Security Alert Fatigue and AI

Advanced Security

Mirza Cutuk

May 25, 2021

Introduction

Security Monitoring

There is a significant number of suspicious events occurring within most enterprise networks and computer systems on a daily basis, that go completely undetected.[27]
Difference between identifying an event as suspicious or not can be the difference between successfully fending off a cyber attack and having the organization and its daily operations severely crippled.

The goal of monitoring the infrastructure is to automate and correlate as much as possible between both events and vulnerabilities and to build intelligence into security tools so that they can provide quality alerts if a known sequence of bad events has occurred or a known vulnerability is being targeted. In simple terms, this means that there has to be a security monitoring and log management strategy set in place, along with an incident response plan and security metrics to measure the response effectiveness.[27]

To monitor events on its vital systems, an organization will most frequently use a (network) intrusion detection system ((N)IDS) or a security information and event management (SIEM) in a combination with a log management tool in some cases, although most SIEM systems are advanced log management tools. Main actors in this story are security analysts and managers who sift through this huge amount of data in order to identify and give attention to the most interesting events. [27]

Intrusion Detection and Prevention System

Having in mind that information security concerns itself with confidentiality, integrity and availability of information systems and the information or data they contain and process, an intrusion would be any action taken by an adversary that has a negative impact on confidentiality, integrity, or availability of that information. There are, of course, different types of intrusions all coming with varying level of damage to the organization:

- Physical theft
- Abuse of privileges
- Unauthorized access by an outsider
- Malware infection

Intrusion detection and prevention is not a single product that an organization buys or implements, but is instead a set of intertwined technologies used alongside proper methodology and skill sets

to recognize and prevent some kind of an intrusion into the organization's systems. Some of these technologies are:

- Anti-Malware Software
- Network-based Intrusion Detection Systems
- Network-based Intrusion Prevention Systems
- Host-based Intrusion Prevention Systems
- Security Information Management Systems
- Network Session Analysis
- Digital Forensics
- System Integrity Validation

We will not go into details of each technology listed above, but it is important to note that each of these has its own strengths and weaknesses. To have an effective intrusion and detection program, it has to be designed to play to their strengths and minimize the weaknesses.[27]

Security Information and Event Management

As we have previously mentioned, computer and network systems produce vast amounts of log data, and a security analyst alone cannot possibly go through each and every event and at the same time analyse and successfully make connections between different events.

Log management systems were supposed to make the task of analysing log data a bit simpler by storing it in one place and indexing it for easier search. After some time, some manufacturers added correlation engines to their products and SIEM was created. A correlation engine analyses specific attributes of the log data (username for example) and compares it with other contextual information (location, time, whitelists, blacklists, known vulnerabilities, etc.). Of course, all of this needs to happen in near real-time, or the purpose of SIEM would be defeated. Correlation engine gives systems the ability to cross-compare current event with specific use cases. A good example of this is that it can alert when a user logs in to a critical system outside of work hours, from a different country, etc. As we can see, SIEM can be a quite useful tool. However, different systems have different log formats and different messages in them. In order for the SIEM to understand, analyse and then correlate the log records, it needs human help. Namely, someone to go through the logs manually and tell SIEM how to interpret each message. With this interpretation SIEM can then normalize (break apart and organize individual pieces of the message) and correlate the message components.[18]

Security Alert Fatigue

Some of the main parts of an organization's security system are IDS and SIEM systems. Both of these connect to the organization's vital systems, monitor them and produce alerts if there is a suspicious activity. However, both of them have some quite significant problems.

According to an industry survey, 51% of organizations say that their security solutions do not inform them or they are unsure if their solution can inform them about the root causes of an attack.[14] These difficulties, as expected, are not usually caused by a lack of data, as most hardware and software components provide well-detailed logs. Consequently, failure to detect and respond to security incidents is not caused by the lack of information that points to attacks, but

by the rapid growth of log data and the application of manual analysis which is simply not feasible anymore.

