A FRAMEWORK FOR AUTOMATING ENVIRONMENTAL VULNERABILITY ANALYSIS OF NETWORK SERVICES

Dimitris Koutras, Panayiotis Kotzanikolaou, Evangelos Paklatzis, Christos Grigoriadis, Christos Douligeris
Department of Informatics, University of Piraeus, 80 Karaoli & Dimitriou st, Piraeus, Greece

NOTE: Corresponding author: Dimitris Koutras, dkoutras@unipi.gr

Abstract – *The primary objective of this paper is to introduce a comprehensive framework designed to automate the assessment of environmental vulnerability status of communication protocols and networked services, within operational contexts. The proposed algorithm leverages the Common Vulnerability Scoring System version 3 (CVSS 3) metrics in conjunction with network security data. The initial step involves the establishment of a network security ontology, which serves to model the environmental attributes associated with the current security posture of communication protocol channels available within an infrastructure. The process commences with the identification and enumeration of all active communication services through a combination of diverse information gathering tools. Subsequently, active network services undergo assessment using a blend of passive scanning and active security analysis tools, which produce the environmental security score. This score can be integrated into vulnerability scoring systems such as CVSS, facilitating the fine-tuning of base CVSS scores, as well as vulnerability mitigation in real-world environments. To validate the proposed framework, we conducted testing across various networks and communication protocols within a controlled environment, thereby offering tangible illustrations for widely-utilized communication protocols.*

Keywords – Communication protocols, CVSS environmental score, network security ontology

1. INTRODUCTION

It is crucial to collect data regarding the implementation and configuration status of the active communication interfaces when searching for vulnerabilities in networks of linked devices. By collecting and examining this data from a security standpoint, implementation and configuration flaws in protocols can be found early on and fixed. Given that the expectation is that malicious actors will do an inventory of network resources as an initial phase of their cyberattack planning, before initiating attacks on real services or data, network administrators must act quickly to identify and fix these vulnerabilities before they have the potential to become exploitable weaknesses. The process of enumeration, in the context of cybersecurity, involves the systematic probing and identification of network resources, including devices, services, and potential entry points. This reconnaissance phase often precedes a full-scale cyberattack, serving as an essential initial step for threat actors to understand the target environment.

Using network vulnerability scanning tools is one technique to obtain data on the current state of network vulnerabilities. The gathered data can then be used to evaluate how network security environmental data may impact the underlying software vulnerabilities using vulnerability scoring systems(CVSS) [1], like the standard vulnerability scoring system [2]. CVSS computes vulnerability scores using sets of metrics. Vulnerability attributes that remain consistent over time and across various environments and implementations are known as base score metrics. Environmental metrics, which include things like the frequency of a target inside an organization, are implementation and organization-specific vulnerability traits [3].

1.1 Motivation and research questions

Identifying and proactively addressing vulnerabilities that can occur in a network is crucial for several reasons. From the perspective of preventative security, detecting and remediating vulnerabilities early minimises the window of opportunity for malicious actors. By addressing weaknesses in the network's defences before they can be exploited, administrators can strengthen the overall security posture [4]. Correcting vulnerabilities at an early stage reduces the risk of data breaches, system compromises, and other security incidents. This proactive approach helps prevent potential damage to critical systems and sensitive data. Efficiently resolving vulnerabilities can reduce the resources, both in terms of time and effort, required to recover from a cyberattack. It is often less costly to address weaknesses before they are exploited than dealing with the aftermath of a successful attack. Maintaining a secure network environment is essential for upholding the trust of customers, partners, and stakeholders. Breaches resulting from unaddressed vulnerabilities can erode trust and damage an organization's reputation.

To identify the true vulnerabilities of network services and communication protocols in real-world operating settings, a variety of vulnerability scanning and configuration management tools and approaches are available

in the literature. Unfortunately, there aren't many automated methods available that can *automatically express* the vulnerabilities found in a widely-used environmental vulnerability score, such as the environmental score model provided by CVSS [1]. Automating the gathering and evaluation of such environmental-specific data is our primary objective. Within this framework, we pose a number of queries that come up during the primary procedure of gathering and sifting network data. How can we accurately characterize the security statuses of networks through the process of modeling the network environment and systematically gathering relevant input? While there are security ontologies in the literature, such as those found in [2] and [5], they are not quite appropriate for our purposes. Furthermore, for the purpose of acquiring network environmental information, a new ontology must be designed.

Another research question is how to use the network environment and security states specified earlier to automatically modify the base score of the vulnerabilities of network services to produce a more accurate environmental vulnerability score. Since there are various network vulnerability scanning methods available, converting the pertinent vulnerabilities from CVSS base scores to environmental ratings typically requires human intervention, professional guidance, and a lot of time. Even though automated analysis cannot replace experienced security analysis, it could nevertheless be helpful in identifying potentially dangerous environmental network vulnerabilities early on.

1.2 Contribution

This paper presents a proof of concept implementation of a framework designed to automate the environmental score evaluation process for network-exposed services. Our primary objective is to automate the CVSS score for environmental vulnerability, taking into account the interrelationships between the systems being evaluated as well as the unique features of the underlying network topology. We first build a network security ontology in order to accomplish this.

We have created a bespoke tool to automate the collection of the necessary data as specified in our ontology. The tool is created in two bricks. The first brick, functionality, `GORAT` is used to construct a comprehensive map of the network and to discern the interconnections between various devices within the network infrastructure., network components, software, network security policies, and users by automating the enumeration of local networks and systems based on network implants.

The second functionality, `Aness`, takes as input the information produced by `GORAT` and elicits automated network environmental security score production, for all the base score of the vulnerabilities related with each active network service. To achieve this, `Aness` implements *protocol-specific rules*, which aggregate the findings from security testing, both active and passive, associated with each specific communication protocol. We have built the protocol-specific logic for three common application-layer protocols (http, smtp, and ftp) in order to validate the suggested framework.

The integration of GORAT with the API represents a significant new addition to this version of the tool. The principal novelty lies in the algorithm, namely the methodology described herein. Additionally, it's worth noting that this tool is designed for administrative use, employing techniques that mimic those of an attacker during its operation. For each protocol, the tool performs a scan for open ports, mimicking the approach of a potential attacker. If it does not detect any notable vulnerabilities, it cannot proceed further.

1.3 Paper structure

In Section 2 we review related work and in Section 3 we describe the proposed model for automated environmental network security evaluation. In Section 4 we describe how the proposed model was implemented while in Section 5 we validate our model by presenting results on known protocols. Finally, Section 6 concludes this paper.

2. RELATED WORK

2.1 Security ontologies

Numerous viewpoints on ontologies that handle security concepts like risk, threats, vulnerabilities, and security measures may be found in the literature [6]. For example, [7] describes the factors that could lead to a threat in their taxonomy of security threats. However, [8] suggests a classification that includes actors, attacks, and assets. In closer alignment with our methodology, [5] suggests an ontology focused on network security to capture features of network security. A malware-based ontology for a security domain is proposed by [9]. Additionally, they divided the process into concepts according to network possibilities and roles. Furthermore, [10] describe a security domain ontology-based device.

An ontology that aims to link metrics like vulnerability, product, and attacker with the Common Platform Enumeration (CPE) model and CVSS 2 is presented in [11]. In this paper, authors employ the CVSS system to access application layer protocols like xcap and http [12]. In a similar vein, [13] analyzes a system's key network assets using the CVSS model. The [2] model records compatible CVEs by utilizing the link between attacks, devices, and vulnerabilities.

In the literature, there exist several security ontologies that model the information data obtained from network security system logs. For instance, [14] looks at data from Intrusion Detection System (IDS) logs in an attempt

to identify risk indicators and analyze the security risk. Furthermore, the authors [15] employ generic (like location) and security metadata (like certifications and access control measures) for their ontology data, in addition to timestamps and IDS logs. Lastly, additional approaches concentrate on data collection for cyber-physical systems [16].

2.2 Network mapping/scanning methodologies

An additional helpful source of information for populating a network security ontology is network mapping and scanning technologies. The primary information source will be the network itself. Attempting to map the systems and network services inside the evaluated perimeter and define the network's scope is the first stage. There are various kinds of scans, including open, partly open, stealth, sweep, and miscellaneous, according to [17]. Thus, the writers categorize cyber-scanning by outlining its traits, strategies, and techniques. Furthermore, [18] offer an approach for mapping Internet of Things networks. Moreover, scanning can be done at the device level, for example, using the syn and ack signals [19]. Several writers have utilized Nmap and other similar tools for host and port scanning [20], [21].

