CRUSOE: A Toolset for Cyber Situational Awareness and Decision Support in Incident Handling

Article *in* Computers & Security · January 2022

DOI: 10.1016/j.cose.2022.102609

Martin Husák
Masaryk University
46 PUBLICATIONS **612** CITATIONS
SEE PROFILE

Michal Javornik
Masaryk University
31 PUBLICATIONS **182** CITATIONS
SEE PROFILE

CRUSOE: A Toolset for Cyber Situational Awareness and Decision Support in Incident Handling

Martin Husák[a,*], Lukáš Sadlek[a,b], Stanislav Špaček[a,b], Martin Laštovička[a,b], Michal Javorník[a], Jana Komárková[a,b]

[a]*Institute of Computer Science, Masaryk University, Brno, Czech Republic*
[b]*Faculty of Informatics, Masaryk University, Brno, Czech Republic*

Abstract

The growing size and complexity of today's computer network make it hard to achieve and maintain so-called cyber situational awareness, i.e., the ability to perceive and comprehend the cyber environment and be able to project the situation in the near future. Namely, the personnel of cybersecurity incident response teams or security operation centers should be aware of the security situation in the network to effectively prevent or mitigate cyber attacks and avoid mistakes in the process. In this paper, we present a toolset for achieving cyber situational awareness in a large and heterogeneous environment. Our goal is to support cybersecurity teams in iterating through the OODA loop (Observe, Orient, Decide, Act). We designed tools to help the operator make informed decisions in incident handling and response for each phase of the cycle. The Observe phase builds on common tools for active and passive network monitoring and vulnerability assessment. In the Orient phase, the data on the network are structured and presented in a comprehensible and visually appealing manner. The Decide phase opens opportunities for decision-support systems, in our case, a recommender system that suggests the most resilient configuration of the critical infrastructure. Finally, the Act phase is supported by a service that orchestrates network security tools and allows for prompt mitigation actions. Finally, we present lessons learned from the deployment of the toolset in the campus network and the results of a user evaluation study.

Keywords: Cyber situational awareness, OODA Loop, Decision support, Network monitoring, Incident response

1. Introduction

Incident handling and incident response is a daily routine for cybersecurity teams within an organization, yet still remain a precarious task. Although many guidelines and handbooks [1, 2, 3] describing the procedures and best practices are of high quality and available for more than twenty years, there is a gap between the high-level procedures on one side and tools and specifics of the environment on the other side. Each network and organization is different, and there are a plethora of tools available that the cybersecurity teams may or may not use [4, 5], which makes it nearly impossible to write detailed guidelines and procedures with respect to available tools and environment. Still, the incident handlers (personnel responsible for incident handling and response) should be aware of the environment and available tools.

We are trying to solve a specific problem: the lack of procedures in incident handling that would tackle situational awareness and decision-making in the cyber environment. Cyber situational awareness (CSA) [6, 7] describes the perception, comprehension, and projection of future changes in the cyber environment, which is something all incident handlers should aim at but often do not manage properly [8]. Carefully observing and understanding the situation before making any decision is paramount for incident handlers. However, it requires good knowledge of the local environment and a well-equipped cybersecurity team [5]. The existing incident handling guidelines describe an overview or essence of the process [3] and particular tasks and phases [2], and summarize best practices [1]. However, CSA is either implicitly expected or mentioned without a deeper explanation. Although it is highly specific to a particular organization, a methodology should be elaborated at least for typical scenarios and environments.

We approach the problem by summoning the theoretical foundations of situational awareness and decision-making. Situational awareness in the cyber environment has been studied for a decade [6] and is a mature concept to follow [9, 8, 5]. Past research leveraged existing tools and found innovative use cases for them. For example, network measurements were successfully employed not only for intrusion detection and asset discovery but also for building a complex overview of the network [10, 11] and a so-called network-wide situational awareness [7]. At the same time, novel tools were proposed to correlate heterogeneous data on the cyber environment and visualize it [9]. Simultaneously, attack volume, network complexity, and lack of workforce in cybersecurity call for automation of incident response and development of decision-support systems.

The main contribution of this paper is the design and implementation of the CRUSOE toolset [12]. The toolset

[*]Corresponding author, email: husakm@ics.muni.cz

serves an incident response team of an organization and enables achieving CSA, decision support, and prompt incident response. The toolset builds on existing and widely-used techniques and tools and complements them with the implementation of missing parts. However, a technology-centric approach is insufficient in practice [13, 14, 15], and, thus, we employ the OODA loop, a procedure that guides a user in observing and understanding the situation and making the right decisions. In addition, we employ mission-centric approach to decision making that consider non-technical aspects of risk assessment and threat mitigation.

This paper is structured in nine sections. Background and related work are summarized in Section 2. Section 3 provides an overview of the proposed toolset. The tools inspired by the OODA loop are presented next; Section 4 discusses the Observe phase and data collection, Section 5 discusses the Orient phase and data comprehension, Section 6 discusses the Decide phase and decision support, and Section 7 discussed the Act phase and enforcing the decisions. Section 8 presents lessons learned from deployment in a campus network and the user evaluation study and its results. Section 9 concludes the paper.

2. Background and Related Work

Background and related work can be viewed in three dimensions. Herein, we discuss each dimension in a separate subsection. First, the overall theme of this paper is the CSA, which is a frequent topic of related work, mostly research papers. Second, the specific application domain of the work presented in this paper is incident handling, a fundamental service of a cybersecurity teams that is mostly discussed in technical literature in the form of best practices and standards. Third, a specific research topic discussed in this paper and in related work is decision support in cybersecurity. Finally, we comment related commercial solutions and our previous work.

2.1. Cyber Situational Awareness

This work was vastly inspired by the theoretical foundations of situational awareness, which was extensively studied in military and aviation, and gained recognition in cybersecurity as CSA [6]. 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 [7]. Each level builds upon the previous one; one cannot fully understand the situation without proper perception nor project the situation without understanding it. A survey of literature on CSA was presented by Franke and Brynielsson [16], and the recent status of research in this area was summarized by Liu et al. [9] and Husák et al. [5].

The fundamental results were achieved and the attention has moved towards their application, deployment, and

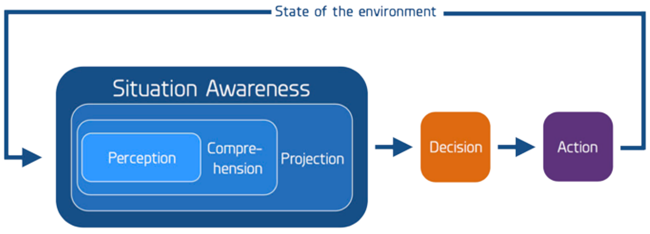


Figure 1: Three-level model of situational awareness.

field studies [5, 8]. Achieving CSA requires a vast range of heterogeneous information that can be provided by a plethora of existing tools [4]. However, it is difficult to process such data, namely because their volume, velocity, and variety effectively make them big data [5]. 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 [5]. Complex tools to enable CSA were proposed in the past, Cauldron [10] and CyGraph [11] 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 [17]. Tools enabling network-wide situational awareness via network traffic monitoring were proposed in the past, examples are NVisionIP [18] and VisFlowConnect [19] from 2004, Nfsight [20] from 2010, and InSight [21] from 2020. The topic and themes are returning with novel tools and approaches to network monitoring and accompanying challenges, such as the analysis of encrypted network traffic and real-time data processing [22].

2.2. Incident Handling

Incident handling is a fundamental activity provided by a cybersecurity team (CSIRT[1]/CERT[2]). For example, the CSIRT Services Framework enumerates services related to incident management that CSIRT teams can provide [23]. 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 [24]. Recently, the incident response became a part of third-generation SOC (Security Operations Center) as defined by CISCO [25].

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 [3] by

[1]Computer Security Incident Response Team
[2]Computer Emergency Response Team

NIST divides incident handling into preparation, detection, analysis, containment, eradication and recovery, and post-incident activities. The guide promotes situational awareness and formulates the need to ensure that incident response procedures are in sync with business continuity processes. The business continuity planning professionals should be aware of the incidents and their impact and are valuable in planning responses to certain situations. The Incident Handler's Handbook [2] 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 business impact and scale. Further, the incident handler should plan the containment, eradication, and recovery by questioning whether the impacted systems can be isolated, patched, turned off, or kept running. The incident handler has to be aware of the impacted hosts, their neighborhoods, and dependencies to properly plan the course of action. Similarly, the incident management guide by ENISA [1] provides practical examples of incident handling issues and summarizes best practices.

Incident handling was a topic of research of Ahmad et al. [13], who found out that practitioners focus on technical aspects while neglecting strategic decision making and are more likely to review high-impact incidents than 'near misses' and incidents from which the team could learn. Recently, Ahmad et al. [14] conducted a case study of developing CSA in an organization for the needs of incident response. The authors emphasize the management practice in contrast with the technological perspective that was dominant in the past research.

2.3. Decision Support in Cybersecurity

Incident handlers are forced to make many prompt decisions, especially during incident triage (prioritization) and their resolution. Decisions are often accompanied by the impact assessment of incidents and security countermeasures. However, according to Spring and Illari, who reviewed human decision-making in incident response [26], 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 expert knowledge of security teams.

Therefore, the researchers usually focus on partial subtasks in decision support, such as finding an optimal countermeasure for ongoing or possible cyber attack out of available response actions [27], optimal purchase of defense tools [28], or their optimal placement [29]. The comprehensive survey of countermeasure selection by Nespoli et al. [30] found out that current methods are often limited in scalability, set of considered countermeasures, and weak in determining countermeasures applicable for a specific type of attack. Besides, standard representation of countermeasures is still missing, except for the very recently published D3FEND framework[3] by MITRE.

Our proposed toolset [12] leverages mapping of enterprise missions to cyber assets for decision support, which was also a topic of related work. For example, Goodall et al. proposed CAMUS – a proof of concept system for automatic mapping of cyber assets to missions and users [31]. Guion and Reith surveyed tools and methodologies for mapping missions to cyber key terrain, including systems, devices, software, and other network entities [32]. A graph-based mission dependency model for CSA compatible with CyGraph and other MITRE tools was proposed by Heinbockel et al. [33]. Two key aspects shall be considered for the use of such mappings. First, missions are usually captured statically, and collecting of vulnerability and threat information is required to perform the risk assessment for missions [32]. Second, the mission-centric risk assessment methods consider that mission can reach its objectives even in the presence of threats and vulnerabilities [34]. Recently, Rodriguez-Bermejo et al. [35] proposed an evaluation methodology for mission-centric CSA capabilities. Happa et al. [15] studied the 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; such themes are addressed in our work.

2.4. Other Related Work

Several pieces of related work are relevant to our work as a whole. For example, Webb et al. [36] proposed a model for security risk management based on the concepts of situational awareness that addresses the deficiencies in practice and poor decision making. Similar tools and toolsets were proposed in the past, although not often reaching implementation. For example, Albanese et al. [37] proposed automated CSA tools to improve analyst's performance. Decision support and risk assessment also enabled by Cauldron [10] and CyGraph [11]; the later also suggest mitigation options.

Practitioners and the industry are also embracing the principles of the OODA loop [8]. For example, AT&T (formerly AlienVault) proposes the use of the OODA loop in incident response [38], although only as a guideline for incident handlers, no novel tools are presented. Similar to our work, they recommend using existing tools for the Observe and Orient phases. In the Decide phase, AT&T recommends using corporate security policy and hard copy documentation to make human decisions; machine-assisted decision making is not yet an industrial standard. Act phase by AT&T focuses on forensics, patch and backup management, and user awareness, which are quite wide topics not very close to prompt incident response that we consider in our work.

[3]`https://d3fend.mitre.org/`

It is worth noting that this paper summarizes long-term research and development effort and builds upon partial results presented earlier. In our previous work, we typically discussed a design or implementation of a certain component or proposed a method that was later implemented in the presented toolset, namely the design of the data collection toolset [39], underlying data model [17], dashboard [40], and decision support [41, 42]. Components of the toolset were also used in research on fingerprinting devices in the network [43, 44, 45], vulnerability enumeration and categorization [46, 47], and incident suppression [48, 49]. Readers interested in technical details and kindly referred to these publications; references are also provided in appropriate sections.

3. Applying the OODA Loop in Cybersecurity

In this paper, we propose CRUSOE [12], a toolset that enables CSA and decision support for incident handling and response. Specifically, we were inspired by the OODA loop, a concept of a recurring cycle of decision-making. In this section, we first introduce the OODA loop and illustrate its use in cybersecurity. Subsequently, we specify the requirements on a toolset that supports the use of the OODA loop in incident handling and response. Finally, we present the overall design of the toolset. A detailed description of the particular tools and components is described in the following sections.

3.1. OODA Loop in Cybersecurity and Incident Handling

The OODA loop was proposed by John Boyd for the need of military aviation and has been used in other fields, most importantly in cybersecurity [7, 50]. The OODA loop consists of four phases (Observe, Orient, Decide, and Act) that are recurrently iterated. The Observe phase stands for the collection of information about the environment. The Orient phase stands for analysis and synthesis of the data and comprehending the environment. The Decide phase represents the decision-making and setting the course of actions. The Act phase represents the execution of the actions and interactions with the environment. It is possible to return to the Observe phase from any phase if the situation changes or new information is required.

