation

In this section, we present the results of the empirical evaluation of both developed modules – NAD and PGA Filter – in a GUARD platform. The network traffic was simulated by using the popular benchmarks presented below.

A. Unknown attack detection in TCP/IP network

The Intrusion Detection Evaluation Dataset [17] (CIC-IDS2017) was used to experiment. The experiment aimed to verify the quality of the NAD component in the task of an unknown type of DoS attack detection. The training dataset comprised benign traffic (40,815 TCP/IP flows). In the hyperparameters tuning and model selection phase, the validation set included benign flows and flows tagged with one of DoS slow loris, DoS Slowhttptest and DoS GoldenEye (10,203 benign and 10,203 malicious flows), while the test set consisted of the same number of benign flows and DoS Hulk (10,293 each). The following combinations of FS and OCC algorithms were tested:

- Autoencoder (no FS)
- Variational Autoencoder (no FS)
- PCA + LOF
- PCA + One Class SVM
- Simple Measures + One Class SVM

The *accuracy* metric was used in the optimisation process. In the described scenario, the best quality was achieved by a model built with PCA with Once Class SVM with the following values of tuned hyperparameters: $kernel = rbf$, ν=0.01, $\gamma = scale$, $max - iter = 10000$. The detailed results of this model on both validation and test set are compared in Table. I.

TABLE I: The results of a model built with PCA with Once Class SVM algorithms for feature selection and model building.

	validation dataset	test dataset
f1	0.78	0.57
f0.5	0.89	0.75
f2	0.69	0.46
accuracy	0.82	0.69
precision	0.98	0.95
npv	0.74	0.62
recall	0.64	0.41
specificity	0.99	0.98

The model achieves worse results on the task of unknown attack detection, which is expected as the model was tuned for validation set (known attacks) characteristics. However, the high value of precision (0.98 and 0.95) and specificity (0.99 and 0.98) is noteworthy — the typical anomaly detection system's high rate of false positive errors is not the case for the model (for balance datasets). Despite the degradation of the model quality for the test set, the results are still relatively high for detecting unknown attacks.

B. Known attack detection and mitigation in TCP/IP network

Effective signature-based detection of DDoS attacks relies entirely on the availability of specialized, high-quality, and current network traffic signatures. Since PGA Filter detects attacks purely deterministically, measuring its success rate is redundant. The process of PGA signature generation is semi-automatic and still under development. As for the quality of PGA signatures, due to their unique nature and lack of reference solutions, it is difficult to assess. To truly determine the effectiveness and usability of our solution, long-term tests on real-life data are required, but since this is still a prototype phase, they are yet to be performed.

However, in the case of signature-based solutions, performance plays no less of a role than detection quality. During a DDoS attacks, IDS must be able to process high traffic volumes so as not to be a bottleneck for the whole security system. Thus, a simulation environment was prepared to measure possible throughput in a simplified scenario of a volumetric DDoS attack. A testing network traffic was prepared. Roughly

[8]`https://ebpf.io/`

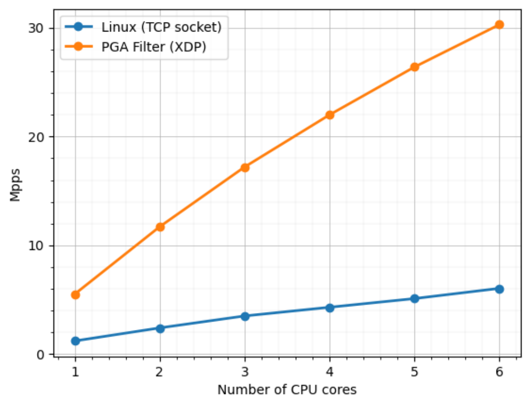

Fig. 4. DDoS filtering throughput.

