A Trusted Platform Module-based, Pre-emptive and Dynamic Asset Discovery Tool

Antonio Jesus Diaz-Honrubia[1], Alberto Blázquez Herranz[1], Lucía Prieto Santamaría[1], Ernestina Menasalvas Ruiz[1], Alejandro Rodríguez-González[1], Gustavo Gonzalez-Granadillo[2], Rodrigo Diaz[2], Emmanouil Panaousis[3], and Christos Xenakis[4]

[1] *Centro de Tecnología Biomédica, E.T.S. de Ingenieros Informáticos, Universidad Politécnica de Madrid, Madrid, Spain, e-mail: {antoniojesus.diaz,alberto.bherranz,lucia.prieto.santamaria, ernestina.menasalvas,alejandro.rg}@upm.es*

[2] *Atos Research & Innovation, Cybersecurity Laboratory, Barcelona, Spain, e-mail: {gustavo.gonzalez, rodrigo.diaz}@atos.net*

[3] *University of Greenwich, London, United Kingdom, e-mail: E.Panaousis@greenwich.ac.uk*

[4] *University of Piraeus, Piraeus, Greece, e-mail: xenakis@unipi.gr*

Abstract

This paper presents an original Intelligent and Secure Asset Discovery Tool (ISADT) that uses artificial intelligence and TPM-based technologies to: i) detect the network assets, and ii) detect suspicious pattern in the use of the network. The architecture has specifically been designed to discover the assets of medium and large size companies and institutions, such as hospitals, universities, or government buildings. Given the distributed design of the architecture, it can cope with the problem of the isolation of different Virtual Local Area Networks (VLANs). This is done by collecting information from all the VLANs and storing it in a central node, which can be accessed by the network administrator, who may consult and visualize the status in any moment, or even by other authorized applications. The collected data is kept in a secure warehouse by the use of a Trusted Platform Module. Moreover, collected data is processed by the use of artificial intelligence in two ways: i) the traffic of each network is analysed so that suspicious patterns can be detected, and ii) identified ports and status are analysed to detect anomalous combinations of open ports in a device.

October 25, 2022

Keywords: Asset discovery, Cyber security, Network visualization, Artificial intelligence, Trusted Platform Module

1. Introduction

A crucial aspect to take into account when managing the cyber security of an organisation is to have an accurate image of its assets (e.g. network devices, desktops, laptops). These are the most prominent targets of any cyber attack, which attempts to exploit asset vulnerabilities. Therefore, organisations must be able to patch these vulnerabilities after they thoroughly discover their assets.

This paper presents the Intelligent and Secure Asset Discovery Tool (ISADT), which has been developed for discovering information about the devices that are connected to the computer network of the institution, including their IP and MAC addresses, open ports, services in use, and suspicious patterns in the traffic of a port or in the combination of used ports. This information may include the operating system of the devices, their open ports, or services running on them, like database management systems, and web servers. The tool is able to detect changes in the characteristics (e.g., IP addresses, open ports) previously detected devices and promptly discover new devices that are connected to the network.

Having a clear picture of the assets in the organization's network has a crucial importance to detect unauthorized connections, anomalous behaviours of devices and services, or the presence of malware. Therefore, a tool like ISADT can protect the network of any organization where the size of that network makes that it impossible to have all the connected devices under control.

Considering this, one of the key functionalities of the proposed tool is that it copes with the challenge of having isolated Virtual Local Area Networks (VLANs) within the same organisation. This is achieved through a fully-distributed architecture over which ISADT is built, since it will help to have an agent inspecting each network segment. In this way, a software agent that is running in each isolated network will be in charge of discovering all the assets associated with that network, and send all the information to a central node. Then, all the information in this central node can be queried by other applications or by the network administrator to detect unexpected connections or traffic.

The work presented in [1] has demonstrated that a Trusted Platform Module (TPM) [2] can be used to guarantee the integrity of source code and the compiled binaries when using a continuous integration approach. This approach can be used in other contexts to secure the information being stored in different platforms, and not only in a software repository. Given the sensitive nature of the information that this central node will gather, a TPM is used to check whether its hardware and software have been altered. This capability guarantees that the central node cannot receive or share information when the system is compromised.

Finally, the real-time detection of adversarial or inappropriate behaviours in the network is an essential part of cyber risk assessment. One of the main novelties of ISADT is the identification of suspicious behaviours within the VLAN environments, which is achieved through the use of two algorithms based on Artificial Intelligence (AI) techniques. The implementation of these algorithms enables the ISADT capability to detect that: (i) among a set of ports that are usually used together, a port is missing; and (ii) an unusual peak of traffic in a given port is occurring.