Figure 2 illustrates the implementation of the loop, including the possible inputs and feedback, in cybersecurity. The Observe phase relies on continuous monitoring and collection of information on the computer network and information systems, including discovering vulnerabilities, network traffic analysis, host identification, and observation of measurable events, such as intrusion detection alerts. The Orient phase may include analysis and synthesis of the data via alert correlation and other means. However, for a human analyst, a comprehensive visualization of the situation in the network is highly beneficial. The Decide phase involves decision-making, such as selecting the mitigation strategy with security policies in mind.

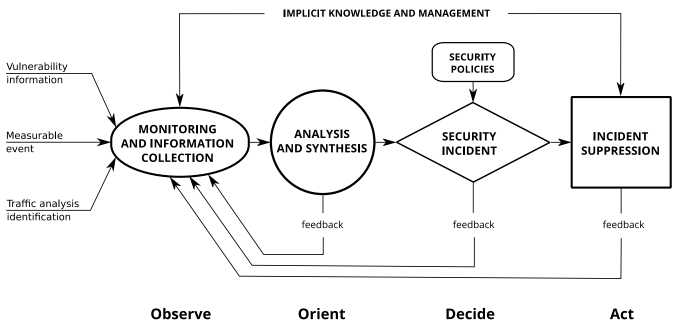


Figure 2: OODA Loop in Cybersecurity.

The incident suppression, such as vulnerability patching or attack mitigation, happens in the Act phase.

An important point to mention is that the iteration through the OODA loop is not something that a computer should do; it is a human operator's mental process. Thus, the toolset is limited to helping the operator, not taking over the decision-making process. The tools provide information and allow taking actions; the responsibility for the actions must remain to the incident handler.

3.2. System Requirements

In our work, we consider the following scenario for the deployment of the CRUSOE toolset. An organization operates a large network with thousands of computers and thousands of users. The network is heterogeneous and partitioned into many segments with various characteristics, e.g., server segment, office desktops, and Wi-Fi access points. The network is administered by a central IT department and a number of local IT administrators at the organization's departments and remote locations. The organization also operates a security team that monitors the network and network traffic and mitigates cyber-attacks via active network defense devices. The expected user of the system proposed in this work is an incident handler of the organization's cybersecurity team. Nevertheless, parts of the system shall also be open to security analysts, managers, and other stakeholders.

The requirements on the CRUSOE toolset were distilled from the requirements on CSA systems in the NATO Cyber Defense Situational Awareness Request for Information [51]. The 35 use cases frequently mention views on assets and their interconnectivity, dependencies, and historical incidents. We also embraced the issues of system modularity and deployability. On the contrary, we did not consider the use cases related to training and simulation because we perceive them as out of the scope of our work or candidates for future work. The interviews with incident handlers of several cybersecurity teams added two more requirements [17]. First, the contacts on persons responsible for administering hosts and services in the network should be easily accessible for incident handlers, which is an issue in large networks with many departments and their local

IT staff. The incident handlers often lose time on finding the right person to resolve an incident with. Second, the incident handlers would appreciate the automation of triage, i.e., prioritizing the incidents by their severity and significance [1].

The system shall first collect all the necessary contextual information on hosts and services in the network [39], structure and interconnect the data, and present them in a comprehensive manner. It is worth mentioning that these requirements are not strict. Even with lower confidence, any piece of information is valuable as it provides at least some context. The goal is not to provide the incident handler with all the information at the highest quality because that would be enormously difficult to achieve. The goal is to gather the basic information on hosts in the network that would otherwise be unknown to the incident handler. Thus, in this case, we prefer a network-wide solution with lower precision over a specialized solution with limited scope.

Assuming the data on the network are collected and accessible, there is a need to support incident response procedures, including decision-making of incident handlers and rapid mitigation of ongoing attacks. The key question of incident prioritization and decision-making in cybersecurity is how a cyber-attack or exploitation of a vulnerability threatens the organization. Mapping the cyber assets to organization mission is a challenging task discussed in the literature [32, 34] and approached by CSA tools [31, 11]. Manual mapping is laborious and prone to omissions, and automated discovery of the mapping is imprecise, making achieving an exact mapping very difficult. Nevertheless, the system shall suggest the impact of an attack on the organization and evaluate the possible mitigation options from the perspective of business continuity. The system shall recommend the course of actions that would lead to a situation in which the risks are reduced, and the network infrastructure provides all the necessary services with respect to security and functional requirements posed on them. The execution of the recommended course of action shall then be facilitated for the incident handler.

3.3. System Design

The CRUSOE toolset consists of four pieces of software; each piece of software helps the user in iterating through one of the phases of the OODA loop. The tools are referred to by the name of the OODA loop phase they represent, i.e., Observe, Orient, Decide, and Act. The toolset design is depicted in Figure 3. The four tools are highlighted by backgrounds of different colors and are further structured into components. The pieces of software are described in detail in the following sections, so only general information and design consideration of the whole toolset are discussed here.

First, the Observe phase is supported by a data collection system [39] The Observe tool orchestrates a plethora of components that collect information on the network and that are presented in Section 4. The most important components process data from active and passive network measurements. The tool is fully automated; no user interactions are expected except for system administration.

Second, the Orient phase is supported by a database to store the data provided by the Observe tool and a dashboard to visualize them, both presented in Section 5. The database is structured according to the custom data model [17] that allows for connecting heterogeneous data. The dashboard [40] facilitates user interaction with the CRUSOE toolset and visualizes the collected data, which enables the comprehension of the cyber environment. The dashboard also provides the user interface to the Decide and Act tools (see Section 7.2).

Third, the Decide tool provides decision support for the user iterating through the Decision phase of the OODA loop. Related work focused on selecting an optimal defense strategy [28, 29, 27]. In practice, however, there is often a limited number of mitigation tools and actions [30], and the use of an advanced recommender system would not be beneficial for its cost. Therefore, we chose to design a decision support system that recommends the most resilient configuration of the protected infrastructure [42]. The system uses stochastic reasoning and leverages mapping of enterprise mission to cyber assets [41] and is presented in detail in Section 6.

Finally, the Act tool, presented in detail Section 7 enables orchestration of active network defense devices and rapid attack mitigation via command propagation from the Dashboard through the unified interface of the orchestration service to a specialized device. The devices also provide feedback available to users in the Dashboard. Similar and even more advanced functionality can be achieved via Software-Defined Networking and related technologies, but these are still not yet widely used or available. Therefore, instead of using state-of-the-art solutions, we implemented a simpler tool usable in conventional networks that allows for deployment and evaluation in a live network without further prerequisites.

4. Observe

The first step in achieving CSA is the perception of the environment. In the cyber environment, this is achieved by collecting the data from sensors, such as active and passive network monitoring tools, intrusion detection systems, and vulnerability scanners. Following the requirements on the CRUSOE toolset, we first derived a list of information to collect. Subsequently, we designed a system for the collection of such data. The system consists of an orchestration service and a set of data collection components that connect to various data sources. The readers interested in technical details are kindly referred to the design paper of the Observe tool [39] and the toolset implementation [12].

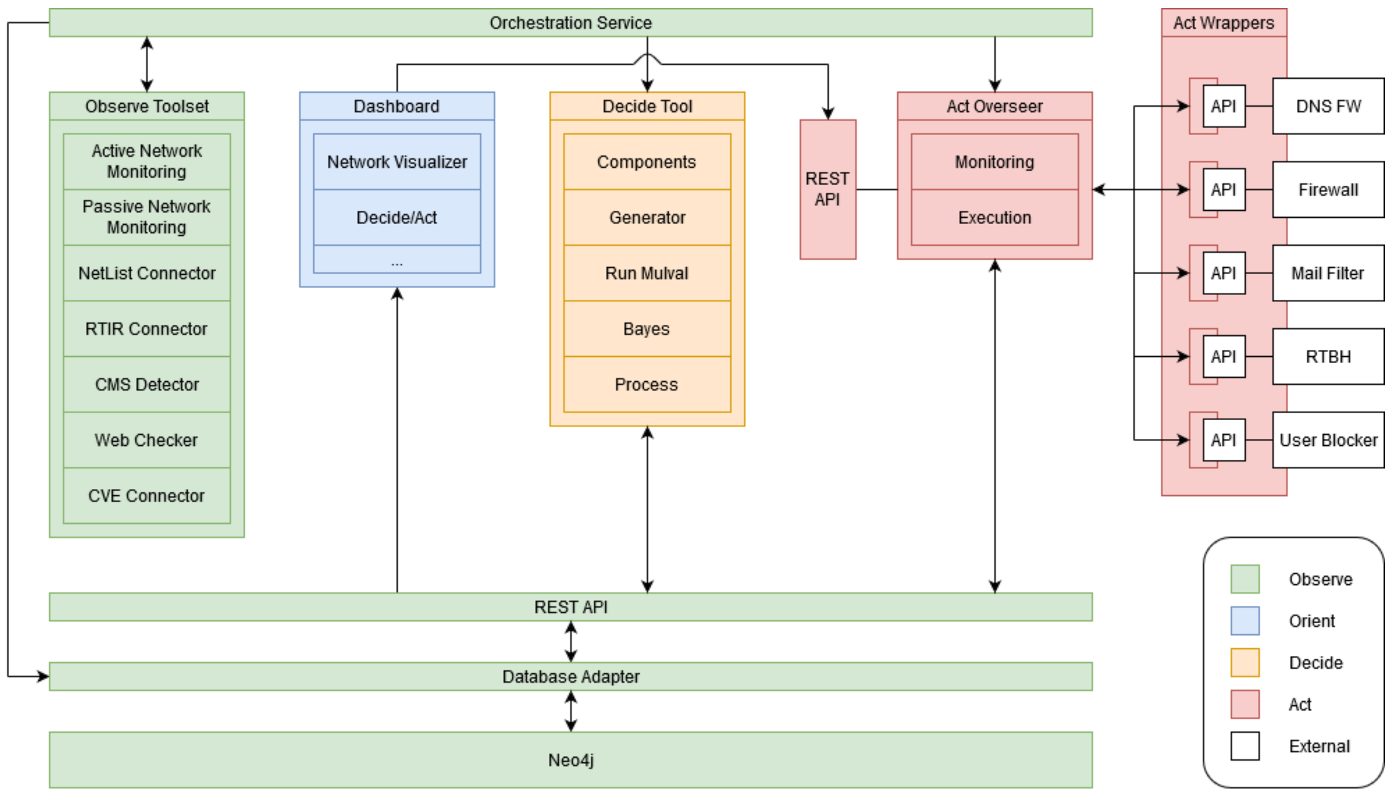


Figure 3: Design and architecture of the CRUSOE toolset.

4.1. Data to Collect

The data to collect can be categorized into network-related and host-related. For the whole network, there is a need to collect the following:

- Network topology, enumeration of network segments, and enumeration of hosts – it is vital to know how the network is structured, where is each segment located (e.g., physical location, department of the organization, physical or virtual machine), and what is its purpose (e.g., workstation and server segments, IP pools of VPN or Wi-Fi access points).
- Contact information on responsible person for each network segment – who is the primary user or responsible IT administrator, who shall be contacted in case of an incident.
- Enumeration of critical infrastructure and its dependencies – which assets are the most valuable or critical for the organization.
- History of security incidents – namely to check if there was a similar incident in the past on the same host or in its proximity or if a particular host or network segment is known to be frequently exploited.

For each host, there is a need to collect the following:

- Operating system (OS) name and version – to estimate the role of the host in the network (e.g., workstation or server); obsolete OS is also a security risk.
- Open ports and network services, including the name and version of the underlying software – namely remotely accessible services are common targets.
- The name and version of Content Management System (CMS) on servers providing web services – CMS are frequently exploited due to the number of known vulnerabilities in older versions and the fact that users often ignore updates.
- TLS certificate, in case HTTPS is used – it is important to know if the certificate is valid.
- Name and version of a web browser used on the system – in case the web browser is the attack vector.
- Name of the antivirus software running on the system and its latest update – to see if the host is sufficiently protected.

The collected data shall be timely and reflect the current state of the network. However, we avoid collecting the data on-demand as that would create an unnecessary delay when they are needed. Instead, we want a system that runs in the background and collects and updates the data continuously. An older entry is not necessarily unusable if it was collected several hours or days ago. It is often advantageous to even have a longer history of the collected data. For example, if a device is turned off and was last active several days ago, its OS fingerprint from those days is more informative than a futile attempt to collect it on demand. Similarly, if there are multiple hosts behind a single IP address, each with a different signature, it is advantageous to have a historical record that suggests which signature is primarily active or if they change regularly.