50% of the network packets were generated as malicious (DDoS attack), each of them matching one of 10 possible TCP/IP PGA signatures. Also, a simple user space application was developed as a reference solution to compare results. It leverages standard TCP/IP sockets, and is capable of applying PGA signatures to filter the incoming traffic. Both PGA filter and reference solution were deployed on a bare metal server with a 100 Gb/s NIC and Intel Xeon E5-1650v4 6 core, 4GHz CPU. Testing traffic was replayed to reach the maximum bandwidth. Statistics were gathered to get an average throughput in packets per second. To determine the scalability, tests for each solution were performed with the limitations on available CPU cores. As shown in Fig. 4 results of PGA Filter leveraging eBPF and XDP are better than the naive reference solution. Also, our solution scales well with the number of cores.

VII. Conclusions

In this paper, we presented the NAD Security Service module and PGA Filter component in the GUARD software architectural model. We focused on detecting unknown DoS attacks and DDoS signatures as known attacks in TCP/IP traffic.

In signature-based DDoS detection, we focused on the traffic originating from botnets or other malicious software employing PGA (Packet Generation Algorithm) mechanisms. We defined a novel PGA signature paradigm and developed the new custom signature-based Intrusion Detection System employing SotA Linux Kernel technologies (eBPF + XDP). Our method is easily deployable on any Linux-based system. It offers support for network card drivers or NIC offloading for better performance. The platform's operation can also be freely extended beyond its DDoS detection capabilities by using provided signature language syntax.

Net Anomaly Detector can be easily modified by adding (or removing) the detection algorithms. The component's performance was validated on a widely used TCP/IP traffic dataset with several types of DoS attacks. The repeatability of the NAD component in the given scenario was high. We showed that identifying malicious TCP/IP flows is good enough to detect attack occurrences in the monitored network. The detector can also mitigate the attack by dropping malicious connections (although with some detriment to some regular users). More diverse datasets should be tested in the future.

REFERENCES

References

[1] Piotr Bazydło, Krzysztof Lasota, and Adam Kozakiewicz. "Botnet Fingerprinting: Anomaly Detection in SMTP Conversations". In: *IEEE Security Privacy* 15.6 (2017), pp. 25–32 (cit. on p. 4).

[2] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011) (cit. on p. 3).

[3] Nevil Brownlee. "One-Way Traffic Monitoring with iatmon". In: *Passive and Active Measurement.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 179–188 (cit. on p. 4).

[4] Anders Carlsson et al. "Measuring a LoRa network: Performance, possibilities and limitations". In: *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 18th International Conference, NEW2AN 2018, and 11th Conference, ruSMART 2018, St. Petersburg, Russia, August 27–29, 2018, Proceedings 18.* Springer. 2018, pp. 116–128 (cit. on p. 3).

[5] L. Cavallaro et al. "Lisabeth: Automated content–based signature generator for zero–day polymorphic worms". In: *Proc. of the 4th International Workshop on Software engineering for secure systems SESS'08.* 2008, pp. 41–48 (cit. on p. 2).

[6] Kim H.A. and Karp B. "Autograph: Toward Automated, Distributed Worm Signature Detection". In: *Proc. of the 13th conference on USENIX Security Symposium, (SSYM'04).* USENIX Association, Berkeley, CA, USA, 2004 (cit. on p. 2).

[7] D. Hackermann and et.al. "A tutorial on learning with Bayesian networks". In: *NATO ASI Series D Behavioural And Social Sciences* 89 (1998), pp. 301–354 (cit. on p. 3).

[8] Nazrul Hoque, Dhruba K. Bhattacharyya, and Jugal K. Kalita. "Botnet in DDoS Attacks: Trends and Challenges". In: *IEEE Communications Surveys Tutorials* 17.4 (2015) (cit. on p. 4).

[9] Steinwart I. and Christmann A. *Support vector machines.* Springer Science & Business Media, 2009 (cit. on p. 3).

