Towards a Data-Driven Recommender System for Handling Ransomware and Similar Incidents

Martin Husák
Institute of Computer Science, Masaryk University, Brno, Czech Republic
husakm@ics.muni.cz

***Abstract*—Effective triage is of utmost importance for cybersecurity incident response, namely in handling ransomware or similar incidents in which the attacker may use self-propagating worms, infected files, or email attachments to spread malware. If a device is infected, it is vital to know which other devices can be infected too or are immediately threatened. The number and heterogeneity of devices in today's network complicate situational awareness of incident handlers, and, thus, we propose a recommender system that uses network monitoring data to prioritize devices in the network based on their similarity and proximity to an already infected device. The system enumerates devices in close proximity in terms of physical and logical network topology and sorts them by their similarity given by the similarity of their behavioral profile, fingerprint, or common history. The incident handlers can use the recommendation to promptly prevent malware from spreading or trace the attacker's lateral movement.**

I. Introduction

The rising complexity and variety of cyberattacks complicate incident response. Traditional approaches to network security based on securing and deploying intrusion detection systems on the network's perimeter are insufficient and can be bypassed. Cybersecurity incident response teams (CSIRT/CERT) and security operation centers (SOC) are often flooded with alerts and have low awareness of particular hosts and services in the network, namely in large networks and organizations. Effective triage and prioritization of threats and incidents are of utmost importance, which inspires the research on alert aggregation and correlation, automation of incident response routines, human cognition augmentation, and decision support.

The major motivation for our work is the need to handle ransomware attacks and similar incidents. Ransomware attacks often use techniques of social engineering, such as phishing emails, to deliver malware and get a foothold in the network. The malware further spreads in the network via exploitation of surrounding computers or sharing infected files and devices among users. Traditional perimeter-based network defense and network-based intrusion detection systems are bypassed, and there is little chance of mitigating the spread of infection. Nevertheless, the behavior of malware can be anticipated to some extent. A typical malware uses a few exploits, focuses on certain types of services, software, or devices, and spreads in close proximity first. Thus, a lateral movement of an attacker can be observed, traced, and even projected. However, that requires detailed knowledge of the local environment and collaboration with users and administrators, if not direct access to the network, which is complicated in large networks, infrastructures, and organizations. The incident handlers would appreciate any piece of information that would guide them through the network and pinpoint nodes that might be involved in an ongoing attack or are immediately threatened. The key question of an incident handler is: if this device is infected, which others devices can be infected or threatened?

In this work, we propose a design of a recommender system for the incident response that uses common network monitoring and security infrastructure to combat threats like ransomware or lateral movement of an attacker. The system uses network traffic analysis, vulnerability assessment, and knowledge of the local environment to profile devices in the network and support incident handlers' decision-making. The profile consists of a devices' location in physical and logical network topology, software and service fingerprints, vulnerabilities, and other contextual data. In case of infection of a device in the network, the system recommends a list of similar devices that are at major risk. The similarity is considered in terms of devices' proximity and similarity of software equipment and vulnerabilities.

This paper is divided into six sections. Section II briefly describes background and related work. In Section III, we propose the design of the recommender system. In Section IV, we describe the calculation behind the recommendations. Section V illustrate the use of the system on an example. Section VI concludes the paper and outlines future work.

II. Background and Related Work

Incident handling is a fundamental activity provided by a cybersecurity team (CSIRT/CERT). For example, the CSIRT Services Framework enumerates services related to incident management that CSIRT teams can provide [1]. The services are divided into five areas – information security incident management, information security event management, vulnerability management, situation awareness, and knowledge transfer. The complete list of the services is listed in RFC 2350 [2]. Recently, the incident response became a part of third-generation SOC (Security Operations Center) as defined by CISCO [3]. The procedures of incident handling are well structured and formalized in the literature and adopted by practitioners. However, each environment is different, and each team uses a different set of tools to gather data, making it difficult to specify local issues and technical details. The Computer Security Incident Handling Guide [4] by NIST divides incident handling into preparation, detection, analysis,

containment, eradication and recovery, and post-incident activities. The Incident Handler's Handbook [5] by the SANS Institute includes highly relevant entries in the incident handler's checklist. For example, the incident handlers are reminded to identify the source and location of the incident and its scale. The incident handler has to be aware of the impacted hosts and their neighborhoods to properly plan the course of action. Our proposed system directly addresses these needs.

