Recommending Similar Devices in Close Proximity for Network Security Management

Vladimír Bouček*, Martin Husák†
*Faculty of Informatics, Masaryk University, Brno, Czech Republic
†Institute of Computer Science, Masaryk University, Brno, Czech Republic
492927@mail.muni.cz, husakm@ics.muni.cz

***Abstract*—This paper presents a prototype of a tool for network security management that recommends similar devices in close proximity to a given machine. The task of recommending similar devices helps in analyzing the impact of cyber attacks, providing early warning and mitigating a spreading infection, or investigating an attack. Our tool uses modern graph-based technologies to store and query the data and existing data models that interconnect heterogeneous information about computer networks. By traversing the graph of network entities and calculating similarity scores, the tool suggests which devices are most likely to be exploited along with or after the exploitation of a device in question. The advantage of our tool is that it considers multiple attack vectors, including social engineering.**

***Index Terms*—Recommendation, Network security, Vulnerability, Incident response**

I. Introduction

Network security management involves numerous routine yet complex and time-consuming tasks. The automation of such tasks reduces the workload of network security operators, speeds up incident response, and facilitates threat assessment or digital forensics [1]. A specific task we aim to automate in our work is finding similar devices in the proximity of another device, which is a common subtask of many complex procedures, such as vulnerability management and incident response [2]. For example, during threat assessment, security analysts look up devices that can be exploited directly or following the exploitation of a different, nearby device. In digital forensics, the investigator explores the attacker's lateral movement, e.g., hopping from one exploited device to another. Such use cases are static, and the look-up of similar devices in proximity is not a time-critical operation. However, the situation changes in case of incident response and attack mitigation [3]. There is a need to act promptly, secure the surroundings of the exploited device, and send early warnings to administrators of devices around [1]. This is especially important in case of ransomware infections that spread fast and harm the organization significantly [4]. It is imperative to stop such infection early, and an automated recommendation of possible targets is highly beneficial. However, existing guidelines do not give advice on how to get an overview of incidents with incomplete data, i.e., how to generalize hypotheses in terms of time and space [5]. This is usually accomplished using the expert knowledge of a security team or database of all the data potentially usable in incident handling, such as Cauldron [6] or CyGraph [7].

In previous work, we designed and implemented CRUSOE, a toolset for achieving and maintaining cyber situational awareness [8]. CRUSOE is inspired by Cauldron [6] and CyGraph [7] and uses an orchestrated set of tools to collect and periodically update the information on the devices in the network using various passive and active network monitoring tools ranging from NetFlow traffic analysis [9] to Nmap network scanner [10] and dedicated vulnerability scanners. The data are stored in a graph database, an emerging and highly comprehensive approach to storing data in the form of labeled nodes and edges. CRUSOE toolset has been implemented and deployed in an operational environment [8]. However, the focus of the toolset is on the collection and presentation of the data; the capabilities to analyze the data or provide recommendations to the users are limited to mapping organizations' missions to network assets and otherwise left for future work, such as this one. The principles of a system that recommends similar devices in close proximity using the traversal of the graph-based data model were presented in another previous work [2] but not implemented. The research on decision support tools and recommender systems in cybersecurity was surveyed recently [1]. However, such research is usually focused on countermeasure selection [11], not suggesting future attacks or targets. Notable exceptions are the work of Polatidis et al. [12], who use a recommender system for attack prediction, and Albanese et al. [13], who proposed automated tools to improve analyst's performance.

The remainder of this paper is structured as follows. Section II presents the design considerations and overview of the proposed tool. Section III discusses the configuration and usage of the system using real-world data. Section IV concludes the paper.

II. Design and Overview

The tool is designed as a stand-alone piece of software. The code is publicly available on GitHub[1]. Nevertheless, the tool needs access to a database, where the data on a computer network are stored. As mentioned previously, the tool uses the database of the CRUSOE toolset [8], i.e., the Neo4j graph database, in which the data are structured according to the CRUSOE data model. The toolset can also be found

[1]https://github.com/CSIRT-MU/recommender-system-for-network-security-management

on GitHub for reference[2], although it is not needed to run the recommender system. A simple graphical representation of the database content is presented in Fig. 1.