[10] Cannady J. "Artificial neural networks for misuse detection, In: National information systems security conference". In: *Proc. of National information systems security conference.* 1998, pp. 368–381 (cit. on p. 3).

[11] Newsome J., Karp B., and Song D. "Polygraph: Automatically Generating Signatures for Poly-

morphic Worms". In: *Proc. of the IEEE Symposium on Security and Privacy (S&P'05)*. IEEE Computer Society: Los Alamitos, US., 2005, pp. 226–241 (cit. on p. 2).

[12] Baum L.E. and Eagon J.A. "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". In: *Bulletin of the American Mathematical Society* 73(3) (1967), pp. 360–363 (cit. on p. 3).

[13] Pawar M. and Anuradha J. "Network Security and Types of Attacks in Network". In: *Proc. of the International Conference on Computer, Communication and Convergence (ICCC 2015)*. Vol. 48. Procedia Computer Science, 2015, pp. 503–506 (cit. on p. 2).

[14] Sounak P. and Mishra B. K. "Survey of polymorphic worm signatures". In: *International Journal of u- and e- Service. Science and Technology* 7 (2014), pp. 129–150 (cit. on p. 2).

[15] Szynkiewicz P. and Kozakiewicz A. "Design and Evaluation of a System for Network Threat Signatures Generation". In: *Journal of Computational Science* 22 (2017), pp. 187–197 (cit. on p. 2).

[16] Safavian S.R. and Landgrebe D. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21(3) (1991), pp. 660–674 (cit. on p. 3).

[17] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." In: *ICISSp* 1 (2018), pp. 108–116 (cit. on p. 5).

[18] SISSDEN. *Deliverable D2.7: Final Dissemination Report*. `https://sissden.eu/download/SISSDEN-D2.7-Final_Dissemination_Report.pdf`. [Online; accessed 13–March-2023]. 2019 (cit. on p. 4).

[19] Paweł Szynkiewicz. "Signature-Based Detection of Botnet DDoS Attacks". In: *Cybersecurity of Digital Service Chains: Challenges, Methodologies, and Tools*. Springer International Publishing, 2022, pp. 120–135 (cit. on p. 4).

[20] Werner T. et al. "Nebula – Generating Syntactical Network Intrusion Signatures". In: *Proc. of the 4th International Conference on Malicious and Unwanted Software (MALWARE)*. 2009, pp. 31–38 (cit. on p. 2).

[21] Chandola V., Arindam B., and Vipin K. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15 (cit. on p. 2).

[22] Li Z. et al. "Hamsa: Fast Signature Generation for Zero–Day Polymorphic Worms with Provable Attack Resilience". In: *Proc. of the IEEE Symposium on Security and Privacy (S&P'06)*. 2006 (cit. on p. 2).

[23] Adam Zagorecki. "A versatile approach to classification of multivariate time series data". In: *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2015, pp. 407–410 (cit. on p. 3).

BIOGRAPHIES

JOANNA KOŁODZIEJ is the Professor of Computer Science at the Research and Academic Computer Network (NASK) Institute in Warsaw. Prof. Kolodziej is the President of the Polish Chapter of the IEEE Computational Intelligence Society. She participated in several international and national projects and is a member of the EU Blockchain Partnership Society. Her research is focused on machine learning and Big Data, security aspects and energy awareness in resource management and data scheduling in clouds, cloud and edge computing, cybersecurity aspects in ICT systems and recently – blockchain technologies.

MATEUSZ KRZYSZTOŃ is an assistant professor at the Research Academic Computer Network (NASK). He received a PhD in technical information technology and telecommunications at the Warsaw University of Technology in 2020. He is Head of Distributed System Group, which researches the Internet of Things, machine learning and cybersecurity.

PAWEŁ SZYNKIEWICZ is a cybersecurity systems architect at the Research Academic Computer Network (NASK). He works at the Network Security Systems Department, tasked with developing and maintaining systems for national security and the private sector.