Incident handlers are forced to make many prompt decisions, especially during incident triage (prioritization) and their resolution. However, according to Spring and Illari, who reviewed human decision-making in incident response [6], existing guidelines do not give advice on how to obtain an overall picture of incident with incomplete data, e.g., how to generalize hypothesis in terms of time (among distinct events) and space (among devices in the network or possible cyber victims). These steps are usually accomplished via the expert knowledge of a security team. In other related work on decision support in cybersecurity, Webb et al. [7] proposed a model for security risk management that addresses the deficiencies in practice and poor decision making. Happa et al. [8] studied the impact of using visualization and decision support tools in SOC, including the propagation of risk sensor alerts to business processes. The study states recommendations for future development, such as including contextual data and enabling collaboration.

The research on decision support tools and recommender system in cybersecurity usually focus on finding an optimal countermeasure for ongoing or possible cyber attack out of available response actions [9], optimal purchase of defense tools [10], or their optimal placement [11]. Polatidis et al. [12] were inspired by a product recommender system in designed a system for attack prediction. Recommender systems are also considered in vulnerability assessment and scoring [13], [14]. Identification of systems at risk via analysis of discussion on the darkweb can be found in work by Almukaynizi, et al. [15]. Nevertheless, network defense and incident handling are lacking recommender systems or similar tools. Exceptions can be found in work by Albanese et al. [16], who proposed automated tools to improve analyst's performance.

Although there is a lack of decision support tools and recommender systems addressing particular tasks, there are tools that enable achieving situational awareness and, thus, make timely and informed decisions. Situational awareness was extensively studied in military and aviation and gained recognition in cybersecurity as so-called cyber situational awareness (CSA) [17]. Following the widely-used definition, CSA consists of three levels: the perception of the elements in the cyber environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future [18]. Each level builds upon the previous one; one cannot fully understand the situation without proper perception nor project the situation without understanding it. The recent status of research in this area was summarized by Husák et al. [19] or Gutzwiller et al. [20]. The fundamental results were achieved, and the attention has moved towards their application, deployment, and field studies. Achieving CSA requires a vast range of heterogeneous information that can be provided by a plethora of existing tools [21]. However, it is difficult to process such data, namely because their volume, velocity, and variety effectively make them big data [19]. Another important issue is providing the user with the right data at the right time and present them in a meaningful manner to avoid information overload and maintain sufficient performance [19]. Complex tools to enable CSA were proposed in the past, Cauldron [22] and CyGraph [23] being prime examples. However, such complex tools are often too complicated to be used effectively, namely due to the high number of required inputs that could not always be provided in practice. More lightweight solutions are called for, albeit those would come with a lower level of detail or precision [24].

III. Design of a Recommender System

To address the challenges stated in the introduction of this paper, we propose a recommender system that would help incident handlers in combating ransomware and similar attacks. Herein, we first summarize the requirements for such a system. Subsequently, we propose an architecture of a prototype.

A. Requirements and Architecture

The major requirement on the recommender system is that it should receive an identifier of a host in the network as input and put a sorted list of similar hosts on the output. The list should contain hosts similar to the one on the input sorted by the risk of their exploitation by an attacker or malware controlling the host on the input. If possible, the outputs shall be accompanied by the key pieces of information or data elements on the recommended hosts, namely those causing the similarity.

The proposed system is heavily data-centric, and, thus, there is a need to gather and store the data as follows:

- The system should collect or be able to access the data on the network and hosts in it. The required data items include a list of network segments, the location of a host in logical and physical network topology, hosts' services, software, and vulnerabilities, and contacts on primary administrators.
- The data shall hold as much information as possible under given circumstances. It is often unfeasible, laborious, or too expensive to have all the data available, and, thus, they do not need to be complete nor exact; the system shall work even with incomplete or slightly misleading data. Nevertheless, the best effort to data collection is expected.
- The data shall be stored in a way that allows for the interconnection of heterogeneous data from different domains. The data should be accessible at any time and continuously updated. The exact update interval depends on particular data; the data collected on the same or previous day should be sufficient.