We also employ tools such as Nmap, not for the purpose of extracting vulnerabilities, but rather to detect and monitor potential incidents within the communication interactions between various systems and devices utilizing protocols from the 802.11 family [22]. Balsam et al. [23] conducted a study using the National Vulnerability Database, a public vulnerability database, to correlate the data provided by the fundamental and temporal components of the CVSS 3.x vector. They also use Python scripts to generate histograms based on data extracted from the databases. The outcomes of this analysis reveal an over-representation of certain numerical values (low(1), high(3) etc.) in the base scores calculated using CVSS v3.x when juxtaposed with the corresponding count of available exploits.

On the other hand, Vasilyev et al. [24] use CVSS metrics by putting the corresponding assets they have captured within a cyberattack vector model for specific systems. We note that they do not extract the score (via variables) as in our case, and they just simply use it. Walkowski et al. [25] clearly focus on a study of the metrics that can affect the base score over the past of time. They study also how some instances can strongly affect the base score. An interesting fact is that Amankwah et al. [26] maintain that the effectiveness of the CVSS metric, however, has been questioned in earlier studies, giving rise to a variety of vulnerability score metrics.

To identify vulnerabilities in web applications with heightened susceptibility, various studies, including those referenced in Sang et al. [27], advocate for the use of automated frameworks. These frameworks are specifically designed to assess the severity of vulnerabilities detected by open-source web scanners. A key aspect of these studies is the adoption of the OWASP 2017 Top Ten as a benchmark, which provides a standardized metric and prioritization scheme for evaluating the severity of identified vulnerabilities.

To uncover a diverse range of security vulnerabilities within these vulnerable datasets, data from the National Vulnerability Database (NVD) by NIST and the Software Assurance Reference Dataset (SARD) were employed as foundational learning materials. This approach underscores the framework's focus on detecting vulnerabilities directly within software code, as opposed to those found in services and protocols. By leveraging these comprehensive datasets, the framework enhances its capability to pinpoint and analyze software vulnerabilities in a more targeted and effective manner.

2.2.1 Comparison with our preliminary work

The primary limitation of the previous framework, as detailed in Koutras et al. [28], did not stem from its algorithmic design but rather from its operational efficiency and the technological stack employed from an architectural perspective. To enhance the performance in the updated version, we initially adopted the Go programming language, known for its efficiency and speed. Subsequently, we integrated an API to facilitate the connection between the legacy application and the NIST API, ensuring seamless data exchange and interoperability. Further augmenting our system's efficiency, we developed a more advanced and rapid client-server communication application using Go. It is important to note that while making these enhancements, we have retained the core logic of the initial framework, ensuring continuity in functionality and design philosophy.

2.3 Vulnerability detection tools

In this subsection, we delve into an analysis of existing research and advancements in the field of vulnerability detection. In [29], the authors delve into contemporary application-independent slow Denial of Service (DoS) attacks. They introduce two new attack models, Slowcomm and Slow Next, along with a simulation tool for attack execution. The tool was utilized to assess the vulnerability of various Internet services, including HTTP, FTP, and SSH servers. The paper also outlines specific attack signatures and detection strategies. In [30], the authors focus on the the identification of the application layer protocol, with a task essential for enabling protocol-specific deep packet inspection. The research acknowledges the complexity of this task, noting that traditional methods like relying on port numbers are insufficient for accurate protocol identification. The study conducts a detailed analysis of Dynamic Protocol Detection mechanisms used in popu-

lar open-source network monitoring tools, using HTTP as a case study. Van-Thuan et al. in [31] present AFLNET, a graybox fuzzer designed for testing protocol implementations. It adopts a mutational approach and leverages state-feedback to enhance the fuzzing process. It starts with a corpus of actual message exchanges between a server and a client. AFLNET functions as a client, replaying and modifying the original message sequence sent to the server, focusing on variations that improve code or state space coverage. It identifies the server states affected by a message sequence using the server's response codes. This feedback allows AFLNET to pinpoint and target progressive regions within the state space.

2.4 Vulnerability databases

Various vulnerability databases exist in the literature, that maintain information about vulnerabilities, including details about the affected systems, severity ratings, and recommended mitigation or patch information. Some of the most widely-used vulnerability databases and sources include:

- The National Vulnerability Database (NVD)[1] is maintained by the National Institute of Standards and Technology (NIST) in the United States. It is a comprehensive and authoritative source of information about security vulnerabilities. NVD provides detailed vulnerability descriptions, Common Vulnerability Scoring System (CVSS) scores and references to related security advisories and patches.

- Common Vulnerabilities and Exposures (CVE)[2], functions as a registry of standardized names, known as CVE identifiers, for publicly-disclosed cybersecurity vulnerabilities. While it doesn't offer in-depth details about these vulnerabilities, its primary role is to establish a consistent way of identifying and referencing them. More comprehensive databases typically connect CVE IDs to detailed vulnerability information.

- The Common Weakness Enumeration (CWE)[3] is a collaborative effort to compile a catalog of weaknesses found in software and hardware. It serves as a universal language for recognizing, addressing, and averting vulnerabilities. While it doesn't function as a direct repository of vulnerabilities, it aids in comprehending the sorts of flaws that can ultimately result in vulnerabilities.

- Exploit Database (Exploit-DB): Exploit-DB is a widely-used resource for security professionals and researchers. It contains a collection of exploits and vulnerabilities. While it's not a comprehensive database of all vulnerabilities, it can be useful for researching known exploits.

- The OSV[4] schema provides a human and machine-readable data format to describe vulnerabilities in a way that precisely maps to open-source package versions or commit hashes.

- CVE Details[5] is a website that provides detailed information on vulnerabilities, including CVE IDs, severity, affected products, and related references.

- GitHub's Security Advisories database[6] includes information on vulnerabilities in open-source software hosted on GitHub. It provides details and links to relevant security patches and advisories.

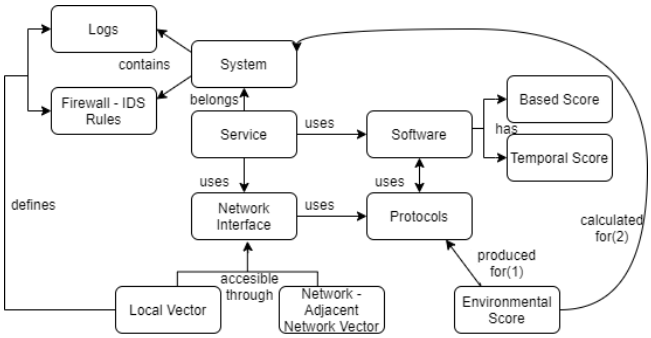


Fig. 1 – Network security ontology

3. SYSTEM ARCHITECTURE

Initially, we introduce the suggested security ontology, which attempts to define the data gathering associated with network security level. Next, we will introduce the suggested algorithm for automating the evaluation of active network connections' environmental security.

3.1 Network security ontology

Determining which data is pertinent to the security level of network services, ports, and communication channels is a necessary step in designing an automated framework for evaluating the environmental network security levels [9, 6].

The goal of the proposed ontology, is to identify and correlate data that is related to the environmental vulnerability level of network services. This is achieved via the definition of abstract entities that affect the level of vulnerability of network services, along with their relationships. Fig. 1 illustrates the proposed network security ontology. More specifically, the *system* is the core entity around which all others are grouped. According to our ontology, the system entity is related to entities that contain information about the system's security state, such as

[1] https://www.nist.gov/programs-projects/national-vulnerability-database-nvd
[2] https://cve.mitre.org/
[3] https://cwe.mitre.org
[4] https://osv.dev/
[5] https://www.cvedetails.com/
[6] https://github.com/github/advisory-database

system logs and relevant *firewall/IDS rules*, which in turn are information accessible via local access, and therefore are related to the *local (attack) vector*. The system also contains *services*, which in turn consist of *software* and *network interfaces*. The network interface is used to instantiate a network *protocol* and is accessible by the local and adjacent network vectors. As far as the *environmental score* is concerned, it is influenced by the protocol entity and all the entities connected to it, in order to ultimately evaluate the environmental vulnerability score of the core entity under examinations, i.e. the system entity.

In our model system, a computing system with a unique network address that operates within a network and provides various network services is described as an entity. To provide services to remote users and systems, each network service may employ a network interface, which in turn implements a network protocol. Consider an ftp server (the system) with a unique IP address that offers file sharing (the service) from a specific network interface (port) by implementing the ftp protocol. Our ontology will represent all network connections as vectors of the form [system,service,IP,port,protocol] and will assign each network link a network environmental score.