Another common issue is that these logs are typically weakly structured, use a variety of inconsistent formats and terminologies, and are spread across different files and systems. To detect sophisticated multi-vector attacks, it is necessary to identify related events and connect them to create a complete picture for the identification of malicious activity. Isolated indicators are often insufficient in their local context and it is therefore necessary to organize and connect log information. Furthermore, the interpretation of log information is highly context-specific, which makes it difficult for monitoring applications to identify relevant information without a deeper understanding of their context. Manual log analysis by human experts does not scale, and there exists a lack of automated mechanisms to bind together and interpret security information.[3] Consequently, all of these factors contribute to security alert fatigue; a situation in which security analysts struggle to cope with the enormous volume of raw log data, to extract insights from these heterogeneous sources, and to identify and respond to increasingly complex attacks.[3]

Artificial Intelligence for Security Alert Fatigue

Human factor plays a major role in analysing and recognizing dangerous system behavior when working with IDS or SIEM systems. The more complex the organizations' system is, there is more data and more people are needed. At one point, the complexity becomes so overwhelming that it is impossible to scale security with the human factor so deeply ingrained in the everyday workings of a security operations center (SOC). It remains a major issue for big organizations around the world, but there is a growing interest to remedy this issue. One of the solutions that showed good results is the so called file integrity monitoring (FIM) system.

FIM is a system in which the changes to files on a system, such as operating systems executables and libraries, trigger warnings. This approach, according to Mark Kedgley (CTO, New Net Technologies), is highly sensitive, and it is not prone to zero-day threats. Additionally, malware that manages to evade antivirus systems or IDS (signature based detection) will be recognized by FIM. However, there is a downside; being a highly sensitive system, regular changes like patching can easily set it off. That is why, Kedgley continues, it is important to automatically learn what good changes look like. Another novel idea related to FIM is that it would be possible to automatically compare the file changes to threat analysis repositories, and essentially crowdsource the knowledge about something being malicious or not. There are such efforts already being made in the industry with the Kaspersky whitelist and Google's Virus Total repository.[20]

Other manufacturers, such as Microsoft, Splunk, Kaspersky, remain in the field of signature based detection, with an additional layer of artificial intelligence. A very intriguing tool is Microsoft's Azure Sentinel, a cloud native SIEM solution that utilizes machine learning techniques to reduce security alert fatigue. It is possibly the most advanced SIEM solution at the moment and it is a very exciting one to say the least. According to the Azure Sentinel website, Microsoft's Fusion technology uses state of the art scalable learning algorithms to correlate millions of lower fidelity anomalous activities into tens of high fidelity cases. This tool has a reported median 90% decrease in alert fatigue due to Fusion's ability to detect complex, multi-stage attacks.[19]

We can see that there are already some approaches that utilize the artificial intelligence to reduce the noise in the SOC. Our approach, like Azure Sentinel's is the signature based one. In the continuation of this paper, we will examine how a neural network can be utilized to reduce security alert fatigue by training it on a data set of known attack categories.

Neural Networks

Neural networks are computer systems that were developed to simulate the human nervous system for machine learning tasks by treating the computational units in a learning model in a manner similar to human neurons. Based on the architecture, we can classify neural networks into two categories: single-layer neural networks (perceptron) and multi-layer neural networks. [2] While perceptrons do have their application and they represent a basis for more complex neural networks, we will focus on and use multi-layer neural networks.