We designed a prototype of a recommender system based on our previous work on cyber situational awareness and enabling technologies [24]–[27]. The system consists of three parts; data collection system, database, and recommender system. Our previous work covered the aspects of data collection and storage and is only briefly commented on in the following subsections. The recommendation subsystem is described here from the design perspective and integration with the rest of the system. The details on the recommendation algorithm are presented in the following section.

B. Data Collection

Our proposed system uses the data collection toolset proposed in our previous work [25]. The toolset continuously collects the data on the network so that they are instantly accessible when needed. The toolset mostly uses active and passive network monitoring, namely NetFlow technology [28] and Nmap tool [29], to map the network, enumerate active devices and services, and fingerprint operating systems and software running on the hosts. The collection of such data is fully automated and provides a sufficient amount of information and level of details for incident triage. Nevertheless, there is also the need to process data, such as network segmentation, logical topology, and contacts on primary users and administrators that can typically not be extracted automatically and have to be provided manually or connected to a local system.

The data to collect can be categorized into network-related and host-related. For the whole network, the system collects the following:

- List of the active hosts in the network (via NetFlow or Nmap).
- Network topology obtained via Nmap's *traceroute* feature from several observation points.
- List of network segments, preferably as a list of IP address ranges. Each segment shall include a brief description defining its location (e.g., physical location, department of the organization, hosting of virtual machines), purpose (e.g., a segment of workstations or servers, IP pool of VPN), and a responsible contact person (e.g., local IT administrator).
- History of security incidents, each with a list of devices involved in the incident (typically available via RTIR[1] or similar system used for incident management).

For each host in the network, the following pieces of information are collected:

- Fingerprint of the operating system (via NetFlow [26] or Nmap).
- List of open ports and network services, including the name and version of the underlying software (via NetFlow and NBAR2 signatures [30] or Nmap).
- If the host is a web server, then also the name and version of Content Management System (via WhatWeb tool[2]).
- If the host provides services encrypted via TLS (e.g., HTTPS), then also the validity of the certificate.
- Name and version of a web browser used on the system (via NetFlow-based analysis of HTTP User-Agent or TLS fingerpring [31]).
- Name of the antivirus software running on the system and timestamp of its latest update (via NetFlow traffic analysis and detection of the communication with a vendor's update server).
- List of vulnerabilities (either obtained via dedicated vulnerability scanner or estimated from fingerprints [27]).
- If the situation allows (e.g., the network is not too large or the organization conducts asset management), then the location, purpose, and contacts on administrators and users can be provided on the host level.

Readers interested in design consideration and technical details are kindly referred to our previous work on the data collection system [25]. It is worth noting that passive network monitoring in a typical setting updates the data every five minutes, including the list of active hosts and their fingerprints [26]. Active scanning takes up to 16 hours in experimental deployment in a /16 network. Passive network monitoring has a broader coverage and provides less detailed and exact information on almost all the devices in the network, while active scanning provides more detailed data, although on approximately four or five times fewer devices [25]. Dedicated vulnerability scanners might be too expensive when used in a large environment and, thus, can be to some extent replaced by mapping fingerprints to vulnerability descriptions via the CPE[3] strings [27].

C. Data Representation and Storage

The collected data are stored in a Neo4j, a technologically mature implementation of a graph database. Graph databases are becoming popular in cybersecurity data processing as they allow for intuitive and extensible data modeling and straightforward and comprehensible visualization of the data [32]. In this work, we use the CRUSOE data model [24] that aims at structuring heterogeneous data on the network for cyber situational awareness. The model is inspired by CyGraph [23], a graph-based data model and tool by MITRE. However, contrary to CyGraph, CRUSOE is more lightweight and favors simplicity and usability over completeness and exactness that are difficult to achieve in an operational setting. A sample database structured using the CRUSOE data model and a set of illustrations can be found at a GitHub repository [4].

D. Recommendation Subsystem

The recommendation subsystem is designed as a service that receives an identifier of a device in the network on the input and returns a sorted list of similar devices in close proximity as the output. Typically identifier on the input is an IP address or a domain name. The subsystem connects to the database

[1]https://bestpractical.com/rtir

[2]https://github.com/urbanadventurer/WhatWeb

[3]https://cpe.mitre.org/

[4]https://github.com/CSIRT-MU/CRUSOE-Data-Model

and looks up the host to assess its profile. Subsequently, it looks up devices in proximity of that host and retrieves their profiles. A graph traversal in network topology representation is leveraged, and a threshold of maximal distance is set in order to include only devices in close proximity, not all devices in the network, which might slow down the calculation and can be counterproductive in large networks. The list of the hosts is prioritized by their similarity and proximity to the host on the input, each host is identified by IP address or domain name, and its score is provided. A detailed explanation of the calculation is provided in the following section.