Since software must be installed and operated on computers in order for network services and protocol implementations to function, implementation vulnerabilities are crucial data to gather when assessing the security of network connections. Vulnerabilities are classified into three primary categories according to the CVSS model [1, 12] : base score, temporal score, and environmental score. The software used to provide network services and protocols will process both the base and the temporal score. For every network connection that is active on a system, the environmental score will be specified as follows.

The accessibility information is defined for each potential access vector to a network interface since network services can be accessed via local access, neighboring network access, or remote network access. Each vector collects data on an interface's accessibility, whether locally or remotely. This includes the entity (process, user, IP, etc.) that is gaining access to the network interface, the time of access, and other associated details. Such data is derived from information about network security devices, including firewalls and intrusion detection systems, that specify access rules.

3.2 Environmental network security assessment

The automated environmental network security assessment is described in Algorithm 1. The proposed algorithm is implemented through concrete functions, while the correlation of the information is based on the relations of the network security ontology presented in Fig. 1. All of the links, definitions, and resources listed in the ontology are used by the algorithm. The research questions that are addressed by the presented algorithm execution are the following:

Algorithm 1: Produces environmental score for protocols and assets per system, per network from $\{\mathcal{T}\}$.

Input : $\mathcal{T}$=Target ip range. $\mathcal{P}$=Available protocols.
Output: $assessedNetInterface[]$ = A set of lists containing the environmental score of the subnetwork.

Algorithm `EnvScoreProduction()`
  $s[] \leftarrow$ **NetworkScanning**$(\mathcal{T})$
    `/* s[] is the list of vectors s[i]=(system,service,IP,port,protocol) representing network connections. */`
  $FI[] \leftarrow$ **NetworkRulesEnumeration**$(s[])$
    `// FI[] is the exported firewall-IDS rule-set from the network.`
  $i \leftarrow 1$
  while $s[i] \leftarrow$ ***hasNext***$(s[])$
  do
    $SV[i] \leftarrow$ **NetworkConnectionsEnummeration**$(s[i])$
      `// SV[] lists network software running in each examined system.`
    $initial\_security\_state[i] \leftarrow$ **PassiveSecurityTesting**$(s[i], SV[i], FI, \mathcal{P})$
      `// Security info retrieved through protocol-specific passive scanning tools.`
    $updated\_security\_state[i] \leftarrow$ **ActiveSecurityTesting**$(initial\_security\_state[i], s[i], SV[i], FI, \mathcal{P})$
      `// Active and protocol-specific security testing.`
    $assessedInterface[i] \leftarrow$ **EnvScoreProduction**$(initial\_security\_state[i], updated\_security\_state[i])$
      `// Evaluation of environmental security score per network interface.`
  end
  return $assessedInterface[i]$

- Q1: What data are we using as input to define the scope of the analysis?
- Q2: Which data relations are important to identify during the process?
- Q3: In which way can the security testing affect the resulting output?

We must first define the network that is being studied in order to respond to the aforementioned queries. Furthermore, for every unique network service interface, we must specify the relationships between hardware, software, and services that carry out a network protocol (i.e. define vectors of the form `[system,service,IP,port,protocol]`). Next, we must do both passive security scans and active security tests to

determine the environmental security level of each network service that has been identified.

The functionality of the suggested algorithm is organized around six different functions, which fall into two primary categories. The first stage, which is carried out by the first three functions, is protocol agnostic and consists mostly of creating the list of network interfaces designated for assessment.

3.2.1 Phase 1. Enumeration of the examined network services

The following phase, which is driven by the final three functions, adopts a character peculiar to the protocol. During this stage, every function modifies its functioning according to the recognized protocols, thereby carrying out security testing—both passive and active—that is customized for the particular protocol under consideration. These tasks ultimately result in the evaluation of the highest level of environmental security for every single network interface.

Based on the terms defined in our ontology (Fig. 1), Algorithm 1 analyzes the *systems* located in the network to produce *services*, and link these services provided by network interfaces with the *software* that they use.

The following functions are utilized:

- *NetworkScanning* - The group of the network addresses of the network under examination, as well as details about the subnetworks, are entered into this function. In order to be handled later by the other functions, it returns a list of network vectors with a specified structure (`[system,service,IP,port,protocol]`).
- *NetworkRulesEnumeration* - The primary objective of the secondary function is to retrieve security data about the network (such as firewall and intrusion detection system rules) from a local standpoint. We'll be using a proprietary instance of our tool, GORAT (covered in Section 4.1), to collect data locally and set up network implants on different systems. Our current objective is to locate firewall and/or IDS rules associated with the analyzed network connections, while more intricate analysis, including rules-file parsing, may be specified in the future.
- *NetworkConnectionsEnumeration* - This function's main goal is to produce thorough software-related information. It is driven by inputs such as the list of supported protocols and the current inventory of network vectors. As per the requirements specified in our ontology, this function is intended to methodically list important software details, such as version information, that are specific to the software that is being used in relation to the pertinent protocols and services.

3.2.2 Phase 2. Environmental security assessment of network services

As previously said, each protocol under examination calls for a particular implementation during this step. First passive security information collecting is carried out for each supported protocol in order to establish an initial security state. After that, active security testing is used to update the initial security state based on validated assaults, depending on a list of supported active attacks. Naturally, depending on the particular context, it's likely that some attacks and vulnerabilities won't be verified, for instance because of network or system hardening.

In our algorithm $\mathcal{P}$ denotes the list of supported protocols.

- *PassiveSecurityTesting* - Depending on the particular protocol used by a network interface or service under investigation, this function consists of a combination of passive information security gathering tools (i.e. for each vector `[system,service,IP,port,protocol]`). The function handles the network service's initial security state. Observe that our study groups vulnerabilities at the network connection level rather than the device level, in contrast to previous studies like [32] and [33].
- *ActiveSecurityTesting* - This function is intended to carry out a wide range of automated active security tests, each specifically customized for the protocol that is being examined. Its goal is to produce a result that is superior to the basic score in terms of security posture refinement and enhancement. This is accomplished by carefully evaluating vulnerabilities in order to confirm or refute any presumptions about security flaws in the network environment. As a result, the output produced by this function goes beyond the first hypotheses and provides a more accurate and up-to-date picture of the security situation. Compared to the base score, the updated and more correct security state is the outcome, since the presumed vulnerabilities might be verified or rejected.
- *EnvScoreProduction* - Finally, this function takes as input the *initial_security_state* and the *updated_security_state* which are the results of the two previous functions respectively. Its scope is the environmental score production according to the CVSS model. We established a few security guidelines, primarily based on NIST publications [34, 35, 36]. The types of attacks that are investigated in each network service protocol serve as filters to evaluate the security data in order to compute the environmental score.

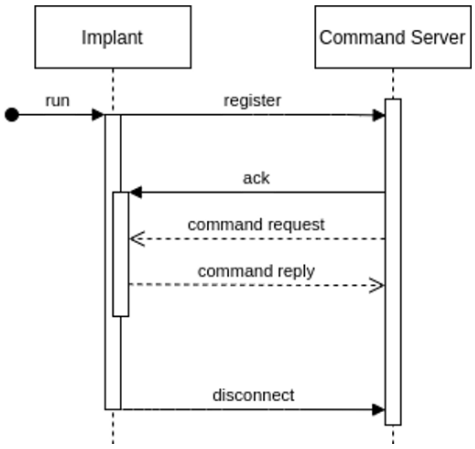


Fig. 2 – Client-Server architecture

4. IMPLEMENTATION

We have designed a proof of concept implementation of the proposed framework presented in Algorithm 1. The framework that has been created consists of two different deployments. These instantiations are directly linked to each other in order to compose the basic framework. This custom tool-framework was developed in GoLang in order to implement the two phases described in Section 3.2.

4.1 Implementing network and system enumeration

GORAT, developed in Go, implements the first phase and the relevant functions NetworkScanning(), NetworkRulesEnumeration() and the NetworkConnectionsEnummeration() from the local vector scope. It follows the Client-Server network architecture (see Fig. 2).

In GORAT, each node is equipped with an implant that can establish a connection with the control server. This control server has the capability to instruct these implants to perform specific tasks and gather responses from them. For each unique IP and port combination, the control server can identify the software running on the system, allowing it to compile a detailed list of network interfaces and services. This information is structured as a vector with the following format: [system, service, IP, port, protocol], as illustrated in Fig. 2.

Communication between the control center and the implants is facilitated through gRPC channels. When an implant initiates a gRPC channel, it essentially establishes an HTTP/2 connection to the control server. This channel can then be reused for sending multiple remote commands to the server.