Multi-Layer Neural Networks

Multi-layer neural networks, as their name suggests, contain more than one computational layer. Specific architecture of multi-layer neural networks we are going to use is called the feed-forward network, because successive layers feed into one another in the forward direction from input to output. Feed-forward network architecture default configuration assumes that all nodes from one layer are connected to those of the next one. Thus we can say that the structure of a neural network is almost fully defined by the number of layers and the number/type of nodes in each layer. The only remaining matter is the loss function that we are trying to optimize in the output layer. Depending on what tasks we are trying to perform with our neural network, it is common to use softmax outputs with cross-entropy loss calculation for discrete prediction and linear outputs with squared loss for real-valued prediction.[2]

Simple AI Model for Security Alert Fatigue

UNSW-NB15 data set

For this project, we have opted for the UNSW-NB15 data set created by Nour Moustafa and Jill Slay at the University of New South Wales at the Australian Defence Force Academy in 2015. We could not find many data sets intended for purpose of attack identification, which shows that this type of research is still relatively new and there is a lot to discover.[22]

UNSW-NB15 is a synthetically generated data set created in a lab environment. Testbed configuration contained two networks and the traffic was generated using the IXIA traffic generator. IXIA tool was configured with three virtual servers; servers 1 and 3 were configured for the normal traffic, while server 2 formed abnormal/malicious activities in the traffic. The servers are connected to hosts via two routers which are in turn connected to a firewall which lets through all the traffic (regardless if normal or abnormal). The tcpdump tool is installed on the router 1 to capture the Pcap files of the simulation uptime. The attack behaviour is obtained from the CVE site for the purpose of a real representation of a modern threat environment.[22]

The features of the UNSW-NB15 data set are extracted using Argus, Bro-IDS tools and twelve algorithms developed using c programming language. [22]

Data set itself has 49 features including labels. These features include a variety of packet and flow-based features. Features 1 - 35 represent the integrated gathered information from data packets (mostly header packets), while features 36 - 47 are additional flow based features. Features 36 - 40 are considered as general purpose features and 41 - 47 are labelled as connection features. Each feature in the general purpose features has its own purpose according to the defence point of view. Connection features are created in order to provide defence with information during attempt to connection scenarios, since the attackers might scan hosts in an unpredictable manner (once per minute or one scan per hour). In order to identify these types of attacks, features 36 - 47 will help to sort according to the last time feature to capture similar characteristics of the connection records. [22]

As for the families of attacks included in this data set, we have the following categories (as named in the original data set) along with their respective number of records in the parentheses: Fuzzers (24,246), Analysis (2,677), Backdoors (2,329), DoS (16,353), Exploits (44,525), Generic (215,481), Reconnaissance (13,987), Shellcode (1,511), Worms (174). Since the attack categories covered by this data set are quite interesting and can have different approaches, we have decided to introduce each of them in a bit more detail in the following section.

UNSW-NB15 Attack Categories

Fuzz Testing

Fuzz testing or fuzzing is a method of feeding random, semi-random or purposely crafted input to an application with the goal of finding bugs in it, and consequently, possible exploitable vulnerabilities. Fuzz testing concept was conceived at the University of Wisconsin by professor Barton Miller. While it is a seemingly simple and straight-forward method, it has evolved to become a very useful tool, and proved to be great help in bug hunting and preventing more serious attacks before the application is rolled out.[5][6]

Even though original fuzzing method had the three following characteristics:

1. the input is random,
2. simple reliability criteria (if an application crashes or hangs it is considered to have failed the test),
3. it can be automated to a high degree and results can be compared across applications, operating systems, and vendors.[6]

New developments in the technology have brought fuzzers that use genetic algorithms, static fuzzing vectors (with values that are known to be dangerous, negative and or very big values for numbers, random binary strings or interpretable characters/instructions for SQL requests, etc.) and even machine learning algorithms.[15]

Biggest strength of original fuzz testing method is the fact that it approaches the system as a black box, which means that the 'attack' is carried out without prior knowledge about the system itself. As such, fuzzing can be a very serious tool or a weapon in terms of cyber security. There are also white- and grey- box variants of the method.