IV. Calculations behind Recommendations

The recommendations are based on the proximity and similarity of the hosts in the network to the host on the input. The recommendation prioritizes similar hosts in close proximity. The hosts in close proximity to an exploited host are at risk. The proximity is understood in several aspects; two hosts can be close to each other in physical and logical network topology, e.g., in the same room or in the same IP range. Alternatively, the two machines can be close to each other if they are controlled by the same users or administrators. The similarity is based on the similarity in software equipment, role, profile, or shared history of the two hosts. Nevertheless, the similarity in software equipment is a prevalent feature due to the fact that the attackers typically exploit certain services or pieces of software.

Formally, we sort the hosts by their risk score. The risk score (R) is calculated as a quotient of the similarity score (S) and distance (D) of the two hosts. Similarity score and distance are further calculated from a number of features that are explained in the following subsections. The formula is:

$$R = \frac{S}{D} = \frac{s_1 * s_2 * ...s_n}{min\{d_1, d_2, ..., d_n\}} \quad (1)$$

In practice, it would be advantageous to assign weights to similarity score, distance, or their elements. Such weights could be extracted from real-world scenarios and further tuned in operations. However, that is a task for future work.

A. Distance Calculation

As mentioned in Eq. 1, the distance between the two hosts is calculated as the minimal value of distance features. Various distance features allow for calculating the distance in different attack scenarios using different attack vectors. For example, worms and other malware spreading over the network will typically spread in the local network or subnet or to similar IP addresses. On the contrary, if the malware spreads via infected files in email attachments or flash drives, then it will typically spread first to machines used by the same users or in the same office or organization's department.

In order to avoid iterating all the hosts in the network, a breadth-first graph traversal can be used to find hosts with minimal distance in any of the distance metrics. The distance in logical network topology is calculated as the length of the path in the graph between two distinct *IP* nodes through the *Subnet* and *Organization Unit* nodes. For example, two IP addresses in the same subnet have a distance of 2; two nodes in the same organization unit but different subnets have a distance of 4. A similar distance measure can be calculated using the *Subnet* and *Contact*, which represents the similarity based on the fact that the same administrator oversees the subnets in which the IP addresses are located. Other similar metrics can be defined if additional data are available.

Our proposal is limited to the data available via our implementation of the data collection system [25]. Nevertheless, other distance features can be defined with additional data available. For example, the physical distance of the two hosts, location in the same room or department, or similarity of their IP addresses. The similarity of the IP addresses can be calculated with their 32-bit integer representation as $d = min\{IP_1, IP_2\}/max\{IP_1, IP_2\}$. As mentioned earlier, weights can be assigned to particular distance features, namely to allow comparing discrete values based on distance in graph to continuous values like the similarity of IP addresses or physical distance.

B. Similarity Calculation

The similarity is based on partial similarities in selected features of the hosts in the network and is calculated as a product of partial similarities $s_1 * s_2 * ...s_n$ as displayed in Eq. 1. Each partial similarity is a value in the range $< 0, 1 >$.

The similarity of the two hosts in their software equipment and open network-facing services are the prevalent features. The malware often uses exploits of specific software or services. For example, if malware uses SSH brute-force password attacks, then Linux machines with SSH servers are at risk. Often we do not know the exact software equipment or user preferences, and, thus, we may only assume similarities. For example, if the malware exploits a vulnerability of the Outlook email client, we shall look up Windows machines, even though we do not know if they have Outlook installed or if it is used.

CPE strings are a solid choice for a representation of a piece of software running on a host. A CPE string contains the software's vendor, name, version, and other details in a structured form. In practice, it is important that CPE strings can be constructed by OS and service fingerprinting software such as Nmap, which makes their retrieval fully automated [25]. Although common metrics (Levenshtein distance, Hamming distance) can be applied, it is more advantageous to design a CPE-specific similarity measure that would consider the similarity between particular elements of CPE and frequent use of wildcards (*) in CPE. Another issue is that there are several CPE strings for each host, such as one for the host's OS and one for each of its services. There is a need to match OS CPEs to each other and, then, increase the similarity for each similar service CPE on the same network port and the number of open services.