GORAT offers three key functionalities:

- OS command execution: It allows the execution of operating system commands.

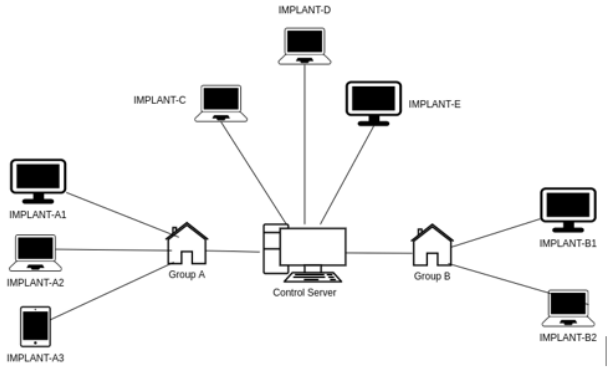


Fig. 3 – Multiple implants architecture

- Concurrent control: GORAT can concurrently manage multiple implants (see Fig. 3).
- Information gathering: It collects information through shell commands and operating system queries.

Each implant generates a unique identifier to register with the control server. During the connection setup, the implant shares user and device system information with the control server. The user information includes the username, user ID, home directory, and user shell, while device information comprises the hostname, operating system, kernel version, architecture, IP address, and more.

4.2 Implementing protocol security assessment

Automated Network Environmental Security Scoring (`Aness`)[7] implements the second phase of the proposed framework, and the relevant functions of *PassiveSecurityTesting*(), *ActiveSecurityTesting*() and *EnvScoreProduction*().

Our proof-of-concept solution presently supports the automated environmental security evaluation for three widely-used application layer protocols: http/https, ftp, and smtp. This is because the second step is protocol-specific. A comprehensive list of communication protocols from all layers might be included in the list[37].

Aness evaluates the environmental vulnerability level of all software associated with a network service under investigation using a combination of open-source security tools and its own unique tools. It is mostly written in Go and depends on bash scripting for interaction. Aness's general functionality is seen in Fig. 4.

The vectors of the network services identified in the previous phase, in the format `[system,service,IP,port,protocol]`, are inputted. Using Nmap and a bespoke port scanner written in Go, it first conducts a thorough enumeration for each

[7] https://github.com/vpaklatzis/Aness-go

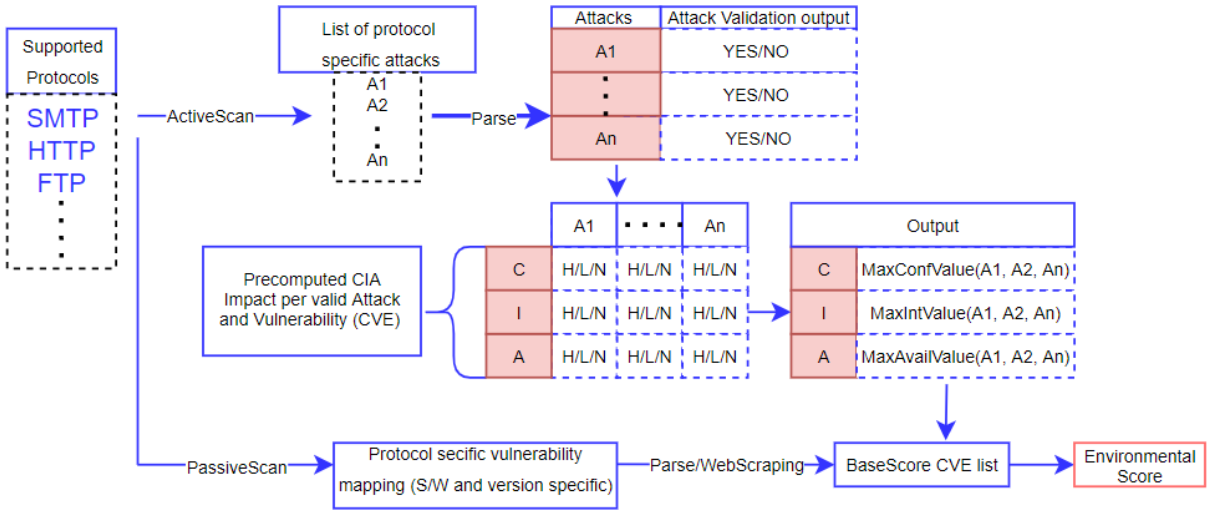


Fig. 4 – Aness architecture and workflow

network service that has been detected. The result of this operation is a structured text file that is parsed to produce the name and software version that has to be tested.

Using the name and version generated by the thorough enumeration phase as parameters, the tool does an automated online search at the NIST National Vulnerability Database [8] for the passive scan. The program makes use of the NIST's official API. A list of all the CVEs linked to the network service and protocol under investigation is the output. Every CVE immediately extracts the base score CVSS vector from the online NVD database and saves it in a local list.

Then, according to the tested communication protocol, it triggers various open-source *active* security analysis tools to test for known security attacks (e.g. for https Nmap and sslyze are used among others). The result of the active security tests is a matrix having as an output a boolean value indicating the success or failure of each examined attack (i.e, the attack validation table shown in Fig. 4).

The program uses a set of dictionaries for each security attack to process the results of the tests. These dictionaries explain how each attack impacts the environmental CVE score of each vulnerability in a piece of software that is used by the particular protocol and network service that is being examined. This is the tool's main, protocol-specific component. The dictionaries for each supported protocol have been created by reviewing each assault and documenting its impact. The environmental attack vector attribute of the applicable CVE is set to 'L' (local) if, for instance, an attack initiated from a remote network fails but succeeds when executed from a local network. In the same way, if a DoS attack can be successfully executed then the availability impact is set to 'H' (high). In case where more than one attacks are available, the final output for each metric is the maximum (worst-case) result verified in any of the attacks (4).

The impact mentioned in this methodology relates to how the program assesses and adjusts the environmental score of Common Vulnerabilities and Exposures (CVEs) based on the outcome of security attacks tested against specific software protocols and network services. This impact is manifested in several key areas:

- Adjustment of CVE scores based on attack context: The program modifies the environmental score of each CVE in a targeted piece of software. This score is adjusted based on how each type of security attack affects the software when used with specific protocols and network services.
- Protocol-specific assessment: The tool employs protocol-specific dictionaries, which are tailored to evaluate the impact of various attacks on different network protocols. This specificity allows for a more accurate and relevant assessment of vulnerabilities in diverse network environments.
- Dynamic assessment of attack impact: The environmental attack vector attribute of a CVE is dynamically altered based on the attack's success or failure in different scenarios. For example, if an attack fails remotely but succeeds locally, the environmental attack vector is set to 'L' (local). This dynamic assessment provides a nuanced understanding of the vulnerability's context and potential exploitation methods.
- Evaluation of different types of attacks: The tool can assess a variety of attack types, such as Denial of Service (DoS) attacks. If a DoS attack is successful, the

[8] https://nvd.nist.gov/

availability impact of the CVE is set to 'H' (high), indicating a significant risk to the system's availability.

- Worst-case scenario analysis: In situations where multiple attacks are possible, the program determines the final output for each metric by considering the maximum (or worst-case) result verified in any of the attacks. This approach ensures that the assessment reflects the highest potential risk associated with each CVE.

Below we describe the logic implemented for each of the currently supported protocols, while Table 1 summarizes the effect on the impact metrics assigned to each attack. The values presented in the table are based on the information that malicious actors may reveal, in relation to the documented vulnerabilities of the system and the examined attacks for each protocol.

Finally, the environmental score and severity are calculated by manipulating the CVE vectors of the online CVSS calculator [9] and scraping the data through the API. The result is the the official environmental score of the examined system software assets. Also it is worth mentioning that the tool can support all the current versions of the CVSS metrics (version 3.x, version 2).

Table 1 – CIA vector metrics per threat

	SMTP			FTP			HTTP		
	StrangePort	OpenRelay	Enum	Anon	Bounce	FireWall	No/Weak TLS version	Invalid Cert Chain	RefererChecker
C	H	N	L	L	L	N	H	H	L
I	H	H	N	L	L	H	H	H	L
A	L	H	N	L	L	H	H	L	N

4.2.1 Examined smtp attacks

The vulnerability against known smtp attacks such as strange smtp port, open-relay [38] and smtp user enumeration is automatically tested.Every attack's effect is correlated with the possible harm it could do. Since a successful weird port attack suggests that there may be a backdoor SMTP service operating on top of the official service, confidentiality and integrity are likely to be severely impacted.