4.2. System Design and Implementation

The system to support the Observe phase is modular and consists of an orchestration service and a set of data collection components. The scheme of the system is depicted in Figure 3. The orchestration service schedules and executes the runs of the data collection components and saves them in the database that will be discussed in detail in Section 5.2. The data collection components are grouped in several logical blocks; each block's components provide similar functionality (e.g., asset discovery or vulnerability assessment) or process the same input data (e.g., network flows or incident history).

The implementation of the orchestration service is using Celery task queue[4] supported by in-memory database Redis[5]. Running a data collection component is referred to as a task. Each task has its role, which defines its timing and execution parameters. The records on the execution are logged, so it is possible to trace if the task finished or failed, how long it ran, its output, and possible errors.

The data collection components conduct the collection of the particular pieces of information. The components often overlap in the types of data they provide, and more components can obtain the same information. However, this is more of an advantage; the components rather complement each other; some may provide more detailed or precise information, while others may have a broader scope and provide at least partial information on an asset that would otherwise remain unnoticed. Table 1 provides a list of the components implemented at the moment and the types of data they provide.

The majority of the data is obtained via passive and active network monitoring. Passive monitoring is represented by network flow (NetFlow) technology [52], a popular tool among the cybersecurity community that enables network-wide situational awareness [18, 19, 20, 21]. NetFlow monitoring aggregates network connections into 5-tuples (source IP, destination IP, source port, destination port, protocol) and collects additional information, such as length of the connection or the number of bytes and packets exchanged. Although NetFlow does not process the packet payload, it provides sufficient visibility into network traffic even in high-speed and large-scale networks; and can be used for intrusion detection [53] and traffic analysis, including the analysis of encrypted traffic [45]. Active network monitoring is represented by a well-known Nmap tool [54]. Active probing can be used to check the liveness of a host, discover open network services, get a host's fingerprint, and so on. It is also a popular choice for vulnerability scanners, e.g., tools that discover vulnerabilities on hosts in the network. Both active and passive network monitoring have their benefits and drawbacks, but the combination and correlation of their outputs can be very powerful.

4.3. Selected Data Collection Components

A number of data collection components are already implemented, and the Observe toolset can be extended with others. Herein, we present the most important ones, namely the passive and active network monitoring components based on NetFlow and Nmap. Subsequently, we briefly comment on other components. If possible, we prefer wrappers of existing tools. Unfortunately, specific pieces of information can be obtained only from specific local sources. Such cases are highlighted in the text, and possible alternatives are proposed.

The passive network monitoring component analyzes network traffic provided by NetFlow monitoring infrastructure. The component conducts the fingerprinting of OS, services, and applications running on the hosts in the network. The OS fingerprinting implements three methods described in related work: detection of communication with specific domains, analysis of HTTP User-Agent, and analysis of default values in TCP/IP headers [43]. The TCP/IP header analysis is further improved with machine learning [44]. The User-Agent analysis is getting obsoleted and will soon be replaced due to the growing adoption of encryption, which prevents analysis of HTTP headers [45]. The detection of communication with specific domains and analysis of HTTP User-Agent is also used to detect the web browser used on the host. Communication with specific domains is further used to detect the presence of antivirus software and estimate the last time of its update; the specific domain is typically an update server. In both cases, the result with the most matches is selected if more different ones are found. Finally, a machine learning classifier is constructed to estimate the services and software running on the hosts in the network. The classifier uses the features such as traffic volume, packet size, packet distribution over time, and TCP/IP header settings and is trained on the network flows containing NBAR2 signatures [55].

Active network monitoring (scanning, probing) complements passive network measurements in the enumeration of active hosts and services in the network and their fingerprinting. The active probing further enables network topology discovery. We designed active network monitoring components as a wrapper to Nmap [54], a well-known and widely-used tool for active probing. Scanning a large network may take a non-trivial amount of time, and, thus, the design takes advantage of parallel scanning from multiple observation points. Two types of scans are performed, network scan and topology scan. The network scan is executed to discover running hosts and services in the network and to identify the software they use. The 100 most commonly used ports are scanned on each IP address. If an open port is found, the advanced features of Nmap are

[4]`https://docs.celeryproject.org/en/stable/`
[5]`https://redis.io/`

Table 1: Overview of data to collect and data collection components.

Data to collect	**Component**	**Approach and Tool(s)**	**Notes**
Host enumeration	Active network monitoring	Responding devices	Any newly observed IP address is added to the list
	Passive network monitoring	Communicating devices	
Network topology	Active network monitoring	Nmap's traceroute method	More precise when executed from several distinct observation points
Network segments	NetList component	Custom input	Environemnt-specific, shall be customized before deployment
Contacts on admins			
Critical infrastructure	Manual input	Custom input	Precise representation for the needs of Decide system (see Section 6)
	Criticality estimator	Betweenness score in network topology graph	Automated, less precise approximation (see Section 5)
History of incidents	RTIR connector	Export from RTIR	Shall be customized to read data from request tracking or incident handling system used in an organization
OS fingerprint	Active network monitoring	Nmap	Lower coverage, higher precision
	Passive network monitoring	Work by Laštovička et al. [43, 45]	Higher coverage, lower precision, multiple approaches
Network service fingerprint	Active network monitoring	Nmap	Nmap's 100 top ports
	Passive network monitoring	Machine learning model	Trained on NBAR2 [55] labeled data
CMS name and version	CMS detector	WhatWeb	Detects name and version of content management system and their plugins
TLS certificate validity	Webchecker	OpenSSL	Checks for (soon to be) expired, self-signed, and invalid certificates
Web browser fingerprint	Passive network monitoring	HTTP User-Agent analysis	Can be complemented by TLS fingerprints [45]
Antivirus software	Passive network monitoring	Detection of communication with specific domains	Last communication with update server indicates last update
Vulnerability Information	CVE connector	Downloads CVE records	Downloads feeds from NVD and several vendors' vulnerability databases

used to further probe the service on the port to identify its software and version. The topology scan is executed to map the network topology and distinguish active network devices from end hosts. The scan is repeatedly executed using Nmap's traceroute method from multiple observation points. A scan from a single observation point would create only a tree structure; the more observation points and repetitions are used, the more is the network topology map similar to the actual network topology. The hosts and services in the network can be arbitrarily deactivated or stop responding, which may interrupt the network topology scan. Thus, the component excludes the unfinished scans from the results.

The outputs of the active network monitoring component are further enriched by two other components that focus on webservers. First, the Webchecker checks if the webserver provides content over HTTP, HTTPS, or both protocols and if the HTTPS server has a valid TLS certificate. Invalid or self-signed certificates are common weaknesses in today's network. Second, the CMS detection component uses WhatWeb tool[6] to name and version of a Content Management System (CMS) and its plugins deployed on the webserver. Popular CMS and their plugins are widely exploited due to their high accessibility, the number of exploits, and low frequency of updates by their administrators. Although there is a need to use additional tools or components to detect them, the task is not complicated, and there are numerous tools available.

The information on vulnerabilities are provided from external sources by the CVE connector named by the CVE records[7], in which the data are structured. The purpose of this component is to put together available information on vulnerabilities discovered in the network so that the incident handler can be immediately provided with a human-readable description and technical details on each vulnerability with an assigned CVE identifier. The data are regularly downloaded from NVD[8], a *de facto* standard library of vulnerabilities. However, it is advantageous to complement the data from NVD with data from vendor's databases, such as the ones provided by Apple, Cisco, Microsoft, or RedHat. Such databases contain fewer vulnerabilities, often only on the vendors' products, but with more detailed and timely information.

The history of cybersecurity incidents associated with a network entity is highly interesting contextual information. Cybersecurity teams typically use incident management or ticketing system to keep track of currently handled

[6]`https://morningstarsecurity.com/research/whatweb`

[7]Common Vulnerabilities and Exposures, `https://cve.mitre.org/`

[8]National Vulnerability Database, `https://nvd.nist.gov/`

incidents; an example of such a system is RTIR[9]. We designed a component that periodically checks RTIR for the status and history of incidents and links them with entities identified by other components. Unfortunately, due to the lack of standards and the number of competing systems for incident handling, there is a need to implement components that would gather similar data from other systems, such as TheHive[10].

Finally, the information on network partitioning or segmentation and contact on administrators of each segment are typically not available unless an organization operates asset management or a similar system. Still, there is no standardized solution on how to store and represent such data. We implemented a simple component that reads a CSV-formatted list of network segments, their designation (e.g., workstations of a department, centrally administered servers, VPN address pool), and emails on their administrators. The component can be replaced with a connector to a particular asset management system or customized to the needs of an organization. Although there are approaches to automatically discover network segmentation, e.g., via behavioral analysis of the hosts [56], there is often a need to keep such information set in stone and use the automated methods for anomaly detection or similar uses.

5. Orient

The goal of the Orient phase is to comprehend the situation and processes in it. Therefore, the tools supporting this phase shall structure the collected information and present them in a comprehensible manner. First, we present the CRUSOE data model, which enables structuring the data collected by the Observe tools. Subsequently, we comment on the database and data processing implementation, including technology selection, detection of critical infrastructure, and matching vulnerabilities to hosts in the network. Finally, we briefly present the Dashboard, a web application that visualizes the data.

A common theme in CSA research is using a graph-based approach to model the data [10, 11, 17]. Graph-based approaches allow for extensible and comprehensive modeling of heterogeneous data and their straightforward visualization. Further, graph databases are becoming increasingly popular and mature technologies even for big data processing in operational deployment, which makes them a solid technological choice [57].

5.1. Data Model

The data collected in the Observe phase shall be structured before being stored and used. Several data models were proposed in the past, M2D2 [58] to correlate alerts and vulnerabilities, Virtual Terrain [59] for modeling the network, or CAMUS [31] to map cyber assets to enterprise missions and users. Nevertheless, we designed our custom data model [17] inspired by CyGraph [11], a complex CSA tool backed by a graph-based data model. Although CyGraph is well thought out, it is laborious and expensive to populate its database with all the necessary data, especially with the heterogeneity of tools and lack of standardization and workforce in cybersecurity nowadays. Our proposed data model is more lightweight and reflects the specific nature of data obtained via available automated sources such as network monitoring and vulnerability databases. Our data model is also inspired by the layered design of the risk assessment model by Innerhofer-Oberperfler and Breu [60]. Our model is logically structured in seven layers, each describing a different domain, namely: host layer, network layer, threat layer, detection and response layer, access control layer, system layer, and mission layer [17].

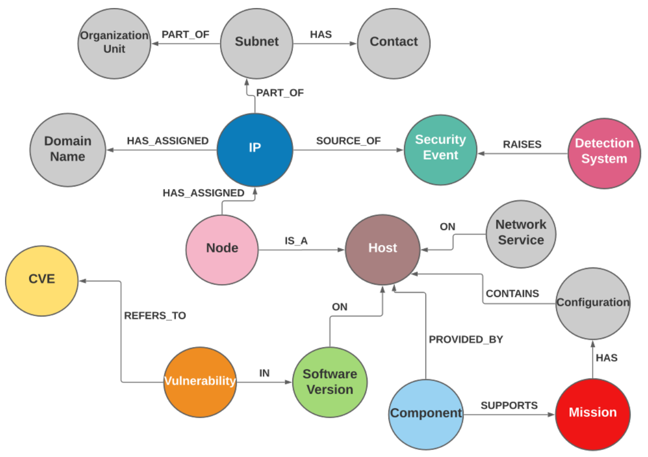


Figure 4: An excerpt from the CRUSOE data model [17].

The graph-based data model allows storing entities and relations as vertices and edges of a graph. The entities and relations capture situations such as *vulnerability X was detected on IP address Y*, *software W is running on the IP address Y*, and *software W supports enterprise mission Z*. This allows complex queries, such as *which vulnerabilities threaten an enterprise mission?* The detailed specification of the data model is presented in our previous work [17]. An excerpt from the data model can be seen in Figure 4. The excerpt illustrates the entities and relations processed by the toolset; the original data model proposal is richer but filling it all would require additional tools.

5.2. Database and Data Processing

The system is backed by a graph database Neo4j[11]. The data in the database are structured using the data model with several minor corrections and detailed specification of particular parameters of nodes and edges as compared to the paper version of the data model [17]. The

[9]https://bestpractical.com/rtir
[10]https://thehive-project.org/

[11]https://neo4j.com/

database is accompanied by REST API and a database adapter. The database adapter facilitates the insertion of data from Observe tools to the database so that the Observe tools do not need to access the database directly or use Cypher, Neo4j's querying language. The REST API provides CRUD (Create, Read, Update, Delete) functionality to all entities specified by the data model so that they can be accessed and modified by other tools, such as Decide and Act. The REST API is implemented using the Django framework[12].