The calculation of similarity between the two CPE strings could be approached as follows. The CPE string is considered as an array of strings, i.e., each data element (vendor, product, version, etc.) is considered separately. Each element has a

value that is decreasing from left to right. Let's assign 0.5 to the vendor, 0.25 to the product, 0.125 to the version, and so on to other elements; the sum of all the values is 1. The elements are processed from left to right, i.e., from the most significant to the least significant, starting with the vendor. The initial value of similarity is 0. If the two elements are the same, the element's value is added to the similarity value. If a wildcard (*) or *ANY* value are encountered in one or both elements, then they are considered the same. If the values are different, the calculation stops, and the similarity value is returned.

Our proposed approach to calculating the similarity of two hosts by their CPE strings can produce one or more features (partial similarities). It might be advantageous to split CPE strings into groups and calculate the similarity of the OS, network services, client software, and other groups separately. In the case of comparing OS fingerprints and their CPE representation, only two strings are compared; both strings shall be available, namely if passive OS fingerprinting is used due to its coverage [26]. The same approach can be used in a situation in which there is only one CPE string available for each host, or only one can be selected (e.g., the most recent or the most frequent one); such comparisons can be performed with web browsers, email clients, antivirus software, and similar categories, in which the users typically use one piece of software on a regular basis.

The situation is complicated with network service; each host may provide a different number of services. If there are no services available on both hosts, then their default similarity is 1. Let's consider the two services the same if they are running on the same TCP or UDP port. Then, if the service is provided by one host and not by the other, this service can either be ignored or a predefined value of, for example, 0.8 can be used to indicate similarity. If both hosts provide a service on the same port, then their CPE strings (obtained via fingerprinting) are compared. The final similarity is then calculated as a multiple of all the factors.

Other features considered for similarity calculation are cumulative, the examples are the number of vulnerabilities on a host or a number of incidents in which a host was involved. In such case, the score is calculated as a quotient of the number of observations to the total number of observations. For example, let V be the number of unique CVEs observed in the network. Host h_1 was found to be vulnerable to v_1 CVEs, while host h_2 is vulnerable to v_2 CVEs. Then, the score of this feature (s) is calculated as v/V, i.e., $s_{h_1} = v_1/V$ and $s_{h_2} = v_2/V$. The similarity of the two hosts in this metric is calculated as $s = min\{s_{h_1}, s_{h_2}\}/max\{s_{h_1}, s_{h_2}\}$.

V. Example of Using the System

Let's illustrate the use of the system on a simple example inspired by incident handling practice. The incident handler receives a report from a user that their device is infected with ransomware. At the moment, it is not known how the ransomware infected the device. Before inspecting the infected machine, there is a need to identify devices threatened by the ransomware, warn their users and administrators, and check for possible backdoors and other traces of lateral movement of an attacker.

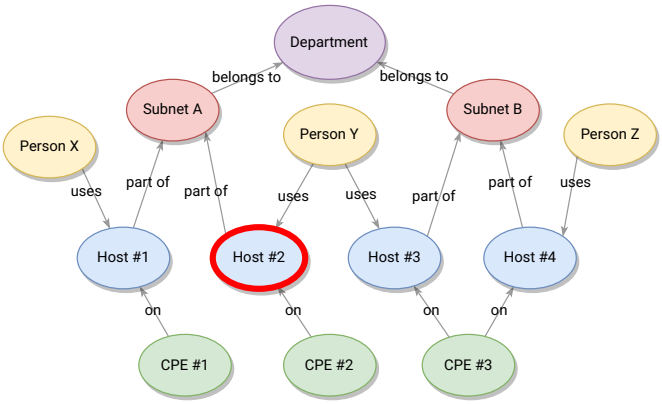

Fig. 1. A simple scenario illustrating the use of the system.

TABLE I
CPE strings in the example scenario.