Since it permits unauthorized usage of the service, a successful open relay attack will have a significant effect on availability and integrity. It may also be utilized in conjunction with denial of service attacks. Lastly, ftp user enumeration is thought to be the least successful method because it does not result in direct exploitation, which diminishes its impact.

4.2.2 Examined ftp attacks

For the ftp connections, `Aness` automatically tests known ftp attacks, including the bounce attack, firewall bypassing (a vulnerability in netfilter and other firewalls that use helpers to dynamically open ports for protocols such as ftp and sip) and the ftp anonymous login [39].

[9] https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

Once more, the tested attacks are mapped to each attack's possible impact. For instance, because they need extra steps to be successful, the anonymous FTP login and bounce attacks are given a low influence on all impact metrics. A successful firewall bypass assault, however, will have a significant effect. However, keep in mind that this attack needs access to a neighboring network.

4.2.3 Examined http attacks

We verify the existence and version of the underlying TLS protocol, the application of known attacks, and the server certificate status (algorithm strength, certificate chain validation, verification of other known http attacks like referer chacking etc.) for http interfaces. Numerous open-source programs are called, including sslyze, Nmap, and sslscan.

4.3 Aness tool architecture

The 'Aness API' is used as a functionality inside the tool, together with the GORAT as some can see in the Fig. (5). Within the architecture, GORAT is seamlessly integrated into the framework and works in conjunction with the API known as 'Annes API'. Annes API acts as the outer layer that communicates with external components, while GORAT operates within this framework, providing critical functionality.

Annes works as a standard HTTP/1.1 API, developed with GoLang and further optimized with the powerful Gin middleware. This combination allows Annes to act as a concurrent web server, capable of handling multiple client requests simultaneously.

What really makes this system unique is GORAT. GORAT works within Annes and greatly enhances its capabilities. GORAT is designed to work with goroutines, which are essentially lightweight, independent threads that execute tasks concurrently. These goroutines are highly efficient and resource-friendly, making them far superior to traditional OS threads used in languages such as C#, Java, Python and others.

The collaboration between Annes and GORAT ensures that the system can efficiently manage concurrent tasks and client requests while maintaining remarkable speed and resource efficiency. GORAT handles the complex tasks behind the scenes, using the power of goroutines, while Annes manages the interactions with external elements.

In essence, Annes acts as the interface and communication layer, while GORAT's goroutines provide the underlying power and efficiency that make this architecture exceptional. Together they create a robust and powerful system capable of seamlessly serving multiple applications.

In summary, the system represents a significant leap in

efficiency compared to its predecessor [28]. It boasts a remarkable speed increase, performing tasks 2 to 3 times faster. For instance, what used to take 4 to 5 minutes now only requires 1 to 2 minutes to complete. Furthermore, it has proven to be considerably more reliable, consistently delivering accurate and dependable results.

5. VALIDATION

In order to validate our framework we followed two different approaches. First, we created a fully controlled custom environment, as shown in Fig. 6. The main purpose is to check that the internal processes contained in the source code of our tool work as expected. Since the weaknesses of these services are already known in advance, we have the advantage of preemptive insight, allowing us to strategically address and mitigate these vulnerabilities more effectively. Secondly, we run the `Aness` tool on a known topology, since it is an administrative tool, but on a server with unknown services. The goal here is to test the effectiveness of the tool in a real environment.

5.1 Validation through customized tested topology

To rigorously validate and assess the efficacy of our proposed framework, a structured approach was undertaken, necessitating the establishment of a dedicated test network environment. This controlled testbed served as a controlled, representative model of real-world operational settings, enabling the evaluation of our framework's capabilities under simulated conditions. As shown in Fig. 6 it involves typical services, including the network communication protocols supported by `Aness`, namely smtp, ftp and http.

Initially `Aness` is installed with implants (GORAT functionality) running on each network (DMZ and internal network) in order to enumerate the network services. For each enumerated network service the output is given to `Aness`, to examine which services are supported for automated security testing.

Within this meticulously crafted test network, we conducted a series of systematic tests and assessments. Central to this validation process was the deployment and utilization of our proprietary toolset within a randomly selected system segment of the network. This approach ensured an objective and unbiased evaluation, allowing us to gauge the framework's performance, scalability, and its capacity to identify vulnerabilities and enhance security in a dynamic, unpredictable context. It is imperative to acknowledge that the Common Vulnerabilities (CVs) detected by the tool within each protocol scenario surpass the values displayed in the tables presented in tables 4, 3, 2. To elucidate the variability in environmental scores and base scores, a representative sample is meticulously chosen for analysis.

For smtp we compare two different setups: a postfix server with the default configuration, and a second instance running on a non-typical port. We also check the domain version. As shown in Table 2, `Aness` can effectively identify the effect of the environmental changes wrt the environmental score of the CVEs of the relevant service. For example, a vulnerability that may be abused to cause a denial of service (e.g. CVE-2021-35525) would be more exploitable, if the system configuration allows unauthenticated service use, as in the case of open relay.

For the ftp service we set up the Proftpd 1.3.7 ftp server in the default configuration. We present the exploitability of this service from two different attack vectors, i.e. testing the effectiveness of the attacks when they are executed locally (AV:L) or from the same network (AV:A – see Table 3). We observe that the bounce attack was valid only from the local network devices, which in turn affects the environmental AV attribute of the relevant vulnerabilities.

For the http service we set up two different configurations based on an Apache 2.4.41 http server. In the first case we deploy a server with a self-signed certificate and without proper certificate chain validation. In the second case the server is properly configured but it is left vulnerable to referrer checker[10]. As shown in Table 4 `Aness` is able to automatically assess the environmental changes of the relevant vulnerabilties.

		Default SMTP	Unknown Port SMTP
CVE	Base score	Env score	Env score
CVE-2021-25218	7.5 High	7.5 High	9.4 Critical
CVE-2021-25216	9.8 Critical	9.5 Critical	9.4 Critical
CVE-2017-5930	2.7 Low	2.7 Low	6.7 Medium
CVE-2021-25214	6.5 Medium	6.5 Medium	8.3 High
CVE-2020-8625	8.1 High	7.8 High	7.7 High
CVE-2021-35525	5.3 Medium	5.3 Medium	9.4 Critical
CVE-2019-16791	5.9 Medium	3.7 Low	7.7 High
CVE-2017-10140	7.8 High	3.3 Low	7.3 High

Table 2 – SMTP Base - Environmental score

		Local	Adjacent Network
CVE	Base score	Env score	Env score
CVE-2020-9273	8.8 High	7.6 High	7.1 High
CVE-2020-9272	7.5 High	8.6 High	8.2 High
CVE-2019-18217	7.5 High	8.6 High	8.2 High

Table 3 – FTP Base - Environmental score

		Case 1	Case 2
CVE	Base Score	Env Score	Env Score
CVE-2020-13950	7.5 High	9.4 Critical	6.5 Medium
CVE-2020-1927	6.1 Medium	9.6 Critical	6.1 Medium
CVE-2019-1934	8.3 High	8.3 High	5.4 Medium

Table 4 – HTTP Base - Environmental score

5.2 Aness functionality behavior in a system with unknown services

In addition to our primary validation approach in a testbed environment, we also validated the effectiveness