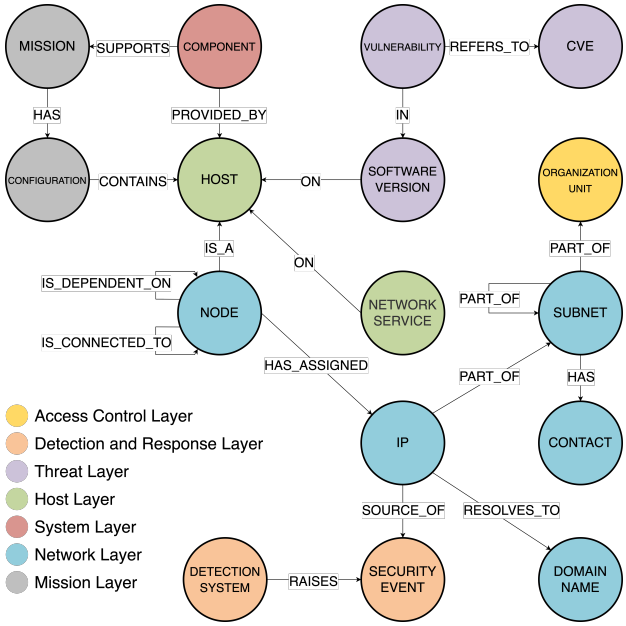

Fig. 1. The data model used in CRUSOE toolset.

The recommendations are based on the proximity and similarity of the devices in the network to the device on the input; similar devices in close proximity are prioritized. The calculation of similarity and distance was proposed in previous work [2]. Formally, the devices are sorted by their *risk score* (R), a quotient of *similarity score* (S) and *distance* (D) of the two devices. It is calculated as follows:

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

For practical reasons, the weights are assigned to partial similarity scores ($s_1...s_n$) and distances ($d_1...d_n$), as discussed later in the text.

The similarity is calculated as a product of all considered partial similarities, each having a value in the range $< 0, 1 >$. We assume that the malware spreads among devices with similar services, pieces of software, and software versions. CPE strings are widely used to represent a piece of software, including operating systems, and can be provided by network scanning software, such as Nmap [10]. Thus, one of the partial similarities is the similarity of CPE strings of operating systems of the two devices. Similarly, other partial similarities based on CPE strings are used to compare network services and their underlying software, such as the Apache web server or the OpenSSH server. Even the set of network services provided by the device is a subject of partial similarity calculation. Other partial similarities include the comparison of antivirus software, content management systems, vulnerabilities, and history of security events. More details on the calculation can be found in previous work [2], but the usable set of partial similarities depends on the available data and their quality.

The distance leverages the graph-based representation of data and stands for the shortest path between the two nodes representing devices in the network. The paths, however, have to conform to certain limitations esteeming from the assumed attack vectors. Worm-like malware spreads autonomously in the network, typically within the same subnet or towards consecutive IP addresses. In this case, the path between the nodes of type *IP* leads through the node of type *Subnet*. On the contrary, malware spreading via social engineering will more likely spread via email attachments, infected flash drives, or cloud storage shared within the same office or department or among the devices used by the same user. In this case, the path between the *IP* nodes will lead through the nodes of type *Organization Unit* or *Contact*.

The implementation uses breadth-first graph traversal starting from the node representing the given IP address. Nodes in Neo4j are not typed, so we had to implement a custom add-on to Neo4j's Cypher language that restricts the breadth-first search to certain node classes. Thus, all the above-mentioned classes are traversed, while irrelevant node classes like *Security Event* or *Vulnerability* are ignored.

III. System Deployment, Configuration, and Usage

The system was evaluated using the data on the campus network of Masaryk University collected by the university's cybersecurity team CSIRT-MU[3] using the CRUSOE toolset [8]. The campus network of Masaryk University serves more than 40,000 users and hosts numerous devices. Data were collected and periodically updated for three months so they reflect the long-term status of the network and its hosts.