CPE format	cpe:part:vendor:product:version:update:edition:language
Weights	0.5, 0.25, 0.125, 0.0625, 0.03125, 0.03125
CPE #1	cpe:2.3:o:microsoft:windows_7:-:sp2:*:*
CPE #2	cpe:2.3:o:microsoft:windows_7:-:sp1:*:*
CPE #3	cpe:2.3:o:microsoft:windows_10:-:*:*:*

The recommender system is used to create such a list. The system looks up the device in the database by its IP address and performs graph traversal to enumerate devices in close proximity. In a simple scenario presented in Figure 1, the IP address is found as *Host #2*. Two machines are found in close proximity, one in the same subnet (*Host #1*) and one under the administration of the same person (*Host #3*). The maximal distance is set to 3 and, thus, the device in the organization unit but in the different subnet (*Host #4*) is not included in the list due to exceeding the maximal distance.

Subsequently, the similarity of the infected devices and each of the hosts in the list is calculated. For simplicity, let's consider the situation in which only the OS fingerprints are available. See Table I for the details. The infected *Host #2* and *Host #1* have the same vendor, product, and version but differ in the remainder of the string. Thus, their similarity is calculated as $0.5 + 0.25 + 0.125 = 0.875$. The infected *Host #2* and *Host #3* have the same vendor but differ in the product, and, thus, their similarity is 0.5.

The two hosts are sorted by their score, i.e., $0.875/2 = 0.4375$ for *Host #1* and $0.5/2 = 0.25$ for *Host #3*. The list of hosts sorted in descending order by their score is returned to the incident handler, who then knows which hosts to focus on when mitigating the ransomware incident.

VI. Conclusion

We proposed a design of a recommended system for the needs of incident handling in computer networks. The system uses existing network scanning and traffic monitoring tools and knowledge of the local environment to find similar hosts in the network in close proximity. Thus, the system can, in

case of compromise of one of the hosts, immediately provide a prioritized list of other hosts in the network that are at risk or could be compromised, too. The system considers various ways of attack propagation or lateral movement, including self-propagating worms and the spread of malware via email and infected devices. This is especially important in handling incidents like ransomware, the major motivation for our work.

This paper is merely the first step in the future research, and, thus, in our future work, were are going to implement the recommender system and deploy it in an operational setting for evaluation. There will be a need to go through several past incidents and infer weights used in the calculation from it. Subsequently, the weights should be adjusted to ongoing incidents and experiences. Simultaneously, the list of features should be revisited if new or higher-quality data are available. The proposed system should also be integrated with other tools for incident handling, namely, to allow for visualization and click-through from one tool to another. Our efforts shall contribute to the orchestration and automation of security operation centers and incident response procedures.

Acknowledgment

This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

References

[1] "Computer Security Incident Response Team (CSIRT) Services Framework," https://www.first.org/standards/frameworks/csirts/csirt_services_framework_v2.1, 2019.

[2] E. Guttman and N. Brownlee, "Expectations for Computer Security Incident Response," RFC 2350, 6 1998.

[3] J. Muniz, G. McIntyre, and N. AlFardan, *Security Operations Center*. Cisco Press, 2016.

[4] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, "Computer security incident handling guide: Recommendations of the national institute of standards and technology," *NIST Special Publication*, vol. 800, no. 61, pp. 1–147, 2012.

[5] P. Kral, "The Incident Handler's Handbook," https://www.sans.org/reading-room/whitepapers/incident/incident-handlers-handbook-33901, 2012.

[6] J. M. Spring and P. Illari, "Review of human decision-making during computer security incident analysis," *Digital Threats: Research and Practice*, vol. 2, no. 2, 4 2021.

[7] J. Webb, A. Ahmad, S. B. Maynard, and G. Shanks, "A situation awareness model for information security risk management," *Computers & Security*, vol. 44, pp. 1–15, 2014.

[8] J. Happa, I. Agrafiotis, M. Helmhout, T. Bashford-Rogers, M. Goldsmith, and S. Creese, "Assessing a decision support tool for soc analysts," *Digital Threats: Research and Practice*, vol. 2, no. 3, 6 2021.

[9] S. Ossenbühl, J. Steinberger, and H. Baier, "Towards automated incident handling: How to select an appropriate response against a network-based attack?" in *2015 Ninth International Conference on IT Security Incident Management & IT Forensics*, 2015, pp. 51–67.

[10] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, "Decision support approaches for cyber security investment," *Decision Support Systems*, vol. 86, pp. 13–23, 2016.

[11] S. Noel and S. Jajodia, "Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs," *Journal of Network and Systems Management*, vol. 16, no. 3, pp. 259–275, 2008.