Neo4j provides a set of graph algorithms we used to implement criticality estimation functionality for the CRUSOE toolset. Although this was implemented as a component of the Observe tool, it does not collect any new data but only extracts additional information from the data already present in the database; the orchestration service allows to schedule the execution of the calculation. The *Criticality Estimator* [39] implements an experimental approach to automatic detection of critical or important hosts in the network, a task that is otherwise laborious and prone to omissions when done manually. The network topology (discovered by the active network monitoring component) is a subgraph of the complex graph stored in the database. We define *critical host* in the network as a host that has a high impact on the flow of information in the network. A *critical host* is represented as a *critical node* in the network topology graph. We assume that removing a *critical node* would decrease or remove the graph's connectivity in terms of graph theory. We apply two graph-theory algorithms implemented in Neo4j. First, the algorithm assessing the impact of the nodes on the flow of information in the network is based on the *betweenness* algorithm. The algorithm computes the shortest paths between each pair of nodes via breadth-first search. Subsequently, each node is weighted by the number of shortest paths going through that node. Nodes with the highest weight are considered the most important or critical for the network. Second, the node popularity is measured by enumerating the incident edges of each node using the *degree centrality* algorithm that first selects the set of nodes and then enumerates all the incoming and outgoing edges of eligible types. Each node is weighted by the number of incident edges, and the nodes with higher weight are considered popular and, therefore, possibly critical.

A crucial issue for the CRUSOE toolset is matching the software instances with vulnerabilities. The toolset uses network measurements to identify the name and version of the software running on the hosts in the network; these are stored in the CPE format[13]. The CPE is used in CVE to identify which versions of the software are affected by a vulnerability. CRUSOE exploits this by matching the CPE records found via OS and service fingerprinting with the CVE records downloaded from NVD and other databases. The CPE records in network measurements and vulnerability databases are not exact; mostly only vendor and product names and product's major version can be found in the records, not minor version nor update [47]. The exact matching is very limited, and, thus, the matching is interpreted only as an assumed vulnerability, not a confirmed one. Nevertheless, a thorough CPE matching was recently proposed by Tovarňák et al. [61] and shall be utilized in further development.

5.3. Dashboard

The dashboard [40] is a web application based on Angular[14] that visualizes the content of the database. The dashboard consists of a menu on the left that allows displaying one of the panels providing the content. An example of dashboard content can be seen in the *Network Visualization* panel in Figure 5. The panel allows the user to traverse the graph database freely. A user first searches for a specific host in the network; then the panel displays its node and neighborhood in the graph, i.e., various host relations with other entities, such as software versions, vulnerabilities, and contacts on administrators. The user may then traverse the nodes in the displayed graph and open their neighboring nodes. The detailed information on a selected node is displayed on the right. In the example, we can see a result of search and traversal; the user searched for an IP address and traversed the graphs to get the information on its OS and its assumed vulnerability. Such a user action corresponds to the typical use of the dashboard and the whole CRUSOE toolset; a user looks up a host in the network and immediately retrieves contextual information, which suggests where is the host located, what software and services it runs, and if it is vulnerable or acted in past incidents.

Several more panels were implemented in the dashboard to provide predefined views that derive from frequent queries on the databases, such as an overview of vulnerabilities in the network and an overview and status of the data collection components. The readers interested in the detailed description and kindly referred to the dashboard paper [40] and the toolset implementation [12]. The *Decide/Act* panel is discussed in later sections as it contains a user interface for tools supporting Decide and Act phases of the ODDA loop in the CRUSOE toolset.

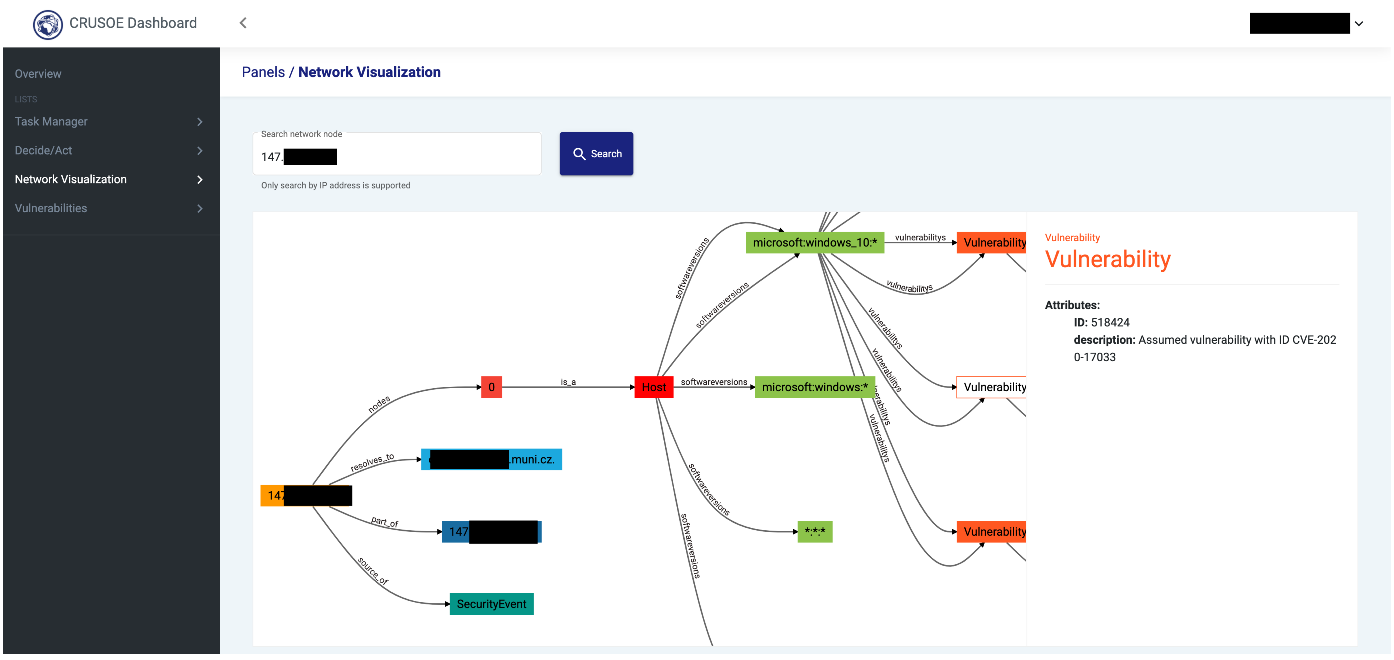
6. Decide

To support the Decide phase, we designed and implemented a decision support system that calculates the potential impact of vulnerability exploitation on the enterprise mission and recommends the most resilient configuration of the underlying infrastructure. In this section, we first present how the enterprise missions (also known as business processes) are mapped to hosts and services in

[12] `https://www.djangoproject.com/`

[13] Common Platform Enumeration, `https://cpe.mitre.org/`

[14] `https://angular.io/`



Panels / Network Visualization

Search network node
147.[redacted]

Only search by IP address is supported

Vulnerability

Vulnerability

Attributes:

ID: 518424

description: Assumed vulnerability with ID CVE-2020-17033

Figure 5: The dashboard envelope and the panel for traversing the graph database.

the network and how the requirements on their confidentiality, integrity, and availability (CIA) are set [41]. Subsequently, we show how the resilience is calculated using the attack graphs, Bayesian networks, and information on vulnerabilities and attacker's position in the network [42]. The description of the design and implementation of the decision support system follows. The section is closed by a commentary on interpreting the outputs.

6.1. Mission Decomposition Model

One of the key building blocks of our proposed approach is an enterprise mission decomposition model based on the constraint satisfaction abstraction logic. A sample mission decomposition model is illustrated in Figure 6. A more detailed description of it, including the necessary theoretical background and an illustrative case study, can be found in our previous work [41].

The mission decomposition model forms a graph with the following entities and relations. An enterprise mission is enabled by a set of *supportive processes*, an abstract representation of critical assets to be protected. The individual supportive processes are enabled by *IT services*, an abstract representation of the components of the supportive processes. The IT service is supported by *cyber components*, particular hosts, and services in the network. Each cyber component is assigned a set of privileges, and the components' interactions are represented as an edge between two components.

We assume multiple alternatives for the mission configuration satisfy the functional requirements expressed in the model via special AND/OR nodes in the graph. The AND/OR nodes are placed between a supportive process and IT services or an IT service and cyber components.

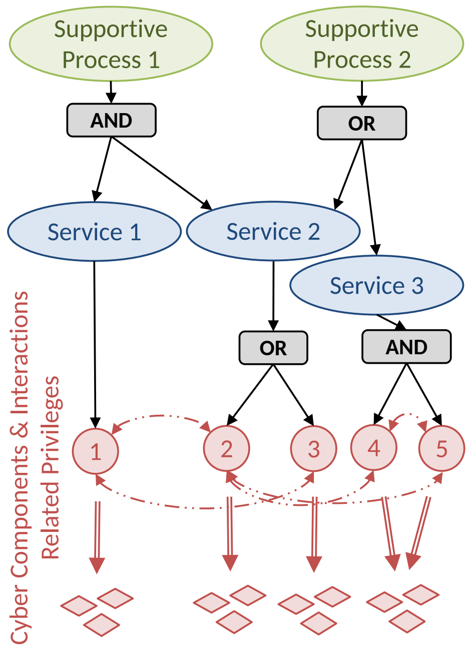


Figure 6: A simple mission decomposition model.

A *mission configuration* is a subgraph in which all the included nodes shall be enabled, and the others shall be disabled. A *satisfying mission configuration* is a configuration that satisfies the functional requirements; an AND-node requires the presence of all its child nodes (IT services or cyber components), and an OR-node requires the presence of at least one of its child nodes.

In terms of security, a mission can be formally described as a system of security requirements that are initially imposed on the individual supportive processes and subsequently on the individual IT services and cyber components. In particular, the model allows mapping the required levels of CIA requirements between the individual supportive process and the relevant cyber components [41].

6.2. Mission Resilience Calculation

The proposed tool recommends the most resilient satisfying mission configuration, and, thus, the resilience of all satisfying configurations is calculated [42]. Assessment of configuration's resilience needs to consider all the vulnerabilities and attacker's position in the network. The vulnerabilities are assessed in the Observe phase. The attacker's position is a set of privileges (assumed to be) held by an adversary. An example of an attacker's position is a compromised user account on a machine that enables the attacker to read data on it.

First, there is a need to find out what can be achieved by an attacker. Attack graphs are a popular tool to enable it. Attack graph depicts all attack paths, i.e., sequences of attacker's actions leading to the state of violated security policy. It was introduced by Phillips and Swiler in 1998 [62] and since then, many types of attack graph tools have appeared [63]. An attack graph can be constructed from the subgraph of a mission decomposition model that includes the cyber components and their privileges and interaction. Using the attack graph, we can find all the paths from the initial attacker's position to the violation of security requirements on a cyber component. The exploitation of a vulnerability of cyber components threatens mission configurations that include such components.

Second, there is a need to calculate probabilities of exploitation and infer their impact on mission configuration. Even though the attack graph depicts possible attack paths, it does not provide information about the probability that the attacker will succeed in violating the security requirements of a mission. Such probability can be obtained using a Bayesian network. The Bayesian network is a probabilistic model that can be represented as a directed acyclic graph with vertices representing random variables and edges representing dependencies and conditional probability distribution for each random variable. Bayesian networks constructed over attack graphs are referred to as Bayesian attack graphs [64]. Constructing a Bayesian attack graph allows for inferring the probability that an attacker will violate the security requirements of the mission while considering dependencies among vulnerability exploits and the attacker's position.

Consequently, configuration's resilience is a probability that the attacker will fail to compromise the security requirements of a mission. The higher the probability is, the more resilient the configuration is. The probability of a complementary event (i.e., attacker will succeed) is computed from probability values of CIA compromise. These values are obtained from the Bayesian attack graph. The possible mission resilience ranges from the least resilient to the most resilient configuration. The mission is as resilient as its currently applied configuration.

6.3. Design and Implementation

The Decide tool was designed and implemented as a decision support system comprising of five modules as depicted in Figure 3. The tool is written in Python and uses MulVAL [65] to generate the attack graphs. Each module implements one step in the calculation of mission resilience. The order of the steps goes as follows:

1. *Components* module extracts all the satisfying configurations of a mission,
2. *Generator* module creates an input file for MulVAL,
3. *Run Mulval* module generates an attack graph,
4. *Bayes* module generates a Bayesian network from the attack graph and infers the probability of violation of mission security requirements,
5. *Process* module selects the most resilient configuration and places it on the output.

Input for the Decide tool consists of a list of devices, including network services and software they provide, an overview of vulnerabilities, the attacker's position (devices and privileges that the attacker can hold), and the mission decomposition model. The data on the network, vulnerabilities, and attacker's position are collected by the Observe tool and stored in the database. The mission decomposition model is serialized into a JSON file and uploaded via REST API to the database.

In the first step, there is a need to identify all the satisfying mission configurations. A mission configuration is represented as a list of cyber components which must be active in to support the mission. The *Components* module loads the description of the mission decomposition model and traverses it from top to bottom to identify satisfying mission configurations. Whenever an AND-node is approached, all of its successors are added to the list. If an OR-node is encountered, an alternate list is considered for each of its successors; only one successor is included in each list. When the graph is traversed, each created list represents one satisfying mission configuration.