In total, the resulting graph database representing the campus network contains 640,165 nodes and 12,765,385 edges of various types. More closely, the dataset contains information on 31,135 IP addresses providing 97,829 network services in 844 subnets and 48 organization units (e.g., departments); 647 different pieces of software and 2,930 estimated vulnerabilities were found in the network. The history of security incidents is represented by 395,815 events related to one or more IP addresses. The total size of the graph database used for testing was 3.5 GB.

A. Configuration

The system first needs to be configured. The configuration includes mostly the weights given to particular features of distance and similarity calculation. A sample configuration file structured in JSON format is presented in Fig. 2. The previous work [2] did not specify the weights; it would be extremely difficult to provide generic weights due to the difference between particular networks and observed phenomena. Therefore, we first estimated the weights by an educated guess during the development. Subsequently, we fine-tuned the weights by analyzing the dataset and finding average and

[2]https://github.com/CSIRT-MU/CRUSOE

[3]Computer Security Incident Response Team of Masaryk University

```
{
    "max_distance": 2,
    "path": {
        "apply": true,
        "subnet": 1,
        "organization_unit": 1.25,
        "contact": 1.15
    },
    "comparators": {
        "os": {
            "apply": true,
            "critical_bound": 0.34927222,
            "diff_value": 0.2,
            "vendor": 0.9,
            "product": 0.075,
            "version": 0.025
        },
        "antivirus": {
            "apply": true,
            "critical_bound": 0.5,
            "diff_value": 0.4,
            "vendor": 0.6,
            "product": 0.25,
            "version": 0.15
        },
        "cms": {
            "apply": true,
            "require_open_ports": false,
            "critical_bound": 0.44568431,
            "diff_value": 0.4,
            "vendor": 0.6,
            "product": 0.25,
            "version": 0.15
        },
        "net_service": {
            "apply": true,
            "critical_bound": 0.25,
            "diff_value": 0.1
        },
        "cve_cumulative": {
            "apply": true,
            "critical_bound": 0.29492334
        },
        "event_cumulative": {
            "apply": true,
            "critical_bound": 0.00036752
        }
    }
}
```


Fig. 2. Configuration file of the recommender system.

median values of similarity and distance features between the pairs of data entries. The supplementary scripts are parts of the software package and help set the tool automatically, without any expert input, only relying on the available data on the network.

The distance calculation setting uses the weights of the paths through nodes of certain types and the maximal distance. For example, in Fig. 2, we can see that the weight of the path going through the organization unit is bigger more the path going through the subnet. This means that two devices belonging to the same subnet are considered closer than two devices that share the same organizational unit. The maximal distance limits the length of the path used when searching for neighboring nodes from the node on the input. The default setting is 2. However, the distance settings are subject to change in future development and deployment.

The implementation of the particular similarity features uses so-called comparators. Each comparator can be turned on or off, for example, in case the data for that comparator are unavailable or of insufficient quality. The comparators can then be configured separately, and new comparators can be inserted into the tool. In the example in Fig. 2, we can see comparators having multiple weights that are used to fine-tune the resulting similarity score given by the comparator. For example, in *os* and *antivirus*, the different weights are used at different levels of detail so that the high-level data like *vendor* that are easy to obtain are valued more than exact *version* that is hard to extract. The other comparators may use *critical bound*, a threshold suggesting that the output of the comparator is to be used only when it exceeds a certain value. Again, this is implemented to prevent data of low quality or significance from skewing the final similarity score.

The exact values were calculated during the dataset analysis. For each comparator, the similarity of each pair of relevant, distinct database entries was calculated. For example, for calculating the mean bound of the OS fingerprint comparator, all the OS fingerprints in the database are involved in the calculation. All the pairs of fingerprints are compared, their similarity is calculated, and the mean value of all the calculated similarities is then used as a critical bound for the OS comparator. Mean bounds for other comparators are calculated correspondingly. Using this approach, the critical bound indicates whether the similarity value is above average among all the potential similarity values in the dataset.