[10] https://Nmap.org/nsedoc/scripts/http-referer-checker.html

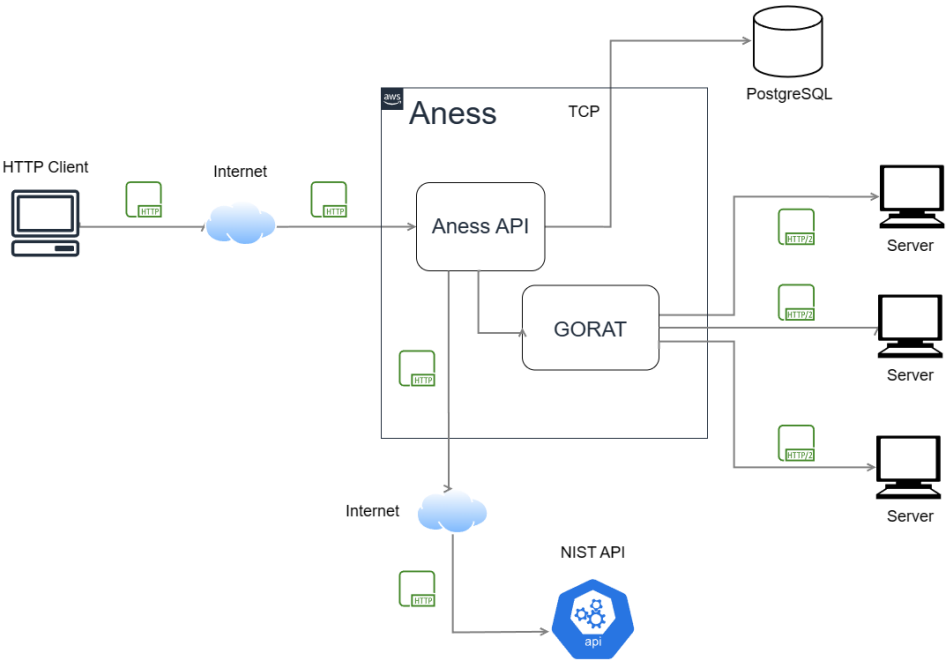


Fig. 5 – Aness architecture and workflow

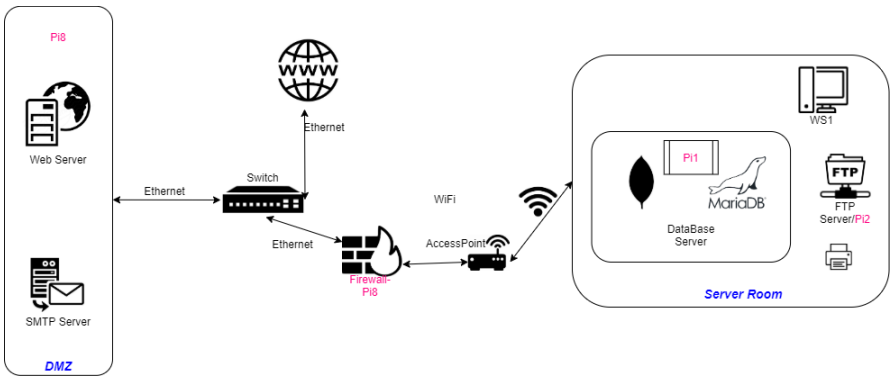


Fig. 6 – Experimental topology

and reliability of our proposed framework in a real-word scenario, wherein we tested a randomly selected server, operating within the network infrastructure of the University of Piraeus. The main goal was to assess the framework's responsiveness and accuracy in a context distinct from the controlled test environment in which it was initially developed and tested. By subjecting our framework to this unanticipated scenario, we sought to ascertain its capability to correctly identify and address vulnerabilities and security-related concerns beyond the confines of controlled conditions. This approach is instrumental in enabling a transparent depiction of the tool's response to a randomly generated scenario.

It is very important to mention the conditions under which the experiments were carried out. First of all, we had the full permission of the network administrator and the system owner to run the network environmental vulnerability analysis on the IP range of the target network. Only the Aness tool was run, so as not to know any information beyond the IP of the target host. This means that we perform the second part of our methodology, the active scan. The experiment was run five times and we got the same results.

As shown in Table 5, the results include several low level vulnerabilities for the SMTP service. An FTP service was not identified in the tested server, while for HTTP vulnerabilities of Low risk were mainly anticipated. We will discuss these in more detail below.

In the case of SMTP, we observe that for almost all CVEs, the environmental score is significanlty lower than the base score. This happens when the vulnerabilities detected are of a low level. This is mainly due to the fact that the environmental security tests run by Aness for the SMTP exhibited no actual risks on known attacks against SMTP, such as OpenRelay, Strange Port or SMTP enumeration. The results indicate that the SMTP service in the system under examination has been properly hardened, e.g. by updating/patching the service itself and/or by ap-

CVE	SMTP BASE SCORE	SMTP ENVIRONMENTAL SCORE	FTP	CVE	HTTP BASE SCORE	HTTP ENVIRONMENTAL SCORE
2023-34108	8.8 HIGH	4.3 (MEDIUM)	IN THIS CASE	2021-44790	9.8 CRITICAL	"YOUR SYSTEM HAS NO
2022-3569	7.8 HIGH	3.3 (LOW)	CVES ARE	2021-44224	8.2 HIGH	VULNERABILITIES THAT ARE
2021-33913	9.8 CRITICAL	5.3 (MEDIUM)	NOT APPLICABLE	2020-13938	5.5 MEDIUM	RELATED TO SSL AND CERTIFICATES.
2021-33912	9.8 CRITICAL	5.3 (MEDIUM)		2018-1312	9.8 CRITICAL	SO, THE SYSTEM'S ENVIRONMENTAL.
2021-35525	5.3 MEDIUM	5.3 (MEDIUM)		2018-1283	5.3 MEDIUM	SCORE MAY BE '0'
2020-12063	5.3 MEDIUM	5.3 (MEDIUM)		2017-15715	8.1 HIGH	
2019-16791	5.9 MEDIUM	3.7 (LOW)		2017-15710	7.5 HIGH	THIS SCORE MAY BE FICTITIOUS
2012-0812	6.1 MEDIUM	4.7 (MEDIUM)		2016-8612	4.3 MEDIUM	BECAUSE YOUR SYSTEM MAY BE
2017-10140	7.8 HIGH	3.3 (LOW)		2017-9798	7.5 HIGH	EXPOSED TO OTHER ATTACKS NOT
2017-5930	2.7 LOW	2.7 (LOW)		2016-8743	7.5 HIGH	COLNSIDERED IN THIS VERSION
2021-25216	9.8 CRITICAL	5.3 (MEDIUM)		2017-9789	7.5 HIGH	OF THE TOOL."
				2017-9788	9.1 CRITICAL	
				2017-7679	9.8 CRITICAL	
				2017-7668	7.5 HIGH	
				2017-3169	9.8 CRITICAL	
				2017-3167	9.8 CRITICAL	

Table 5 – Real case results table

plying additional network security controls that would mitigate such SMTP-oriented attacks. The conclusion is that in this case, we had reasonable results.

In the case of the FTP protocol, the result was that no CVE was detected to be active. According to the specification of the `Aness` tool, the table that extracts the CVEs from the NIST database must be empty to get this result. For this to happen: (a) no vulnerabilities exist for the specific FTP instance; or (b) the code has not been updated to any new changes on the website or in the NIST database API regarding the extraction of information; or (c) there is no such service installed. In our case, the code of the tool is updated, so after the analysis was conducted, we verified with the system administrator that no FTP service was running in the examined server. Therefore, the testing result was verified to be correct.

In the case of the HTTP protocol, we first see that the list of identified CVEs is the longest among the tested services, which is quite reasonable. Then we notice that for each CVE a base score is provided, as extracted by `Aness` from the NIST database. However, in the environmental score, there is a message explaining that the system does not appear to have any SSL-related or Certificate-related vulnerabilities, which are the main examined vulnerability indicators of `Aness` for the HTTP service. As indicated by the tool, this may require further manual investigation by the administrator, since it implies that: (a) either all HTTP vulnerabilities are properly mitigated or (b) vulnerabilities may exist beyond the scope tested by this tool. From a technical point of view, since `Aness` employs various HTTPS security tools like sslyze, the results may be used by the administrator as a first level of assurance that "low hanging fruits" on certificate and SSL/TLS configuration do not exist. If needed, further investigation on other mitigation controls related with HTTP vulnerabilities may be additionally run for additional assurance.

6. DISCUSSION - CONCLUSION

In this paper, we have presented a comprehensive methodology and tool aiming at automating the assessment of environmental security analysis for network services. This approach relies on the establishment of relationships between network connections, network security systems, the various services in operation and the underlying software components. Through this systematic framework, we have sought to streamline the vectorization of network interfaces to be assessed, thereby increasing the effectiveness of the assessment process.

For each of the identified network interfaces, our approach facilitates the formulation of protocol-specific passive and active security tests. These tests, which are designed to be executed automatically, contribute significantly to a wider assessment of the true environmental security state characterizing each examined link. Since the tool can automate a series of known passive and active security tests, it may assist the administrator in quickly acquiring a reasonable perception of the security exposure of network services. Although it may not provide a full assurance for the security level of the tested network services, it may be used as an initial evidence of their environmental security state in a "one-click" manner. The time gained for the administrator in comparisons to independently performing all the security tests covered by `Aness`, may be utilized by the administrators to perform targeted and deeper additional security tests may then be executed at various network or software layers.

However, besides its benefits, we have also identified areas where improvements can be made to enhance its effectiveness. Primarily, it is important to acknowledge that the `Aness` tool heavily relies upon the capabilities of the Nmap tool. Consequently, any discrepancies or errors within the Nmap and the .nsa scripts can potentially impact the efficacy of the tool's results. This necessitates vigilance and periodic review to ensure the seamless alignment of these components.