A very simplistic example of fuzzing is as follows. Let us consider a website form that has a field which takes as input user's choice between three options. If the form stores the answer index, possible options are most likely 0, 1 and 2. Fuzzing will examine what happens when we input 3 or 255, a float or an unsigned/signed integer, or something similar that passes the 'first line of defense', but might cause malfunctions when it is already inside the application.[15]

Reconnaissance

Reconnaissance, in simple terms, is finding the weak spots in the target system, so that the attacker can devise a plan of attack. On its own, reconnaissance does not propose a direct threat to the system. However, it can provide a great deal of information about the system, its architecture and people using it. [9] There are two types of reconnaissance:

- passive reconnaissance - gathering information about the target with minimal or without any interaction. This includes checking the website information, searching the web, even checking job postings to find out as much as possible without leaving a footprint. Consequently, passive reconnaissance is also commonly referred to as footprinting.[21][13][12]

- active reconnaissance - gathering information about the target through different methods that require interaction with the target. Some of the commonly used tools include nmap, traceroute, netcat, ping, etc. Active reconnaissance is also known as scanning, since the attacker is probing the system to see which parts of the application could be exposed. Since reconnaissance requires interaction with the system, it leaves a trace in the system logs. For this reason, scanning is quite often done slowly, so that the trace would not be recognized.[21][13][12]

Shellcode

In a characteristic code injection attack, the attacker transmits some kind of malicious input that will exploit a memory corruption vulnerability in a program running on the target's computer. This injected code is known as shellcode and it carries out the initial attack stage. This usually involves download and execution of a malware binary executable on the compromised host, or opening a command shell to the attacker, thus enabling them to take control of the target system. It is typically written in low level languages such as assembly or C, since the attacker is targeting a memory vulnerability. There are several categories of shellcode:

- local - used usually when the attacker has limited access to a machine, but can exploit a vulnerability such as buffer overflow of a higher privileged process, giving them the same privileges as the targeted process.[26]
- remote - used when the attacker is targeting a process on a different machine on a local network, intranet, or a remote network. If successful, the attack will provide the attacker with the access to the target across the network (remotely).[26]
- download and execute - shellcode that is used as the initial stage of an attack, namely to start the download and execution of some type of malware on the target machine.[11]
- staged - similar to the the download and execute in its nature, staged shellcode is used when the amount of data that can be injected into the target system is limited to execute useful shellcode. In this situation, the attacker will execute the attack in stages. First, a small piece of code will be injected, and this code will download and execute a bigger piece of shellcode which will then be able to perform something useful for the attacker.[24]
- egg-hunt - when the attacker can inject a big piece of shellcode into the process, but does not know where in the process it will end up, they will inject the big piece into the process first. Second, they will inject a smaller piece of shellcode to a predictable location. Smaller piece will then find and execute the bigger piece of shellcode.[8]
- omelette - used when the attacker can inject several pieces of smaller shellcode. The final shellcode piece will then find and combine the already injected smaller pieces into a larger useful shellcode block. [8][23]

Analysis

This category (based on the UNSW-NB15 data set) includes different attacks, namely:

- port scanning Port scanning is a method of determining which ports on a network are open and could be receiving or sending data. It is also a process for sending packets to specific ports on a host and analyzing responses to identify vulnerabilities. This scanning can't take place without first identifying a list of active hosts and mapping those hosts to their IP addresses. This activity, called host discovery, starts by doing a network scan. The goal behind port and network scanning is to identify the organization of IP addresses, hosts, and

ports to properly determine open or vulnerable server locations and diagnose security levels. Both network and port scanning can reveal the presence of security measures in place such as a firewall between the server and the user's device. After a thorough network scan is complete and a list of active hosts is compiled, port scanning can take place to identify open ports on a network that may enable unauthorized access. [29]

- spam Most frequent type of spam is email spam, which can be used for purposes such as advertising, but also for fraudulent activities and phishing. [30]
- html file penetration HTML file penetration is a vulnerability occurring in web applications that allows users to insert(inject) html code through a specific parameter or an entry point. This type of attack is usually used in combination with social engineering to trick valid users of the application to open malicious websites or to insert their credentials to a falsified login form that redirects the users to a page that captures their credentials. [17]

Backdoor

Backdoor refers to any method by which either authorized or unauthorized users are able to bypass normal security measures and gain high level user access (root access) on a computer system, network, or software application. Once they're in, cyber criminals can use a backdoor to steal personal and financial data, install additional malware, and hijack devices.