In the second step, the input file for MulVAL is generated from the data on the network. The security situation is represented by the facts and rules, which correspond to the syntax of MulVAL [65]. We approach the threat modeling by projecting threat scenarios (attack graphs), and, for this purpose, we specified threat actions as MulVAL rules, i.e., essential attacks anticipated in the process. The main threat action for which we prepared MulVAL predicates is vulnerability exploit. The vulnerability exploits in our model can impact the CIA and cause privilege escalation and code execution. The vulnerability exploit in this scope is a technique used for initial access to the network via public-facing network services or for lateral movement within the network via remote services. Further, we also consider threat action of the valid accounts compromise to log in to services and lateral movement between hosts in the same network subnet. We differentiate the confidentiality, integrity, and availability as goals based on vulnerability's impact [46]. Each vulnerability affects different security requirements to a different extent, while code execution and privilege escalation affect all of the security requirements. Thus, we can take into consideration all three security requirements simultaneously.

As the third step, the *Run Mulval* module executes MulVAL, which then generates an attack graph based on the created input file. In our tool, we iterate over mission's configurations and their components. The attack goals is each attack graph are violations of security requirements posed on a component processed in the current iteration. For example, if confidentiality and integrity of the component are required, there are two final goals in the attack graph. Consequently, attack paths depict ordering of individual vulnerabilities of cyber components exploited by an attacker. The attacker's position is considered in this step. The default attacker's position is the Internet. Therefore, the created attack graphs usually depict generic scenarios where an attacker from the Internet gains access to the network via exploitation of public-facing network services and then moves through the network until the attacker reaches goals.

In the fourth step, the *Bayes* module calculates configuration resilience using the Bayesian attack graph constructed over the attack graph from MulVAL. The generated attack graph represents a threat scenario consisting of dependent threat events partially ordered in time. The generated Bayesian attack graph consists of vertices and edges from the generated attack graph. The probabilities for vertices related to vulnerabilities are derived from exploitability metrics of CVSSv3[15]. Other leaves (i.e., facts from the input file) have assigned probability 1.0. The AND nodes in the output of MulVAL are nodes that correspond to rules for attack graph generation and are assigned conditional probability tables that contain probability 0.8 (taken from MulVAL) for the case when all parents are TRUE, otherwise 0.0. Lastly, the OR nodes in the graph only merge two attack paths. Therefore, all rows of their probability tables contain probability 1.0 except for the case when all of the parents are FALSE. In such a case, the probability is 0.0. The probability inference was implemented using pgmpy [66]. We decided to choose variable elimination as an exact inference algorithm with exponential time complexity in the worst-case. However, if the right ordering of variables is specified, the algorithm has much better performance. Since it is an NP-hard problem to find the best ordering of variables, we used default heuristics in pgmpy called MinFill.

The *Process* module is responsible for determining the final probabilities of CIA violation which are then stored in the database. The final probabilities for a configuration are the worst-case results for one of the cyber components in the context of the whole configuration. It means that other components belonging to the configuration influence final probabilities when they and their vulnerabilities appear in the attack graph.

6.4. Interpretation and Use of the Recommendations

The configuration with the highest value of resilience for each mission is recommended by the Decide tool. The configuration is represented as a list of cyber components belonging to the mission. However, it is not directly applied since it may not be the optimal decision. Reconfiguration of infrastructure is usually beneficial when a critical vulnerability or other severe flaw appears in critical infrastructure. On the contrary, it is not worth reconfiguration if the possible configuration can only slightly improve cybersecurity posture, i.e., when the difference between probability values for current and new configuration is negligible. Quality of service and economic aspects shall be taken into account for the final decision. Changing the configuration, e.g., switching off redundant components, can decrease users' comfort who will have to rely on the remaining devices and services. Changing the configuration frequently also requires employees' time and effort to change their working habits. The final decision is left on a human operator or mission stakeholders, who may consider the reasoning behind the proposed configuration as too weak to justify the reconfiguration.

For these reasons, the Decide tool does not only determine the most resilient configuration but also passes all configurations with their evaluations to the dashboard. The *Decide/Act* panel (see Section 7.2) displays probability values for compromise of security requirements (CIA); the user can choose the proper configuration based on their own expert knowledge of the protected infrastructure and its missions. Selecting an application of configuration triggers a sequence of steps that switch off all cyber components that do not belong to the configuration under the condition that their individual security assessments exceed a certain threshold. Therefore, the algorithm does not block those components with a low security assessment score, such as vulnerability-free cyber components.

7. Act

The Act tool aims to facilitate the execution of mitigation actions in the incident response. The tool follows the

[15]Common Vulnerability Scoring System, `https://www.first.org/cvss/specification-document`

recommendation by the Decide tool, applies mitigation actions via devices in the network, and collects feedback on the measures taken back to the beginning of the OODA loop. A human operator is still necessary during this activity, but the software facilitates the required manual operations, enables automated network reconfiguration, and provides immediate information about the actions taken. It simplifies the work with available network security tools and streamlines the incident response process. The Act tool does not implement any mitigation options but makes effective use of the existing infrastructure.

Nowadays, network security typically relies on devices that monitor and manipulate network traffic to detect, prevent, and mitigate attacks. These devices include, for example, DNS and application firewalls, and tools blocking user access to network resources. Within the CRUSOE toolset, these elements are collectively referred to as Network Security Tools (NST). It should be mentioned that functions of the NSTs might very well be supplemented by the Software-Defined Networking (SDN), as it also provides advanced traffic redirection features [67]. However, SDNs are still too rarely deployed in production environments. Consequently, we focus on environments where this technology is missing.

The NSTs deployed in the network are very heterogeneous, as new ones emerge and the current ones evolve simultaneously with the development of attacker processes and with the discovery of new possible attack vectors. However, this way of gradual development leads to changing and unstable architectures, incompatible implementations, and divergent incident handling workflow processes. Consequently, APIs of different NSTs often support different sets of functions, the return types of these functions are in incompatible formats, and the incident handlers need to know the workflow processes specific to each unique NST.

The Act tool aims at solving two main obstacles standing in the way of incident response automation. The first obstacle is the aforementioned heterogeneity of the NSTs, acquired through their dynamic and continuous development. Any NST API needs to support a set of predefined functions and return its results in a predefined format before integrating it into a single system. Simultaneously, any changes to an NST API will break the tools that use this API, which may cause a chain of failures of NSTs and require major maintenance. We deal with the heterogeneity by deploying API wrappers that use the existing API functions where possible and allow defining new ones where necessary. Consequently, the original API is not changed, and the compatibility is maintained. The second obstacle is the inherent decentralization of NSTs. The NSTs are deployed over a heterogeneous and both logically and physically segmented network administered partly by the central and partly by the local administrators. The Act tool overcomes this obstacle by introducing a central managing element into an otherwise decentralized environment.

7.1. System Design and Implementation

The Act tool is modular and consists of a central logic component (Act Overseer) and five API wrappers as depicted in Figure 3. The Act Overseer provides scheduling and on-demand execution of operations with NSTs. It is a central point that manages all communication between NSTs and the rest of the CRUSOE toolset, thus overcoming the decentralization. The API wrappers provide predefined interfaces for the connected NSTs. Their role is to overcome the heterogeneity of NSTs and ensure that every NST supports all the services required to interact with the CRUSOE toolset.

The Act Overseer is divided into two logical blocks based on the type of operations that it implements. The *Monitoring block* provides services that collect information about the current state of the connected NSTs (e.g., status and free capacity). The *Execution block* implements services that modify the configuration of NSTs and thus directly influence the infrastructure (e.g., by filtering network traffic and restricting user access).

The primary function of the Monitoring block is to keep up-to-date information about the state of the NSTs and to ensure feedback from the NSTs. The block uses the Orchestration service of the Observe tool (see Section 4.2) to periodically run liveness checks of all the connected NSTs. A liveness check verifies the connection with an NST and updates its current availability and free configuration capacity in the central database.

The Execution block manages all configuration changes of NSTs and, thus, is responsible for the application of chosen countermeasures. The activation of this block is initiated by the incident handler who selects a specific network configuration recommended by the Decide tool (see Section 7.2). The Execution block loads the chosen network configuration from the central database and transforms it into separate configurations, i.e., sets of rules compatible with the available NSTs. Subsequently, it compares the newly created NST configurations with the currently active ones, creates a list of necessary changes, and orchestrates NSTs to apply them.

An API wrapper forms a layer between the NSTs and the Act Overseer. This layer is situated on the border between the CRUSOE toolset and the NSTs connected in the network outside of CRUSOE. One of the aims of the CRUSOE toolset is to be compatible with a wide range of environments, which requires the border layer to be specific enough to provide a set of reliable interfaces to NSTs and, at the same time, general enough to support a wide range of these devices. The concept of wrappers is suitable for this situation, as it allows the specification of API frameworks separated from environment-specific NST implementations.

The wrappers define a set of CRUD operations required by the CRUSOE toolset from the NSTs. The wrappers specify the description, input, and output features but do not contain any processing logic that needs to be implemented while binding them to the environment-specific

NST and API. This represents additional costs of system deployment, but it is necessary to abstract from specific implementations. We implemented wrappers for five distinct NSTs that are common in current networks, namely:

- The *DNS firewall* is an NST that can block queries for specific domains. Typically, it is a DNS resolver equipped with a blacklist of known malicious domains. A query for a blacklisted domain is not forwarded nor answered. We implemented and evaluated a DNS firewall based on the DNS Response Policy Zones in the BIND 9 DNS server [16] [48, 49].
- The *Firewall* monitors network traffic, typically at the boundary of the internal network. It enables blocking access from the external network to devices on the internal network based on the IP address of a device. An example of a simple firewall is the basic firewall in Linux – UFW[17].
- The *Mail Filter* allows blocking the forwarding of messages from specific e-mail addresses in the internal network. The function of this NST corresponds to a mail server supporting message filtering (e.g., Sendmail[18]).
- The *Remote Triggered Black Hole* (RTBH) is an active defense feature by Cisco that allows filtering unwanted network traffic at the backbone network [68].
- The *User Blocker* is able to block access of specific users, based on their identifier, to resources located in the internal network. An example of such an access block is locking a user account in the Microsoft Active Directory environment.

7.2. Integration with the dashboard

The *Decide/Act* panel was implemented in the dashboard to serve as an interface to the Decide and Act tools. The panel presents the latest recommendations by the decision support systems and displays mitigation system controls that allow the user to enforce the recommended configuration, e.g., by allowing and disabling the hosts and service at the firewall. The panel consists of five widgets, as shown in Figure 7. For reference, the widgets are framed in red and denoted by numbers that correspond to the numbers in the text.

The first widget is the Devices for Active Network Defense. It displays information about each of the integrated NSTs summarized into two columns. The NST status column shows the current state of the NST – green denotes an OK state, yellow denotes a state with issues that require attention, and red denotes that the NST is unavailable. The used/total capacity column of the widget displays how many configuration rules have already been added to the NST and how many more can yet be added. This widget is designed so that incident handlers can assess the state of the NSTs by a single glance. The second widget sets the value of *Security threshold*, a value considered when a mission configuration is being applied on the NST. Setting the threshold prevents blocking of devices that are not critical for any mission but achieve an average security rating higher than the threshold. The security rating of a device is calculated as the average of its CIA values assigned by the Decide tool. The third widget is the Decide Configurations List. It displays the network configurations recommended by the Decide tool. Details of the configuration are available on-demand by clicking the configuration name. The incident handler chooses a configuration for each mission and applies them on the NSTs all at once by clicking the *Apply selected configs* button. The fourth widget, Act Feedback Log, displays actions taken when the configurations are applied to NSTs so that the incident handler may observe the process in real-time. When the configuration process finishes, the log displays a report summarizing the successfully added NST rules along with any errors that might have occurred. The fifth widget, Missions, visualizes mission configurations. When the incident handler chooses a configuration, the widget colors in red every device that the configuration blocks. This enables the handler to evaluate the impact of the configuration on the network before it is applied on the NSTs.