[12] N. Polatidis, E. Pimenidis, M. Pavlidis, and H. Mouratidis, "Recommender systems meeting security: From product recommendation to cyber-attack prediction," in *Engineering Applications of Neural Networks*, G. Boracchi, L. Iliadis, C. Jayne, and A. Likas, Eds. Cham: Springer International Publishing, 2017, pp. 508–519.

[13] L. Karlsson, P. N. Bideh, and M. Hell, "A recommender system for user-specific vulnerability scoring," in *Risks and Security of Internet and Systems*, S. Kallel, F. Cuppens, N. Cuppens-Boulahia, and A. Hadj Kacem, Eds. Cham: Springer International Publishing, 2020, pp. 355–364.

[14] P. Huff, K. McClanahan, T. Le, and Q. Li, "A recommender system for tracking vulnerabilities," in *The 16th International Conference on Availability, Reliability and Security*, ser. ARES 2021. New York, NY, USA: Association for Computing Machinery, 2021.

[15] M. Almukaynizi, E. Marin, E. Nunes, P. Shakarian, G. I. Simari, D. Kapoor, and T. Siedlecki, "Darkmention: A deployed system to predict enterprise-targeted external cyberattacks," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2018, pp. 31–36.

[16] M. Albanese, H. Cam, and S. Jajodia, *Automated Cyber Situation Awareness Tools and Models for Improving Analyst Performance*. Cham: Springer International Publishing, 2014, pp. 47–60.

[17] S. Jajodia, P. Liu, V. Swarup, and C. Wang, *Cyber situational awareness*. Springer, 2010, vol. 14.

[18] A. Kott, N. Buchler, and K. E. Schaefer, *Cyber Defense and Situational Awareness*. Springer, 2014, vol. 62.

[19] M. Husák, T. Jirsík, and S. J. Yang, "SoK: Contemporary Issues and Challenges to Enable Cyber Situational Awareness for Network Security," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ACM, 2020.

[20] R. Gutzwiller, J. Dykstra, and B. Payne, "Gaps and opportunities in situational awareness for cybersecurity," *Digital Threats: Research and Practice*, vol. 1, no. 3, 9 2020.

[21] A. Evesti, T. Kanstrén, T. Frantti, T. Kanstren, and T. Frantti, "Cybersecurity Situational Awareness Taxonomy," in *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. IEEE, 6 2017.

[22] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams, "Cauldron Mission-centric Cyber Situational Awareness with Defense in Depth," in *2011 – MILCOM 2011 Military Communications Conference*, 11 2011, pp. 1339–1344.

[23] S. Noel, E. Harley, K. H. Tam, M. Limiero, and M. Share, "CyGraph: Graph-Based Analytics and Visualization for Cybersecurity," *Handbook of Statistics*, vol. 35, pp. 117–167, 2016.

[24] J. Komárková, M. Husák, M. Laštovička, and D. Tovarňák, "CRUSOE: Data Model for Cyber Situational Awareness," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ACM, 2018.

[25] M. Husák, M. Laštovička, and D. Tovarňák, "System for continuous collection of contextual information for network security management and incident handling," in *The 16th International Conference on Availability, Reliability and Security*, ser. ARES 2021. New York, NY, USA: Association for Computing Machinery, 2021.

[26] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky, "Passive OS fingerprinting methods in the jungle of wireless networks," in *2018 IEEE/IFIP Network Operations and Management Symposium*, 2018.

[27] M. Laštovička, M. Husák, and L. Sadlek, "Network Monitoring and Enumerating Vulnerabilities in Large Heterogeneous Networks," in *2020 IEEE/IFIP Network Operations and Management Symposium*, 2020.

[28] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.

[29] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US), 2008.

[30] Cisco, "QoS: NBAR Configuration Guide, Cisco IOS XE Release 3S - Classifying Network Traffic Using NBAR," https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/configuration/xe-3s/qos-nbar-xe-3s-book/clsfy-traffic-nbar.html, 2018.

[31] M. Laštovička, S. Špaček, P. Velan, and P. Čeleda, "Using TLS Fingerprints for OS Identification in Encrypted Traffic," in *2020 IEEE/IFIP Network Operations and Management Symposium*, 2020.

[32] S. Lagraa and R. State, "What database do you choose for heterogeneous security log events analysis?" in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 812–817.