B. Usage and outputs

We assume the system to be installed along with the CRUSOE toolset or elsewhere, given it can access the graph database. Both command-line and REST API interfaces are implemented. The user runs the system and specifies an IP address or domain name of a device in the network, from which to start the search for similar devices in close proximity. The specified device is typically reported to be infected. The recommender tool then queries the database and outputs a list of results formatted either as a table for human users or a JSON file for automatic processing. An example of the JSON-formatted output can be seen in Fig. 3. The user can choose between simple output and rich output with all the details. The details allow for the explainability of the recommendations and pinpointing important similarities between the two devices.

The system performs well with certain constraints; the most computationally extensive part is traversing the graph database. If the *max_distance* is set to 2, meaning that nodes in very close proximity are searched, and the whole recommendation process takes less than 10 seconds. However, setting the value higher to 4 or more, thus traversing a larger part of the graph, may cause the system to return results in 30 seconds or more.

```
[
    ...
    {
        "ip": "147.*.*.*",
        "domains": [
            "*.cz."
        ],
        "contacts": [
            "*@*.cz"
        ],
        "os": {
            "vendor": "linux",
            "product": "linux_kernel",
            "version": "*"
        },
        "antivirus": null,
        "cms": null,
        "cve_count": 932,
        "security_event_count": 183,
        "network_services": [
            {
                "service": "NTP",
                "port": 123,
                "protocol": "UDP"
            }
        ],
        "risk": [
            5.67566254945783e-06
        ],
        "distance": 4,
        "path_types": [
            "Organization"
        ],
        "warnings": [
            {
                "message": "Similar OS between
                    hosts.",
                "partial_similarity": 1.0
            },
            {
                "message": "High number of common
                    net services between hosts",
                "partial_similarity": 1.0
            }
        ]
    },
    ...
]
```


Fig. 3. Sample output of the recommender system.

IV. Conclusion

We presented a tool that, given an identifier of a device in the network, recommends similar devices in close proximity. Such recommendation is highly valuable in network security management, namely in vulnerability and attack impact assessment, early warning and mitigation, and digital forensics. A prominent and timely use case is the estimation of the spread of a ransomware infection and its mitigation [2], [4]. Because ransomware uses various attack vectors, our tool considers most of them simultaneously. The proximity of the devices is based not only on IP address similarity or belonging to the same IP range or network segment but also belonging to the same department, physical proximity, or the same user or administrator. The similarity is then based on OS and service fingerprints, common vulnerabilities and present security measures, and other features. The exact set of features may depend on the availability of the corresponding data.

In our future work, we are going to deploy the tool along with the instance of the CRUSOE toolset [8] used by the CSIRT-MU and enable the team to use its API or a web interface that we plan to develop. Subsequently, we are going to evaluate the usage of the system and its accuracy in recommending relevant devices. Moreover, we are going to research the possibilities of auto-configuring the tool, namely setting the weights of the comparators by mining historical security incident data or via genetic algorithms. The anonymized content of the database used for evaluating the system will be published as a dataset to enable the community to test the tool without the need to fill the database.

Acknowledgment

This research was supported by OP JAK "MSCAfellow5_MUNI" (No. CZ.02.01.01/00/22_010/0003229).

References

[1] M. Husák and M. Čermák, "SoK: Applications and Challenges of Using Recommender Systems in Cybersecurity Incident Handling and Response," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ser. ARES '22. New York, NY, USA: Association for Computing Machinery, 2022.

[2] M. Husák, "Towards a data-driven recommender system for handling ransomware and similar incidents," in *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2021.

[3] 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.

[4] T. McIntosh, A. S. M. Kayes, Y.-P. P. Chen, A. Ng, and P. Watters, "Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions," *ACM Computing Surveys*, vol. 54, no. 9, Oct 2021.

[5] 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.

[6] 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.

[7] 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.

[8] M. Husák, L. Sadlek, S. Špaček, M. Laštovička, M. Javorník, and J. Komárková, "CRUSOE: A toolset for cyber situational awareness and decision support in incident handling," *Computers & Security*, vol. 115, p. 102609, 2022.

[9] 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.

[10] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*, 2008.

[11] 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.

[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] 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.