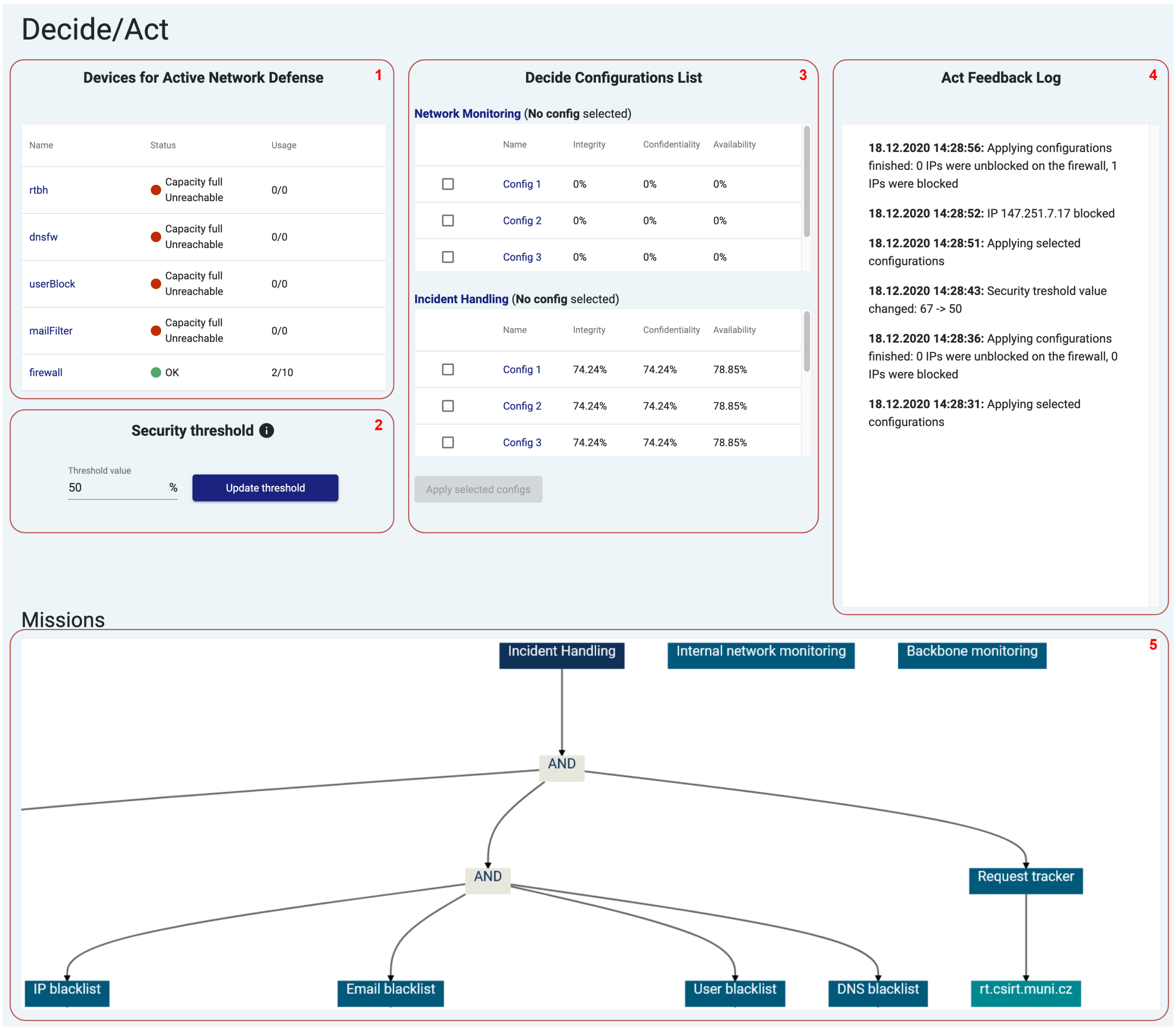
8. Evaluation and Discussion

The CRUSOE toolset [12] was deployed in a live environment to evaluate its performance and feasibility of the suggested goals. Moreover, we conducted an evaluation study with the help of potential users of the toolset. In this section, we first describe the deployment and the characteristics of the environment. The deployment is followed by listing lessons learned from the deployment gathered by the developers. The lessons learned are mostly technical and dealing with data volume and performance. Subsequently, we describe the methodology of the user evaluation and the results, focusing on usability of the toolset, are presented and discussed in the final subsection.

8.1. Deployment Scenario

The toolset was deployed in the network of Masaryk University in collaboration with the university's cybersecurity team[19]. The campus network is large and heterogeneous, consisting of a /16 network segment and serving more than 40,000 users. Roughly 25,000 unique IP addresses from the pool are actively used, around 19,000 unique IP addresses actively communicate each day, and the network traffic volume is around 20 Gb/s. There are

[16] `https://www.isc.org/bind/`
[17] `https://help.ubuntu.com/community/UFW`
[18] `https://linux.die.net/man/8/sendmail.sendmail`
[19] CSIRT-MU, `https://csirt.muni.cz/`



Devices for Active Network Defense

Name	Status	Usage
rtbh	● Capacity full Unreachable	0/0
dnsfw	● Capacity full Unreachable	0/0
userBlock	● Capacity full Unreachable	0/0
mailFilter	● Capacity full Unreachable	0/0
firewall	● OK	2/10

Security threshold

Threshold value: 50 %

Update threshold

Decide Configurations List

Network Monitoring (**No config** selected)

	Name	Integrity	Confidentiality	Availability
☐	Config 1	0%	0%	0%
☐	Config 2	0%	0%	0%
☐	Config 3	0%	0%	0%

Incident Handling (**No config** selected)

	Name	Integrity	Confidentiality	Availability
☐	Config 1	74.24%	74.24%	78.85%
☐	Config 2	74.24%	74.24%	78.85%
☐	Config 3	74.24%	74.24%	78.85%

Apply selected configs

Act Feedback Log

18.12.2020 14:28:56: Applying configurations finished: 0 IPs were unblocked on the firewall, 1 IPs were blocked

18.12.2020 14:28:52: IP 147.251.7.17 blocked

18.12.2020 14:28:51: Applying selected configurations

18.12.2020 14:28:43: Security treshold value changed: 67 -> 50

18.12.2020 14:28:36: Applying configurations finished: 0 IPs were unblocked on the firewall, 0 IPs were blocked

18.12.2020 14:28:31: Applying selected configurations

Figure 7: The Decide/Act dashboard panel containing the widgets for: (1) Devices for Active Network Defense, (2) Security Threshold, (3) Decide Configurations List, (4) Act Feedback Log, and (5) Missions.

numerous servers and network services available in the network; all of them are hardly enumerable. The network is also structured into around 500 network segments, often locally administered. Thus, the team cannot reach most of the hosts in the network and relies on network monitoring infrastructure comprising of several NetFlow probes located on entry points to the network and at the premises of university facilities. The probes are equipped with network-based intrusion and anomaly detection systems. The security team can manipulate network traffic, filter emails, and restrict user privileges in the central identity and access management system.

The toolset was deployed on a primary server (8-core Intel Xeon CPU E5-2680 v3 2.50GHz, 32 GB RAM) and one additional worker (4-core Intel Xeon CPU E5-2680 v2 2.80GHz, 16 GB RAM). The primary server hosted the Observe orchestration service and most of its components, database, dashboard, Decide tools, and Act Overseer service. The additional worker hosted the active network monitoring component so that the network probing could be performed from multiple locations in the network. The passive network monitoring components were connected to the NetFlow monitoring infrastructure, and the NSTs in the network were connected to Act Overseer service via wrappers. The users, namely the incident handlers of CSIRT-MU, were given access to the dashboard.

The university environment is highly challenging to protect against cyberattacks due to its openness, heterogeneity, and the need to process sensitive data in research and administration. Namely, the research in life sciences, in which Masaryk University mainly participates, brings critical requirements for protecting information and research infrastructure. Researchers often work with sensitive data, such as patient data from clinical trials and intellectual property, and often share them with external partners, such as healthcare facilities, industry, and academic institutions abroad. Such an environment poses high requirements for the CIA. Further, the protection of the academic network is complicated by frequent changes of the infrastructure related to the setup and execution of experiments. Large computing infrastructures can be set up temporarily for a short-lasting experiment. Research instruments are often highly unique and unfamiliar to common IT administrators and incident handlers. The instruments are often equipped with legacy IT systems that are vulnerable, which complicates their operations, or are expensive to maintain, which complicates any reconfigurations associated with threat mitigation.

8.2. Lessons Learned

The Observe tools continuously gathered the data on the protected network. Having the information at hand proved to be advantageous compared to requesting the data on-demand. There is no need to interact with a compromised system or to wait for the scans or queries to complete. CRUSOE toolset immediately provides all the information about the machine, its software, vulnerabilities, and role in the network that are sufficient for triage.

The combination of active network scanning and passive monitoring proved to be viable; they complement each other, and the weak spots of one approach are filled in by the other. The active scanning of the whole network in our deployment took up to 16 hours and covered only around 10 % of active devices during one day. Such low performance required the scans to be repeated many times and build the database over a long period of time. The passive monitoring enumerated active devices on the fly and provided basic information immediately. However, the level of detail passive monitoring provided was lower than of the scans. Furthermore, passive approaches face two challenges for future deployment, encrypted traffic and new computation paradigms. The rising volume of encrypted network traffic reduces the visibility into network traffic and complicates fingerprinting. New methods of fingerprinting are arising [45], yet this field is still open for further research. Similarly, the paradigm shift towards data-centric networking and cloud computing complicates asset management. The organization's team loses control over the environment and needs new tools to approach the monitoring and management of the split network.

The mapping of disclosed vulnerabilities to identified assets in the network is another challenge we encountered. A vast majority of CVE records describe the affected software by its vendor, product name, and version. However, in many cases, such identification was not enough to connect the vulnerability to a device with the software in the specific version. Especially for large software like OS, the version itself poses no information about the presence of vulnerability, e.g., vulnerability for Windows 10 cannot be mapped to all Windows 10 machines without the knowledge of installed patches. Readers interested in this topic are kindly referred to a detailed report [47].

The database is a central component of the proposed system, and its performance is crucial for the whole system. Neo4j turned out to be a technologically mature choice with sufficient performance on the hosting machine. The system was under development for more than one year without deleting older entries. Over time, we figured out that if the database contains large volumes of historical data, the periodical updates of data become too slow, reducing the quality of the service. Thus, we added a cleanup component to the *Orchestration Service* that deletes records older than three months which we identified as a sound compromise between performance and the need to check network history. During the testing, the three months of usage in the live environment created 697,783 nodes and 22,119,299 edges with detailed information about 29,535 unique IP addresses in the network [39].

Incident handlers need to orient quickly in the data; the most common procedure to do so is referred to as drill-down/roll-up [51]. The handler first needs to drill-down and get as much information about a specific host as possible to estimate the severity of the incident. The incident handler then needs to roll-up to obtain a bigger picture of the host relations and possibly other impacts on the network. The dashboard allows such operations by clicking through the network visualization panel. The incident handlers could get all information by entering the corresponding IP address without constructing any queries in the system or knowing any specific query language. This leads to faster intelligence gathering and lowers the requirements on handlers' skills and knowledge. The incident handlers may took advantage of the dual monitor setup at their workplace and have RTIR opened on one monitor and the dashboard on the other one. For any IP address or other entity they encounter in RTIR, they query the dashboard and have the contextual information displayed next to the incident information.

To estimate the attack's severity and impact, the incident handler must understand how important the target is for the organization. The toolset automatically estimates the network node's criticality, and the node can appear in a mission decomposition model. Both pieces of information are accessible in the dashboard. Thus, the incident handler can quickly assess if and how the incident is related to the organization's critical infrastructure. Thus, the initial incident assessment and triage can become faster and easier for the incident handlers using the dashboard.

The decision support required mission models that had to be created in collaboration with their administrators and other stakeholders [41]. It took considerable time to create a mission representation, but it saved time to assess protected infrastructures' resilience repeatedly. We modeled CSIRT-MU's own missions, such as *network monitoring* that covers monitoring infrastructure and *incident handling* covering services required for incident response. An example of a model representing a medical system was presented in previous work [41].

Another lesson learned was related to the testing of the Decide tool in the live environment. An organization's cybersecurity posture usually fluctuates between a strong state without security issues and a weaker state with some security issues. Natural changes of cybersecurity posture (such as vulnerability discovery and its patching) are helpful to show that the Decide tool works as expected.

In addition, we observed the overall duration of the computation and the number of processed entities. Although the algorithms should not scale well in the worst case, the practical results show that the computations are feasible for operational needs unless the complexity of an attack graph (i.e., count of vertices and edges) grows too much. In typical cases (e.g., the count of vulnerabilities for a mission does not grow over 60), it takes approximately one minute to compute the results for a mission. In addition, according to Verizon's Data Breach Investigation Report (DBIR 2020), attack paths reconstructed from data about incidents usually have up to six steps [69]. Thus, without any problems, the algorithm assesses the current security situation, considering "usual" incidents.

As a final note, we consider the OODA loop to be a solid choice for decision-making in cybersecurity and incident handling support. However, it was developed for situations in which the user needs to act promptly, which might not always be the case in cybersecurity. Rapid incident response, e.g., in order of minutes from detection to mitigation of a threat, shall definitely benefit from embracing the OODA loop. On the contrary, continuous vulnerability management or threat assessment do not need such rapid iterations and may only benefit from structuring the process. Another issue with the OODA loop is that it lacks the aspects of knowledge management. The incident handler, who iterates through the OODA loop, often encounters repeating or similar situations, for which it might be better to make the decision once and keep the knowledge for later uses. The knowledge can be shared among incident handlers to unify the decision-making among the personnel, especially if it has to conform to policies. A viable alternative would be the MAPE-K model [70] that follows the loop of Monitor-Analyse-Plan-Execute, which is, in essence, very similar to the OODA loop. The major difference is the K, i.e., knowledge continuously built during the iterations. However, informal knowledge management in the form of shared notes, wiki, or a list of exemplary incidents is a common practice among cybersecurity teams; there is not much space for any tool that we would include in our proposed toolset.