Even though cyber criminals can utilize backdoors to steal personal data or install malware, backdoors are not only used by attackers. Backdoors can also be purposely built in by software or hardware developers as means of gaining access to their technology after the distribution. These kinds of backdoors can be useful for example for helping customers who are locked out of their devices or for troubleshooting and resolving software issues.

Unlike other cyber threats that make themselves transparent to the user (ransomware), backdoors are known for being a discrete method of intrusion. [4]

DoS

Network resources (web servers for example) have a finite capacity of the number of requests that they can process at one time. Additionally, the channel that connects the server to the Internet also has a finite bandwidth/capacity. Whenever the number of requests exceeds the capacity limits of any component of the infrastructure, the service suffers in the following ways:

- Response to requests will be significantly slower than normal
- Some or all users' requests may be totally ignored

Distributed Network Attacks or Distributed Denial of Service (DDoS) attacks are the type of attack that takes advantage of the finite capacity limits that apply to any network resources. DDoS attack will send a big number of requests to the attacked web resource with the aim of exceeding the website's capacity to handle multiple requests, halting the website's normal functioning. [28]

Exploits

With the technology constantly advancing and (quite often distributed) development teams rushing to catch up with the market and user demands, it is bound that there will be some exploitable vulnerabilities in the applications. If an attacker knows about a vulnerability of an application or an operating system, they can use it to their advantage, if the vulnerability has not been patched or it is completely unknown (zero day exploit) to the manufacturers and users of the application or system.

Generic

A generic block cipher attack is an attack that works (more or less) on all block-ciphers (with a given block and key size), without consideration about the structure and the inner workings of the block-cipher. Some examples are brute force attack, but also the so-called birthday attack. The birthday attack is based on the mathematical problem whose formulation states that if given a large enough group of people, there is a high probability that two or more people will have the same birthday. For every 23 people there is about a 50% chance of two people having the same birthday. This eventually becomes 100% when large enough.

The same logic applies to block ciphers. Block ciphers have predefined block size and the more they are used for encryption, the greater the probability of running into some kind of collision, and collisions, as we know, make the algorithm vulnerable. The birthday attack cannot be stopped due to the fact that it has a very strong mathematical foundation and attackers can therefore take advantage of it. However, one mitigation method is increasing the block sizes to reduce collision probability. [25]

Worms

A Computer virus is a type of malware that works in such a way that it inserts a copy of itself into another programming, effectively becoming a part of that program. It can spread from one computer to another and can range in severity from almost benevolent to extremely damaging to data and sometimes even causing denial-of-service conditions. Almost all virus infections start with the user executing the malicious host file. When the host file is executed, so is the virus code. Viruses spread when the software or documents (hosts) they are attached to are transferred from one computer to another via the network, a disk, file sharing, or infected email attachments. [16]

Computer worms are similar to viruses in terms of replicating functional copies of themselves and causing a similar type of damage. One difference between these two is that worms are standalone software applications and do not require a host to spread. Worms either exploit a vulnerability on the target system or use social engineering to get the users into executing them. A worm enters the target system through a vulnerability and utilizes file or information transport features on the system, allowing it to travel unaided. [16]

Model

For this project we recognized two different possible classification model approaches:

- Binary Classification Model - this model is trained on the UNSW-NB15 data set and it simply recognizes whether network behavior is normal or abnormal. This kind of model would be a stepping stone in remedying the security alert fatigue issue by eliminating false positives and providing the security analyst with genuine alerts that they could further analyse to find out what the exact threat is. This is the model we have opted for in this project.