Furthermore, the tool's operation hinges on the utilization of the NIST vulnerability database, a dependency that is inherently robust and unlikely to engender issues. An additional aspect meriting consideration pertains to the augmentation of the tool's protocol coverage. By expanding its capabilities to encompass a broader spectrum of protocols, the tool could potentially offer a more comprehensive assessment of security in diverse network environments. Thus, from a technical perspective, the tool's continued development should involve periodic evaluations to mitigate any redundancy stemming from embedded third-party tools and to accommodate an expanding repertoire of protocols for assessment. These considerations stand as integral facets of maintaining and enhancing the tool's efficacy in the realm of security assessment.

6.1 Future plans

In the future, we plan to expand the list of supported protocols, extending our reach to a wider range of network communication standards. As it can be seen from the source code of our framework, each protocol is handled independently, and does not depend on the others. In addition, coordination of all protocol-specific implementations is handled by a centralized orchestrator. This design allows for extensibility, to easily add capabilities for testing additional protocols. At the same time, we plan to refine and extend the criteria that underpin our security testing for each supported protocol. This enhancement will strengthen our ability to provide increasingly granular insight into the security landscape of network interfaces.

To accelerate the pre-computation of protocol analysis logic and improve the dynamic mapping of attacks to specific Common Vulnerabilities and Exposures (CVEs), we are actively exploring the integration of machine learning techniques into our methodology. This strategic addition promises to speed up the assessment process while maintaining a high level of accuracy. Our research also includes the implementation of more efficient and effective security policy extraction rules derived from security systems such as firewalls and Intrusion Detection Systems (IDS). The use of machine learning approaches in this context has the potential to optimize rule extraction and thus improve the overall security posture within network environments.

REFERENCES

[1] FIRST. "Common Vulnerability Scoring System version 3.1 Specification Document Revision 1". In: *NIST* (2019), pp. 1–24. URL: https://www.first.org/cvss/.

[2] Songyang Wu, Yong Zhang, and Wei Cao. "Network security assessment using a semantic reasoning and graph based approach". In: *Computers and Electrical Engineering* 64 (2017), pp. 96–109. ISSN: 00457906. DOI: 10.1016/j.compeleceng.2017.02.001. URL: https://doi.org/10.1016/j.compeleceng.2017.02.001.

[3] Georgios Spanos, Angeliki Sioziou, and Lefteris Angelis. "WIVSS: A new methodology for scoring information systems vulnerabilities". In: *ACM International Conference Proceeding Series* (2013), pp. 83–90. DOI: 10.1145/2491845.2491871.

[4] Diptiben Ghelani. "Cyber Security, Cyber Threats, Implications and Future Perspectives: A Review". In: *Science PC* (Sept. 2022). DOI: 10.22541/au.166385207.73483369/v1. URL: https://doi.org/10.22541%2Fau.166385207.73483369%2Fv1.

[5] Danny Velasco Silva and Glen Rodríguez Rafael. "Ontologies for network security and future challenges". In: *Proceedings of the 12th International Conference on Cyber Warfare and Security, ICCWS 2017* (2017), pp. 541–547. arXiv: 1704.02441.

[6] Anoop Singhal. "Security Ontologies for Enterprise Level Risk Assessment". In: *Network* (2012).

[7] Andreas Ekelhart, Stefan Fenz, Markus Klemen, and Edgar Weippl. "Security ontologies: Improving quantitative risk analysis". In: *Proceedings of the Annual Hawaii International Conference on System Sciences* (2007), pp. 1–7. ISSN: 15301605. DOI: 10.1109/HICSS.2007.478.

[8] Andrew Simmonds, Peter Sandilands, and Louis Van Ekert. "An ontology for network security attacks". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3285 (2004), pp. 317–323. ISSN: 16113349. DOI: 10.1007/978-3-540-30176-9_41.

[9] Leo Obrst, Penny Chase, and Richard Markeloff. "Developing an ontology of the cyber security domain". In: *CEUR Workshop Proceedings* 966 (2014), pp. 49–56. ISSN: 16130073.

[10] Mario Vega-Barbas, Víctor A. Villagrá, Fernando Monje, Raúl Riesco, Xavier Larriva-Novo, and Julio Berrocal. "Ontology-based system for dynamic risk management in administrative domains". In: *Applied Sciences (Switzerland)* 9.21 (2019). ISSN: 20763417. DOI: 10.3390/app9214547.

[11] Elena Doynikova, Andrey Fedorchenko, and Igor Kotenko. "Ontology of metrics for cyber security assessment". In: *ACM International Conference Proceeding Series* (2019). DOI: 10.1145/3339252.3341496.

[12] Sami Petäjäsoja, Heikki Kortti, Ari Takanen, and Juha Matti Tirilä. "IMS threat and attack surface analysis using common vulnerability scoring system". In: *Proceedings - International Computer Software and Applications Conference* (2011), pp. 68–

73. ISSN: 07303157. DOI: 10.1109/COMPSACW.2011.22.

[13] Liu Rui, Yan Danfeng, Lin Fan, and Yang Fangchun. "Optimization of hierarchical vulnerability assessment method". In: *Proceedings of 2009 2nd IEEE International Conference on Broadband Network and Multimedia Technology, IEEE IC-BNMT2009* 95.2 (2009), pp. 458–462. DOI: 10.1109/ICBNMT.2009.5348535.

[14] Oluwasefunmi T. Arogundade, Adebayo Abayomi-Alli, and Sanjay Misra. "An Ontology-Based Security Risk Management Model for Information Systems". In: *Arabian Journal for Science and Engineering* 45.8 (2020), pp. 6183–6198. ISSN: 21914281. DOI: 10.1007/s13369-020-04524-4.

[15] Jesus Gonzalez and Mauricio Papa. "Passive scanning in modbus networks". In: *IFIP International Federation for Information Processing* 253 (2007), pp. 175–187. ISSN: 15715736. DOI: 10.1007/978-0-387-75462-8_13.

[16] Quang Do, Ben Martini, and Kim Kwang Raymond Choo. "Cyber-physical systems information gathering: A smart home case study". In: *Computer Networks* 138 (2018), pp. 1–12. ISSN: 13891286. DOI: 10.1016/j.comnet.2018.03.024.

[17] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. "Cyber scanning: A comprehensive survey". In: *IEEE Communications Surveys and Tutorials* 16.3 (2014), pp. 1496–1519. ISSN: 1553877X. DOI: 10.1109/SURV.2013.102913.00020.

[18] Pranshu Bajpai, Aditya K. Sood, and Richard J. Enbody. "The art of mapping IoT devices in networks". In: *Network Security* 2018.4 (2018), pp. 8–15. ISSN: 13534858. DOI: 10.1016/S1353-4858(18)30033-3. URL: http://dx.doi.org/10.1016/S1353-4858(18)30033-3.

[19] Matthias Niedermaier, Florian Fischer, Dominik Merli, and Georg Sigl. "Network Scanning and Mapping for IIoT Edge Node Device Security". In: *International Conference on Applied Electronics* 2019-September (2019). ISSN: 18037232. DOI: 10.23919/AE.2019.8867032.

[20] Ashiqur Rahman, Kantibhusan Roy Kawshik, Atik Ahmed Sourav, and Al-Amin Gaji. "Advanced Network Scanning". In: *American Journal of Engineering Research (AJER)* 96.5 (2016), pp. 38–42. ISSN: 2320-0936. URL: www.ajer.org.

[21] Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman. "Can We Classify an IoT Device using TCP Port Scan?" In: *2018 IEEE 9th International Conference on Information and Automation for Sustainability, ICIAfS 2018* (2018). DOI: 10.1109/ICIAFS.2018.8913346.

[22] Dimitris Koutras, Panos Dimitrellos, Panayiotis Kotzanikolaou, and Christos Douligeris. "Automated WiFi Incident Detection Attack Tool on 802.11 Networks". In: *2023 IEEE Symposium on Computers and Communications (ISCC)*. 2023, pp. 464–469. DOI: 10.1109/ISCC58397.2023.10218077.

[23] Artur Balsam, Maciej Nowak, Michał Walkowski, Jacek Oko, and Sławomir Sujecki. "Analysis of CVSS Vulnerability Base Scores in the Context of Exploits' Availability". In: *2023 23rd International Conference on Transparent Optical Networks (ICTON)*. 2023, pp. 1–4. DOI: 10.1109/ICTON59386.2023.10207394.