8.3. Evaluation Methodology

Measuring the CSA and how does it improve cybersecurity is a challenging issue; it is difficult even to define performance and metrics [8]. Preliminary steps in this area were investigated very recently, e.g., in the work of Rodriguez-Bermejo et al. [35], who propose an evaluation methodology for mission-centric CSA capabilities that covers correct technical implementation, core functionality, and user acceptance. In another example, Happa et al. [15] conducted a user evaluation study of decision support tools for SOC analysts. Staheli et al. [71] surveyed the trends in research on visualization for cybersecurity and found out that usability testing, simulations, and surveys are the most common approaches to the evaluation. However, evaluations of cybersecurity tools are still rare [71], and the evaluations of tools supporting CSA are nearly non-existent [8]. Thus, there is a need to fill this gap.

The CRUSOE toolset was evaluated from the perspective of usability by researchers and practitioners during the experimental deployment. We defined a list of evaluation goals. Specifically, we wanted to know:

- What were the users' needs, if they were familiar with the concepts of CSA and OODA loop, and if they would like to apply them in their work.
- How well does the CRUSOE toolset implement the concepts, and whether it is usable.
- How usable are the particular components of the

toolset and whether the provided data are of sufficient quality.

We created a questionnaire that asks the users about the usability of the toolset and its components. We were inspired by the questionnaire used in the work of Happa et al. [15], who conducted a similar study and pinpointed issues to consider when collecting and evaluating the results. The questionnaire contains 22 questions and is attached as supplementary material to this paper.

We gathered a representative sample of cybersecurity researchers and practitioners of CSIRT-MU. We conducted individual interviews with the participants, in which we asked the participants about the concepts of CSA and OODA loop and explained it if needed. Subsequently, we let the participants interact with the dashboard. The participants were given several simple tasks to help them understand the tool and its usage, namely:

- An IP address was reported to behave anomalously; use the Network Visualization panel to get to know the suspicious host. The participant is asked to comment on vulnerabilities, history of security incidents, and other information, if available, for a randomly queried IP address and how they would perceive the host given the data.
- A vulnerability is a threat to the network. Use the Vulnerabilities panel to find the network segments hosting the most vulnerable machines. The quality of data inputs is discussed.
- Critical infrastructure is vulnerable. Use the Decide/Act panel to find and apply the most resilient configuration of the infrastructure. The procedures and changes to apply are discussed.

After completing the tasks and discussions, the participants are asked to fill the questionnaire.

8.4. Results and Discussion

We conducted the evaluation with 10 participants affiliated with CSIRT-MU, 4 of them are employed as incident handlers and the others are researchers, analysts, and technical staff. We acknowledge that such a number is quite low, but all the participants are familiar with at least the fundamentals of incident handling and provided expert knowledge on cybersecurity operations. The detailed results are displayed in Figure 8.

No participant had any limitations that would affect their interaction with a computer (e.g., color blindness), so we could not collect feedback in this regard. Even though the participants were mostly not familiar with the details of our research and development, all of them were familiar with the concepts of the CSA and the OODA loop prior to participating in the experiment. We assume they learned them from our previous presentations within the university. Prior knowledge of these concepts might be lower in other teams. All the participants were positive about finding such concepts beneficial for their work.

In questions 9-12, we asked about the usability of tools supporting the Observe and Orient phases. The participants found the dashboard user-friendly and the visualizations mostly comprehensive. We received feedback asking for more interactivity of the dashboard, e.g., *"the network visualization should be more interactive. I can get the data from other parts of the CRUSOE toolset, but cannot access them on 'one click'"*. The quality of the data collected by the Observe tools was considered insufficient by a few participants. One participant specifically pointed at OS fingerprinting: *"Host fingerprinting is too general and very likely leads to a large number of false positives."* Indeed, the data quality is a known issue and is hard to achieve when also considering the scalability and performance when processing the data on a large network.

Questions 13-17 were related to the Decide and Act components. We received highly positive feedback on the mission decomposition model. The participants were divided on whether to rely on the recommendations by the decision support tool. This is not necessarily wrong, as one of the participants stated: *"The recommendations are fine, but the final decision should be on the incident handler."* Indeed, we experienced that it is too risky to let the machine decide and act autonomously. The incident handlers are especially cautious when applying any restrictive measures. In the past, they often could not see the impact of their decision to, for example, restrict the network traffic of a host in the network. Any misstep in incident response may harm network operations, reduce users' comfort, and, in the long term, sabotage the reputation of the cybersecurity team. As one participant stated: *"I need to better understand why the config is optimal before accepting it."* This calls for more explainability in the recommendation calculation and presentation in future work. A suggestion was formulated by a participating stating: *"What significantly increases the level of situational awareness and subsequently helps in selecting the optimal mitigation strategy is the ability to see and quantify the risks of technical assets in the context of the security requirements imposed on the mission."* We believe there are is a potential for further research and development in this regard. Simultaneously, the incident handlers may benefit even from simple suggestions, such as a warning when the security state of critical infrastructure jeopardizes enterprise missions. The Act tools received positive feedback with only a few negative comments, such as: *"Ad question 15: The meaning of the numbers is not clear, as each tool has different settings."*

Figure 9 shows the rating of the toolset features. Certain features, such as vulnerability enumeration and the Network Visualization panel, were rated only positively. This suggests that such features are not only beneficial to the users but also technologically mature for application and deployment. All the features were rated positively on average; most of them were rated neutrally at worst. There were several negative ratings or certain fea-

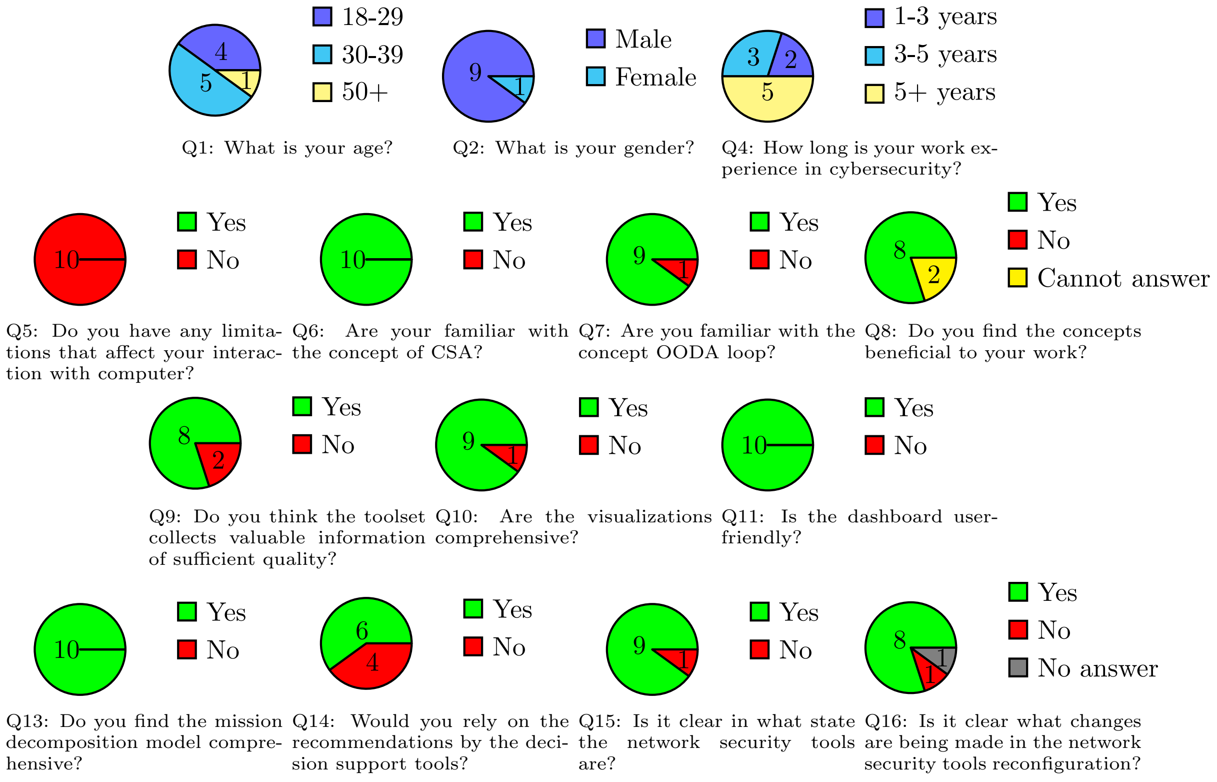


Figure 8: Results of the user evaluation.

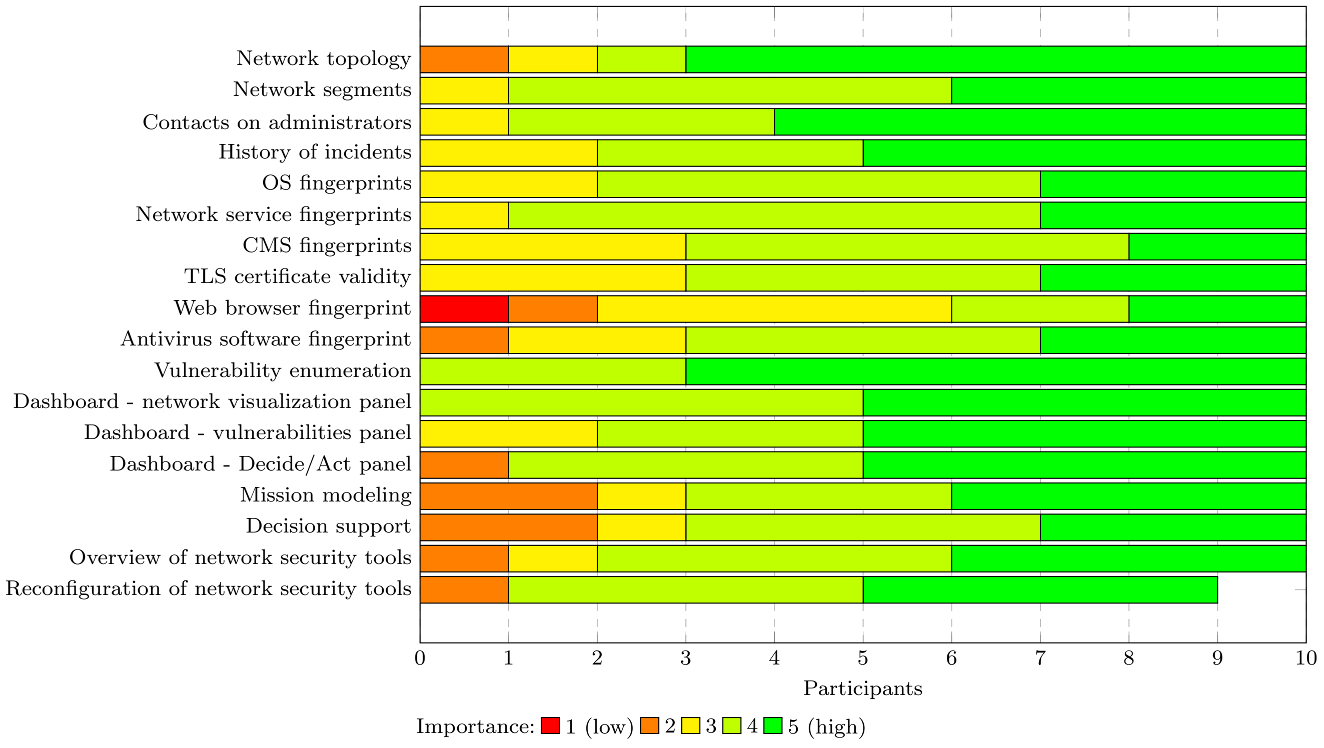


Figure 9: Results of the evaluation of CRUSOE features.

tures, mainly on mission modeling and decision support, which might correspond to the participants varying opinions on whether to rely on the decision support tool or not. Technological immaturity or lower clarity of the recommendations (as discussed above) may also play a role in this case. Nevertheless, decision support raises challenges for future work. Otherwise, there was only feature rated negatively, the web browser fingerprinting. Indeed, such a piece of information is not often needed during incident response. However, other participants rated it positively. It is also worth mentioning that the OS fingerprinting was rated positively despite the critique in the comments. This may indicate that the area of OS fingerprinting deserves further development.

Q19: How well does the CRUSOE toolset implements the OODA loop?

★★★★☆ 4.10

Q20: How well does the CRUSOE toolset improves you CSA?

★★★★☆ 4.00

Q21: Would you like to use it in your work?

★★★★☆ 4.20

Figure 10: User evaluation of the CRUSOE toolset.

The final evaluation of the toolset is generally positive, as can be seen in Figure 10. The participants found the toolset to implement the concepts of CSA and OODA loop well and expressed interest in using it in their work. On the one hand, the overall score of 4.22 in question 21 is promising. On the other hand, it suggests there still are some aspects of the toolset that could be improved. The responses to previous questions point to the issues of data quality in the Observe phase, interlinking the data presented in the Orient phase, explanations of the recommendations in the Decide phase, and improved presentation of the changes made in the Act phase.