- Multi-Class Classification Model - this is a more advanced model, as it, after being trained on the UNSW-NB15 data set, recognizes that network behavior is abnormal, but also classifies it and provides the analyst with the information about which of the nine families of attacks that specific attack comes from.

Implementation Details

For both models, we used neural networks implemented using the TensorFlow platform and Keras API.

TensorFlow 2 is an end-to-end, open-source machine learning platform. It is simply an infrastructure layer for different programming scenarios. It combines four key abilities:

- Efficient low-level tensor operation execution on CPU, GPU, or TPU
- Gradient of arbitrary differentiable expressions Computation
- Scaling computation to many devices
- Exporting programs (graphs) to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is a deep learning API written in Python, utilizing the machine learning platform TensorFlow 2.0 (or TensorFlow 2). It provides the essential abstractions and building blocks to develop and ship machine learning solutions quickly. [1]

Binary Classification Model

Data Pre-processing

Our training data set contains 2000 records of aforementioned nine attack categories in addition to normal network behavior. Since each data row was labeled with the attack category, our first step was to generate new labels, e.g to mark behavior as normal or abnormal. This allowed us to perform binary classification (since we ended up with two labels). Since we generated new labels ourselves, we did not have to use one-hot encoding for each data row.

Neural Network Structure

After pre-processing the data, we moved forward with the creation of our neural network.

Neural network hyperparameters are chosen based mostly on trial and error, so we took this experimental approach to building our model. We explored all parameters apart from the number of epochs, which was left constant due to long training times and good initial results at value of 100 epochs. After testing a different numbers of hidden layers, we concluded that the best approach for our problem is to have one hidden layer between input and output layers. Our input layer, of course, has 39 nodes, since we extracted 39 features with numerical values, and our output layer has one node, since we only give one output value, more precisely, normal or abnormal behavior. Our hidden layer uses the ReLU (Rectified Linear Unit) activation function, which is commonly used in machine learning today, due to its ability to improve learning with neural networks. Output layer uses the classic sigmoid function, as this is the standard function used for the output layer when working on binary classification.

When it comes to number of nodes in the hidden layer, we again used trial and error approach, and after several trials, we have found that the optimal number of nodes in the hidden layer is 10. We have tried smaller and greater values, but it seems that 10 is the balance between making the neural network extract information better without losing too much of the information, with the accuracy value of 92.21%. Results from our several runs can be seen in the table below.

100 epochs	Batch size = 5		Accuracy	
First layer	Hidden layer	Output layer	Mean	Variance
39	39	1	88.24%	10.30%
39	20	1	90.85%	3.84%
39	10	1	92.21%	2.03%
39	5	1	89.59%	3.20%

Next hyperparameter to choose was the batch size. We tested multiple values and observed that in the current configuration, best value for batch size was 5, with a reported accuracy of 92.21%, as seen in the results table below.

100 epochs	Accuracy	
Batch size	Mean	Variance
4	83.27%	16.04%
5	92.21%	2.03%
8	90.58%	4.52%
16	90.58%	4.14%
20	88.72%	8.57%
32	86.60%	9.67%

We were looking for ways to improve the accuracy of the training, and after some research realized that it would be good to prepare the data before feeding it to the neural network, since neural networks 'like' to have consistent input values (both in scale and distribution). One effective data preparation scheme is standardization.Data is simply rescaled so that the mean value for each attribute is 0 and the standard deviation is 1. This preserves Gaussian and Gaussian-like distributions whilst normalizing the central tendencies for each attribute. After preparing the data we tweaked the parameters further and we concluded that the optimal configuration with data preparation is a three-layer neural network (39-10-1), with a batch size of 8 and accuracy of 95.85%, as seen in the table below.