[24] Vladimir Vasilyev, Anastasia Kirillova, Alexey Vulfin, and Andrey Nikonov. "Cybersecurity Risk Assessment Based on Cognitive Attack Vector Modeling with CVSS Score". In: *2021 International Conference on Information Technology and Nanotechnology (ITNT)*. 2021, pp. 1–6. DOI: 10.1109/ITNT52450.2021.9649191.

[25] Michał Walkowski, Maciej Krakowiak, Marcin Jaroszewski, Jacek Oko, and Sławomir Sujecki. "Automatic CVSS-based Vulnerability Prioritization and Response with Context Information". In: *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2021, pp. 1–6. DOI: 10.23919/SoftCOM52868.2021.9559094.

[26] Richard Amankwah, Jinfu Chen, Patrick Kwaku Kudjo, Beatrice Korkor Agyemang, and Alfred Adutwum Amponsah. "An automated framework for evaluating open-source web scanner vulnerability severity". In: *Service Oriented Computing and Applications* 14.4 (Dec. 2020), pp. 297–307. ISSN: 1863-2394. DOI: 10.1007/s11761-020-00296-9. URL: https://doi.org/10.1007/s11761-020-00296-9.

[27] Sanghoon Jeon and Huy Kang Kim. "AutoVAS: An automated vulnerability analysis system with a deep learning approach". In: *Computers and Security* 106 (2021), p. 102308. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102308. URL: https://www.sciencedirect.com/science/article/pii/S0167404821001322.

[28] Dimitris Koutras, Christos Grigoriadis, Michalis Papadopoullos, Panayiotis Kotzanikolaou, and Christos Douligeris. "Automating environmental vulnerability analysis for network services". In: *2022 IEEE Symposium on Computers and Communications (ISCC)*. 2022, pp. 1–7. DOI: 10.1109/ISCC55528.2022.9912946.

[29] Marek Sikora, Radek Fujdiak, and Jiri Misurec. "Analysis and detection of application-independent slow Denial of Service cyber attacks".

In: *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 2021, pp. 1–6. DOI: 10.1109/ISI53945.2021.9624789.

[30] Jan Grashöfer, Christian Titze, and Hannes Hartenstein. "Attacks on Dynamic Protocol Detection of Open Source Network Security Monitoring Tools". In: *2020 IEEE Conference on Communications and Network Security (CNS)*. 2020, pp. 1–9. DOI: 10.1109/CNS48642.2020.9162332.

[31] Van-Thuan Pham, Marcel Böhme, and Abhik Roychoudhury. "AFLNET: A Greybox Fuzzer for Network Protocols". In: *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. 2020, pp. 460–465. DOI: 10.1109/ICST46399.2020.00062.

[32] Gemini George and Sabu M. Thampi. "A Graph-Based Security Framework for Securing Industrial IoT Networks From Vulnerability Exploitations". In: *IEEE Access* 6 (2018), pp. 43586–43601. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2863244.

[33] Syed Rizvi, Ryan Pipetti, Nicholas McIntyre, Jonathan Todd, and Iyonna Williams. "Threat model for securing internet of things (IoT) network at device-level". In: *Internet of Things* 11 (2020), p. 100240. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100240. URL: https://www.sciencedirect.com/science/article/pii/S2542660520300731.

[34] Murugiah Souppaya and Karen Scarfone. "NIST Special Publication 800-153: Guidelines for Securing Wireless Local Area Networks (WLANs)". In: *NIST* (2012), p. 17. URL: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-153.pdf.

[35] Karen Scarfone and Paul Hoffman. "Guidelines on Firewalls and Firewall Policy Recommendations of the National Institute of Standards and Technology". In: *Nist Special Publication* (2009).

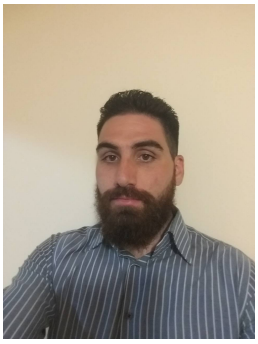
[36] National Institute of Standards and Technology (NIST). In: *Nist Special Publication* (2020). URL: https://www.nist.gov/privacy-framework/nist-sp-800-115.

[37] Dimitris Koutras, George Stergiopoulos, Thomas Dasaklis, Panayiotis Kotzanikolaou, Dimitris Glynos, and Christos Douligeris. "Security in iomt communications: A survey". In: *Sensors (Switzerland)* 20.17 (2020), pp. 1–49. ISSN: 14248220. DOI: 10.3390/s20174828.

[38] Klaus Steding-Jessen, Nandamudi L Vijaykumar, and Antonio Montes. "Using low-interaction honeypots to study the abuse of open proxies to send spam". In: *INFOCOMP Journal of Computer Science* 7.1 (2008), pp. 44–52.

[39] Debdeep Dey, Archisman Dinda, Poornima Panduranga Kundapur, and R Smitha. "Warezmaster and Warezclient: An implementation of FTP based R2L attacks". In: *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE. 2017, pp. 1–6.

AUTHORS



Dimitris Koutras is a PhD candidate in network security, at the University of Piraeus, Department of Informatics. He obtained a bachelor's degree in information technology and computer technology engineering from the Technological educational institute of Central Greece (Lamia-2016). Then he also obtained a Master of Science (M.Sc.) in security management engineering from the University of Piraeus (Piraeus-2019). He currently participates in European research programs as a researcher of University of Piraeus Research center. He also participates as a trainer in various cybersecurity seminars, concerning maritime, healthcare and supply chain. He holds an ISO 27001:2013 lead auditor, certification in information security. His research work concerns network security analysis, risk assessment and operating systems security. Dimitris has collaborated with UPRC on several major projects, including CyberSec for Europe, CyberSecPro, MELITY and ARTEMIS, funded by the European Union and the Greek government. Dimitris is also a published author, with five publications, one of which was praised as an editor's choice.

Panayiotis Kotzanikolaou is an associate professor in network security and privacy at the University of Piraeus, Department of Informatics and the director of the MSc in cybersecurity and data science. He has a degree in computer science (1998) from the University of Piraeus and a Ph.D in ICT security (2003). Formerly, has served as a security auditor at the Hellenic Authority for the Security and Privacy in Communications (ADAE), and has also worked as a security consultant in the private sector. He has participated in various national and European R&D projects. He has participated as a Program Committee member in international conferences and he is a reviewer in various international journals. He has published more than 100 papers in books, journals and international conferences. He is a member of the Greek Computing Society and has received various certifications in information security (CISSP, ISO 27001 Lead Auditor).

Evangelos Paklatzis is an undergraduate student at University of Piraeus, Department of Digital Systems, majoring in "Software and Data Systems". He has been working professionally as a software engineer for over 2 years, developing and maintaining software in projects for the Greek public sector, telecommunications, banking, and investment industries. He has expertise in a variety of programming languages and technologies including C#, Go, Java, SQL, Docker, Kubernetes and Azure. Also, he holds the Azure Fundamentals certification by Microsoft.

Christos Grigoriadis is a highly qualified and experienced lecturer with a background in production and management engineering and secure engineering technologies.He also holds an ISO 27001:2013 lead auditor certification in information security. Currently, he is conducting his PhD research on the topic of machine learning methodologies for the identification and assessment of cumulative vulnerabilities and cascading attacks on interconnected systems. Christos has worked with UPRC on several significant projects, including CyberSec4Europe, MELITY, and ARTEMIS, funded by the European Union and the Greek government. He has also contributed to the projects AI4Healthsec and Cybersecpro as a researcher for the Belgian company Focal Point. Christos is also a published author, with nine publications in journals such as ACM Sensors and Elsevier Computer and Security.

Christos Douligeris is a professor at the Department of Informatics, University of Piraeus, Greece. Formerly, he held positions with the Department of Electrical and Computer Engineering at the University of Miami. He was an associate member of the Hellenic Authority for Information and Communication Assurance and Privacy and the President and CEO of Hellenic Electronic Governance for Social Security SA. Dr. Douligeris has published extensively in the networking scientific literature and he has participated in many research and development projects. He is the co-editor of a book on ''Network Security'' published by IEEE Press/ John Wiley and he is on the editorial boards of several scientific journals, as well as on the technical program committees of major international conferences. He has been involved extensively in curriculum development both in the USA and Greece. His latest work has focused on the use of big data and artificial intelligence techniques in several areas, mainly in telecommunications planning and management and in security analysis of port information systems. Moreover, he has been working in data analytics techniques in learning and education and emergency response oerations. Prof, Douligeris is the director of the Network Research Lab (http://netlab.cs.unipi.gr/gr/), which is closely cooperating with the Security Lab.