Considering this, the tool introduces the following novelties with respect to the state of the art on asset discovery and the processing of the discovered information:

- It handles the problem of network isolation in medium and large size networks.
- It uses the TPM technology to enhance the security of the collected information.
- It uses AI-based algorithms to detect suspicious patterns in the usage of ports.
- It analyses the time series, produced by the traffic in each port, to detect traffic peaks and warn the network administrator in the case that an anomalous peak is detected.

The results about the information collected by the tool and the characteristics that it owns are compared with the most extended tool for the detection of open ports, Nmap [3]. These results show that the proposed ISADT has more capabilities than Nmap and it can scan larger networks faster than it.

The remaining of this paper is structured as follows. Section 2 presents the related works concerning asset discovering, Section 3 describes the proposed distributed architecture of this tool, Section 4 demonstrates the functionalities of the tool, Section 5 presents a discussion about the architecture considering its functionality and Section 6 concludes this paper.

2. Related Work

Nowadays, there is an increasing trend in cyber-attacks. For instance, as it is presented in [4], cyber-criminals acquire, launder, and reinvest about $1.5 trillion annually. Moreover, the European Confederation of Institute of Internal Auditing's (ECIIA) states that cyber security has been the top risk faced by business in the last years [5]. A lot of research has been done to prevent these cyber-attacks and, some of the efforts aim at monitoring the network [6] [7] and keep a perfect knowledge about the devices and systems that are connected to it, as well as their characteristics so that if a new vulnerability is discovered, in any of them, the organization may take immediate decisions preventing cyber incidents, through exploitation of vulnerabilities, at their root.

Fingerprinting is the process of collecting information about the network devices, their characteristics and their associated services [8]. Fingerprinting can be classified into active and passive. In passive methods, network traffic is monitored without actively engaging with the host and without generating any traffic. On the other hand, in active methods, packets are sent to hosts and their responses are analysed.

In the field of active fingerprinting, Barroux J. C. in [9] presents a method that makes use of the information, about the devices in the network, that is gathered by Simple Network Management Protocol (SNMP) packets. The author describes that an integrated resource installs an agent in a selected node from each network. This agent sends the collected information back to the integrated resource which collects and stores it.

The installation of the agent is transparent to users, since it is done by means of remote procedure calls from the integrated resource to the selected node on each network. However, this approach has some drawbacks: the remote installation of software in other devices might represent an intrusion and a possible risk in the target devices. Moreover, the approach does not take into account the problem of network isolation, by which most of VLANs cannot be reached from others. Finally, the proposed system, even though it

can discover part of the software installed in the network, cannot discover all the open ports nor their associated services.

Another approach, not based on SNMP packets, but on the use of Address Resolution Protocol (ARP) spoofing techniques, is presented in [10]. This is based on the idea of forcing all traffic in a network to pass through a central host designated for that purpose, allowing it to detect all devices connected to the network. Although this approach does not need to perform an unauthorised installation of an agent on a remote device, it cannot solve the aforementioned problems of port scanning and network isolation (and what is more, it does not work in a scenario with multiple private networks).

A very powerful tool for active scanning of a network is Nmap [3], which is able to scan open ports and, even, infer the service (with its version) that might be running in the ports. The main disadvantages of Nmap is that it cannot be used for scanning networks with private IP addresses and that it is a non-concurrent tool.

Despite these disadvantages, Nmap has been widely used for network inspection. For instance, it is used in [10] for scanning the devices, their operating systems, and their open ports the network of a university in India. Later, the authors assess the threats and vulnerabilities and create a remediation plan based on those assessments. Similar works can be found in the literature [11] [12].

A tool similar to Nmap, but making use of the powerful concurrent capabilities of the Golang programming language is presented in [13]. While this tool, given its concurrent nature, is faster than the original Nmap, it still have the limitation of not being able to scan networks with private IP addresses.

Regarding passive methods, packet sniffing is the most widespread technique to be used. Packet sniffing intercepts all traffic that goes through a network. The intercepted packets can be decoded, and their data might be obtained. Even though packet sniffing is usually associated with malicious activities to intercept information, it can also be used in an ethical way if confidential information is not monitored. In this sense, it can be used to discover assets in networks, as it is shown in [14] if only packet headers are analysed. In that work, authors also state that it can be used to prevent and detect the intrusions in the network (e.g. detecting new IP addresses' requests).