100 epochs	Accuracy	
Batch size	Mean	Variance
4	95.70%	1.51%
8	95.86%	1.01%
16	95.26%	1.37%
32	94.61%	2.26%

Our accuracy measure was obtained by using the logarithmic loss function (binary cross entropy) during training, since this is the preferred loss function for binary classification problems. The model also uses the efficient Adam optimization algorithm for gradient descent and collects accuracy metrics to evaluate the model.

Training

In order to train the network, we used the stratified k-fold cross validation. This is a resampling technique that provides an estimate of the performance of the model. It accomplishes this by splitting the data into k-parts, training the model on all parts except for one which is used as a test set to evaluate the performance of the model. This process is repeated k-times and the average score across all constructed models is used as a robust estimate of performance. It is stratified, meaning that it will look at the output values and attempt to balance the number of instances that belong to each class in the k-splits of the data.[7]

Testing and Results

UNSW-NB15 data set also includes a testing data set, so we used 1836 records from it to evaluate our model. We fitted the model with the train data and tried to predict whether the behavior consisted in data vector is abnormal or not. With our best configuration that trained with accuracy of 95.86%, we obtained real accuracy (on the test data set) of 93.57% and a false positive rate of

6%. Compared to the reported 40% false alarms received by the security operations centers, this is a big improvement.[10]

Multi-Class Classification Model

We have seen that the binary classification models behaves very well, and gives quite an improvement compared to the standard security alerting systems based solely on correlation engines and human skill. An additional improvement to the model we developed could be a multi-class classification model. This model would again have to be trained on UNSW-NB15 data set, but it would not give us information about whether the behavior is abnormal or not. Instead, this model would use threat features to recognize one of the nine attack families and alert the security analyst with more information. This project is certainly more challenging and requires more time and effort, but the payout could be significant as it would be another stepping stone towards better security in larger organizations.

Conclusion

Promising results from the binary classification model are a true inspiration for the continuation of this work. They show that signature based detection is not going anywhere soon, and that we could make significant improvements to our existing security alert systems by using AI. We know that there are organizations that already took this challenge, but the scarcity of source material shows that this field has not yet picked up the large-scale interest necessary for fast improvements and discoveries. Security alert fatigue will continue being a big problem for security operation center everywhere, and we hope that AI based detection along with human skillset can be a factor contributing to a safer digital environment.

Bibliography

[1] *About Keras*. URL: https://keras.io/about/. (accessed: 21.05.2021).

[2] Charu C. Aggarwal. *Neural Networks and Deep Learning, A Textbook*. Springer, 2018. ISBN: 978-3-319-94463-0.

[3] Kabul Kurniawan Andreas Ekelhart Elmar Kiesling. "Taming the logs – Vocabularies for semantic security analysis". In: *SEMANTiCS 2018 – 14th International Conference on Semantic Systems* (2018). DOI: https://www.sciencedirect.com/science/article/pii/S1877050918316156.

[4] *Backdoor Computing Attacks - Definition Examples—Malwarebytes*. URL: https://www.malwarebytes.com/backdoor/. (accessed: 19.05.2021).

[5] P. Miller Barton. "CS 736 Project List". In: (1988). DOI: http://pages.cs.wisc.edu/~bart/fuzz/.

[6] P. Miller Barton. *Fuzz Testing of Application Reliability*. URL: http://pages.cs.wisc.edu/~bart/fuzz/. (accessed: 13.05.2021).

[7] *Binary Classification Tutorial with the Keras Deep Learning Library*. URL: https://machinelearningmastery.com/binary-classification-tutorial-with-the-keras-deep-learning-library/. (accessed: 25.05.2021).

[8] Avast Bradshaw Stephen. *Omlette Egghunter Shellcode*. URL: http://thegreycorner.com/2013/10/31/omlette-egghunter-shellcode.html. (accessed: 17.05.2021).

[9] James Burton. *Cisco Security Professional's Guide to Secure Intrusion Detection Systems*. Syngres Publishing, 2003. ISBN: 1932266690.