9. Conclusion

In this work, we presented CRUSOE [12], a toolset that facilitates CSA, decision-making, and incident response in computer networks. The toolset is inspired by the OODA loop and allows the users to *Observe* the cyber environment, *Orient* in the collected data, *Decide* what actions to take to increase the resilience of the environment, and *Act* to mitigate threats. It collects information on hosts and services in the network via active and passive network monitoring and presents them in a comprehensive manner in a dashboard. The decision support tool allows for selecting the most resilient configuration of the IT infrastructure while satisfying the operational requirements of an organization. Finally, the toolset facilitates threat mitigation via orchestration of active network defense devices.

The toolset was deployed in a live environment and evaluated in collaboration with potential users. A modular approach and the use of existing infrastructures facilitated the deployment. Multiple tools providing similar data complemented each other quite well, as was the case with OS fingerprinting or vulnerability discovery, in which active probing was more detailed, but passive network traffic analysis provided less detailed data on more hosts in the network [47]. The passive network traffic analysis had to overcome issues with the wide adoption of encryption, which seems to be resolved at least in the case of OS fingerprinting [45]. Even though some participants of the user evaluation criticized the quality of such data, they considered it useful.

Decision support can be very helpful but is limited by the need to manually model the critical infrastructure of an organization [41] and the risk of too frequent or costly reconfigurations with negligible effect [42]. The user evaluation further showed that there is also a need to keep humans in the loop to make a final decision, and the recommendations should be explained, which calls for further research and development. The orchestration of active network defense is merely an engineering task that is laborious in a common network but shall be fairly simple with wider adoption of SDN.

We expect the cybersecurity community to embrace further the concepts of CSA and the tools that facilitate it [8]. Therefore, in our future work, we will investigate further the readiness and usability of the proposed toolset and competing tools and evaluate their impact on cybersecurity operations. We expect novel requirements to arise in response to emerging threats, namely on data collection and presentation. An interesting option would be to include identity management systems and connect systems with their users or asset management systems and include the physical location of the devices. Such information may be used to combat threats related to compromised accounts, insider attacks, or ransomware. The decision support opens interesting research challenges, namely in the application of AI. However, trust in the recommendations by the AI might be an issue. Thus, we may more likely see increased semi-automation that keeps humans involved.

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] M. Maj, R. Reijers, D. Stikvoort, Good Practice Guide for Incident Management, `https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management` (12 2010).

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

[3] P. Cichonski, T. Millar, T. Grance, K. Scarfone, Computer security incident handling guide: Recommendations of the national institute of standards and technology, NIST Special Publication 800 (61) (2012) 1–147.

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

[5] M. Husák, T. Jirsík, 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.

[6] S. Jajodia, P. Liu, V. Swarup, C. Wang, Cyber situational awareness, Vol. 14, Springer, 2010.

[7] A. Kott, N. Buchler, K. E. Schaefer, Cyber Defense and Situational Awareness, Vol. 62, Springer, 2014.

[8] R. Gutzwiller, J. Dykstra, B. Payne, Gaps and opportunities in situational awareness for cybersecurity, Digital Threats: Research and Practice 1 (3) (9 2020).

[9] P. Liu, S. Jajodia, C. Wang, Theory and Models for Cyber Situation Awareness, Springer International Publishing, 2017.

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

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

[12] CSIRT-MU, CRUSOE: A Toolset for Cyber Situational Awareness and Decision Support in Incident Handling Inspired by the OODA Loop, `https://github.com/CSIRT-MU/CRUSOE` (2021).

[13] A. Ahmad, J. Hadgkiss, A. Ruighaver, Incident response teams – challenges in supporting the organisational security function, Computers & Security 31 (5) (2012) 643–652.

[14] A. Ahmad, S. B. Maynard, K. C. Desouza, J. Kotsias, M. T. Whitty, R. L. Baskerville, How can organizations develop situation awareness for incident response: A case study of management practice, Computers & Security 101 (2021) 102122.

[15] J. Happa, I. Agrafiotis, M. Helmhout, T. Bashford-Rogers, M. Goldsmith, S. Creese, Assessing a Decision Support Tool for SOC Analysts, Digital Threats: Research and Practice 2 (3) (6 2021).

[16] U. Franke, J. Brynielsson, Cyber situational awareness – A systematic review of the literature, Computers & Security 46 (2014) 18–31.

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

[18] K. Lakkaraju, W. Yurcik, A. J. Lee, NVisionIP: Netflow Visualizations of System State for Security Situational Awareness, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM, 2004, pp. 65–72.

[19] X. Yin, W. Yurcik, M. Treaster, Y. Li, K. Lakkaraju, VisFlowConnect: Netflow Visualizations of Link Relationships for Security Situational Awareness, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM, 2004, pp. 26–34.

[20] R. Berthier, M. Cukier, M. Hiltunen, D. Kormann, G. Vesonder, D. Sheleheda, Nfsight: Netflow-Based Network Awareness Tool, in: Proceedings of the 24th International Conference on Large Installation System Administration, LISA'10, USENIX Association, USA, 2010, pp. 1–8.

[21] H. A. D. E. Kodituwakku, A. Keller, J. Gregor, Insight2: A modular visual analysis platform for network situational awareness in large-scale networks, Electronics 9 (10) (2020).

[22] T. Jirsík, P. Čeleda, Cyber Situation Awareness via IP Flow Monitoring, in: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020.

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

[24] E. Guttman, N. Brownlee, Expectations for Computer Security Incident Response, RFC 2350 (6 1998).

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

[26] J. M. Spring, P. Illari, Review of human decision-making during computer security incident analysis, Digital Threats: Research and Practice 2 (2) (4 2021).

[27] S. Ossenbühl, J. Steinberger, 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.

[28] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, F. Smeraldi, Decision support approaches for cyber security investment, Decision Support Systems 86 (2016) 13–23.

[29] S. Noel, S. Jajodia, Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs, Journal of Network and Systems Management 16 (3) (2008) 259–275.

[30] P. Nespoli, D. Papamartzivanos, F. G. Mármol, G. Kambourakis, Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks, IEEE Communications Surveys Tutorials 20 (2) (2018) 1361–1396.

[31] J. R. Goodall, A. D'Amico, J. K. Kopylec, Camus: automatically mapping cyber assets to missions and users, in: Military Communications Conference, 2009. MILCOM 2009. IEEE, IEEE, 2009, pp. 1–7.

[32] J. Guion, M. Reith, Cyber terrain mission mapping: Tools and methodologies, in: 2017 International Conference on Cyber Conflict (CyCon US), IEEE, 2017, pp. 105–111.

[33] W. Heinbockel, S. Noel, J. Curbo, Mission dependency modeling for cyber situational awareness, in: NATO IST-148 Symposium on Cyber Defence Situation Awareness, 2016, pp. 1–14.

[34] F. R. L. Silva, P. Jacob, Mission-centric risk assessment to improve cyber situational awareness, in: Proceedings of the 13th International Conference on Availability, Reliability and Security, ACM, 2018, p. 56.

[35] D. S. Rodriguez-Bermejo, R. D. Medenou, R. P. de Riquelme, J. M. Vidal, F. Torelli, S. L. Sánchez, Evaluation methodology for mission-centric cyber situational awareness capabilities, in: Proceedings of the 15th International Conference on Availability, Reliability and Security, ARES '20, Association for Computing Machinery, New York, NY, USA, 2020.

[36] J. Webb, A. Ahmad, S. B. Maynard, G. Shanks, A situation awareness model for information security risk management, Computers & Security 44 (2014) 1–15.

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

[38] J. Fritz, Incident Response Methodology: The OODA Loop Explained, `https://cybersecurity.att.com/blogs/security-essentials/incident-response-methodology-the-ooda-loop` (1 2019).

[39] M. Husák, M. Laštovička, 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, ARES 2021, Association for Computing Machinery, New York, NY, USA, 2021.

[40] L. Matta, M. Husák, A dashboard for cyber situational awareness and decision support in network security management, in: IFIP/IEEE International Symposium on Integrated Network Management (IM 2021), 2021.

[41] M. Javorník, J. Komárková, M. Husák, Decision support for mission-centric cyber defence, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES '19, ACM, New York, NY, USA, 2019, pp. 34:1–34:8.

[42] M. Javorník, J. Komárková, L. Sadlek, M. Husák, Decision support for mission-centric network security management, in: 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020), 2020.

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

[44] M. Laštovička, A. Dufka, J. Komárková, Machine Learning Fingerprinting Methods in Cyber Security Domain: Which one to Use?, in: 2018 14th International Wireless Communications

Mobile Computing Conference (IWCMC), 2018, pp. 542–547.
[45] M. Laštovička, S. Špaček, P. Velan, P. Čeleda, Using TLS Fingerprints for OS Identification in Encrypted Traffic, in: 2020 IEEE/IFIP Network Operations and Management Symposium, 2020.
[46] J. Komárková, L. Sadlek, M. Laštovička, Community based platform for vulnerability categorization, in: 2018 IEEE/IFIP Network Operations and Management Symposium, 2018.
[47] M. Laštovička, M. Husák, L. Sadlek, Network Monitoring and Enumerating Vulnerabilities in Large Heterogeneous Networks, in: 2020 IEEE/IFIP Network Operations and Management Symposium, 2020.
[48] S. Špaček, M. Laštovička, M. Horák, T. Plesník, Current Issues of Malicious Domains Blocking, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, 2019, pp. 551–556.
[49] S. Špaček, V. Rusňák, A.-M. Dombajová, DNS firewall data visualization, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, 2019, pp. 743–744.
[50] R. Zager, J. Zager, OODA loops in cyberspace: A new cyber-defense model, Small Wars Journal (10 2017).
[51] T. Moye, R. Sawilla, R. Sullivan, P. Lagadec, Cyber Defence Situational Awareness Demonstration/Request for Information (RFI) from Industry and Government (CO-14068-MNCD2), NCI Agency Acquisition (2015).
[52] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, A. Pras, Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX, IEEE Communications Surveys & Tutorials 16 (4) (2014) 2037–2064.
[53] M. F. Umer, M. Sher, Y. Bi, Flow-based intrusion detection: Techniques and challenges, Computers & Security 70 (2017) 238–254.
[54] G. F. Lyon, Nmap network scanning: The official Nmap project guide to network discovery and security scanning, Insecure. Com LLC (US), 2008.
[55] 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).
[56] J. Smeriga, T. Jirsik, Behavior-aware network segmentation using ip flows, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES '19, Association for Computing Machinery, New York, NY, USA, 2019.
[57] S. Lagraa, 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.
[58] B. Morin, L. Mé, H. Debar, M. Ducassé, M2D2: A formal data model for IDS alert correlation, in: International Workshop on Recent Advances in Intrusion Detection, Springer, 2002, pp. 115–137.
[59] J. Holsopple, S. J. Yang, B. Argauer, Virtual terrain: a security-based representation of a computer network, in: Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, 2008, p. 69730E.
[60] F. Innerhofer-Oberperfler, R. Breu, Using an enterprise architecture for it risk management, in: ISSA, 2006, pp. 1–12.
[61] D. Tovarňák, L. Sadlek, Graph-based cpe matching for identification of vulnerable asset configurations, in: IFIP/IEEE International Symposium on Integrated Network Management (IM 2021), 2021.
[62] C. Phillips, L. P. Swiler, A graph-based system for network-vulnerability analysis, in: Proceedings of the 1998 Workshop on New Security Paradigms, NSPW '98, ACM, New York, NY, USA, 1998, pp. 71–79.
[63] K. Kaynar, A taxonomy for attack graph generation and usage in network security, Journal of Information Security and Applications 29 (2016) 27–56.
[64] N. Poolsappasit, R. Dewri, I. Ray, Dynamic Security Risk Management Using Bayesian Attack Graphs, IEEE Transactions on Dependable and Secure Computing 9 (1) (2012) 61–74.
[65] X. Ou, S. Govindavajhala, A. W. Appel, MulVAL: A Logic-based Network Security Analyzer, in: USENIX Security Symposium, Baltimore, MD, 2005.
[66] A. Ankan, A. Panda, pgmpy: Probabilistic graphical models using python, in: Proceedings of the 14th Python in Science Conference (SCIPY 2015), 2015.
[67] R. Sahay, G. Blanc, Z. Zhang, K. Toumi, H. Debar, Adaptive policy-driven attack mitigation in sdn, in: Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures, XDOMO'17, Association for Computing Machinery, New York, NY, USA, 2017.
[68] Cisco Systems, Remotely triggered black hole filtering - destination based and source based, `https://www.cisco.com/c/dam/en_us/about/security/intelligence/blackhole.pdf` (2005).
[69] G. Bassett, D. Hylender, P. Langlois, A. Pinto, S. Widup, 2020 Data Breach Investigations Report, Tech. rep., Verizon (2020).
[70] J. O. Kephart, D. M. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50.
[71] D. Staheli, T. Yu, R. J. Crouser, S. Damodaran, K. Nam, D. O'Gwynn, S. McKenna, L. Harrison, Visualization evaluation for cyber security: Trends and future directions, in: Proceedings of the Eleventh Workshop on Visualization for Cyber Security, VizSec '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 49–56.