[10] *Cybersecurity 101: What You Need To Know About False Positives and False Negatives*. URL: https://www.infocyte.com/blog/2019/02/16/cybersecurity-101-what-you-need-to-know-about-false-positives-and-false-negatives/. (accessed: 25.05.2021).

[11] *Download and LoadLibrary shellcode released*. URL: https://web.archive.org/web/20100123014637/http://skypher.com/index.php/2010/01/11/download-and-loadlibrary-shellcode-released/. (accessed: 18.05.2021).

[12] *Ethical Hacking - Fingerprinting*. URL: https://www.tutorialspoint.com/ethical_hacking/ethical_hacking_fingerprinting.htm. (accessed: 25.05.2021).

[13] *Ethical Hacking - Footprinting*. URL: https://www.tutorialspoint.com/ethical_hacking/ethical_hacking_footprinting.htm. (accessed: 25.05.2021).

[14] *Exposing the Cybersecurity Cracks: A Global Perspective*. 2014.

[15] *Fuzzing—OWASP*. URL: https://owasp.org/www-community/Fuzzing. (accessed: 14.05.2021).

[16] Bruce Heldman. *Learn About Viruses, Bots, Malware and Trojans*. 2015. URL: https://help.queens.edu/hc/en-us/articles/202958394-Learn-About-Viruses-Bots-Malware-and-Trojans. (accessed: 20.05.2021).

[17] *HTML Injection - Penetration Testing*. URL: https://pentestlab.blog/tag/html-injection/. (accessed: 19.05.2021).

[18] Jon Inns. "The evolution and application of SIEM systems". In: *Network Security* 2014.5 (2014), pp. 16–17. DOI: https://www.sciencedirect.com/journal/network-security/vol/2014/issue/5.

[19] Ram Shankar Siva Kumar. "Reducing security alert fatigue using machine learning in Azure Sentinel". In: (2019). DOI: https://azure.microsoft.com/en-us/blog/reducing-security-alert-fatigue-using-machine-learning-in-azure-sentinel/.

[20] Steve Mansfield-Devine. "Getting smarter about alerts". In: *Computer Fraud Security* (2015). DOI: https://www.sciencedirect.com/science/article/pii/S1361372315300956.

[21] Netwrix. *"Netwrix CompTIA Security+ Exam Study Guide"*.

[22] Jill Slay Nour Moustafa. "UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems". In: *Military Communications and Information Systems Conference (MilCIS)* (2015). DOI: https://research.unsw.edu.au/projects/unsw-nb15-data-set.

[23] Michalis Polychronakis. "Comprehensive Shellcode Detection using Runtime Heuristics". In: (2010). DOI: https://dl.acm.org/doi/10.1145/1920261.1920305.

[24] Beyond Binary Reeves Oliver. *Staged vs Stageless Handlers*. URL: https://buffered.io/posts/staged-vs-stageless-handlers. (accessed: 17.05.2021).

[25] Palash Sarkar. "Generic Attacks on Symmetric Ciphers". In: *The 7th International Conference on Cryptology in India, Kolkata* (2006). DOI: https://www.isical.ac.in.

[26] Logsign Team. *How to Prevent Shellcode Injection*. URL: https://www.logsign.com/blog/how-to-prevent-shellcode-injection/. (accessed: 18.05.2021).

[27] John R. Vacca. *Managing Information Security, 2nd Edition*. San Val, 2013. ISBN: 9781417642595. URL: http://books.google.com/books?id=W-xMPgAACAAJ.

[28] *What is a DDoS Attack? - DDoS Meaning*. URL: https://www.kaspersky.com/resource-center/threats/ddos-attacks. (accessed: 20.05.2021).

[29] *What is port scanning?* URL: https://www.avast.com/business/resources/what-is-port-scanning. (accessed: 19.05.2021).

[30] *What is Spam and a Phishing Scam - Definition.* URL: https://www.kaspersky.com/resource-center/threats/spam-phishing. (accessed: 19.05.2021).