The system presented in [9] could be considered as a passive method sniffing SNMP packets in each single network in the case that the integrated

resource does not remotely install anything in nodes of the targeted network. Packet sniffing is usually used for network monitoring too. For instance, authors in [15] also sniff SNMP packets to obtain metrics in a network specifically design for cloud computing. The authors propose to install an agent in each network and collect all the information in a non-relational database. In contrast with the work in [9], the installation is not done using intrusive methods.

While previous works focus only on addressing the problem using active or passive methods, ISADT implements a combined approach in which both methods are combined. Therefore, while active methods are used to detect advanced characteristics of the network usage (e.g. the services that are being used), passive methods provide network usage and ensure that even when it is not possible to actively scan the network, some basic information is obtained (e.g. the IP addresses and ports being used). Moreover, ISADT does not use intrusive methods. Conversely, the network administrator needs to install an agent in each of the networks to be monitored. Finally, ISADT includes advanced analysis by making use of AI-based techniques and of a TPM to secure the information, something that is not provided by the state-of-the-art.

3. Intelligent and Secure Asset Discovery Tool (ISADT)

One of the main limitations that existing solutions have is dealing with the network isolation. ISADT copes with this challenge by implementing a distributed approach to detect assets in different isolated networks. Moreover, ISADT uses AI techniques for two purposes; firstly, to detect the ports that are often used together and their associations, making it possible to raise alerts when a port that is commonly used together with others is not actually being detected, then, an analysis of the time series produced by the traffic per unit of time that goes through each port is performed to detect unusual traffic peaks.

The other main innovation of the proposed ISADT is that both security hardening and reliability of communications are provided by the use of a Trusted Platform Module (TPM) [2]. This device is used to ensure that the hardware and the software are not maliciously altered, disabling the Application Programming Interface (API) of ISADT to retrieve or to store information when an essential change is detected.

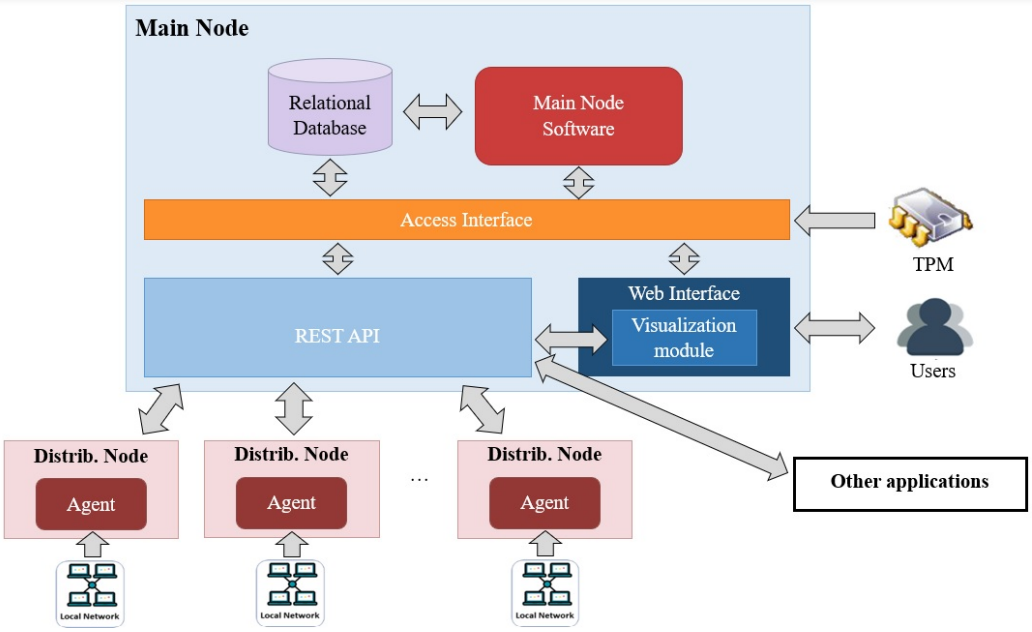


Figure 1: ISADT's distributed architecture.

The distributed architecture of ISADT, is illustrated in Figure 1. In this architecture, similar to [9] or [15], an *agent* must be running in each isolated network. However, in the case of ISADT, the installation of the agent does not imply an unsupervised intrusion in the network, since it must be firstly registered and authorized by the network administrator for each VLAN.

The devices in which the agent is installed are called Distributed Nodes (DNs). Each agent will be in charge of discovering all devices that are connected to the VLANs, which can be reached from DN, and then analysing all traffic that goes through them. All gathered information is sent to a Main Node (MN), which must be reachable from all DNs, and stored in a relational database. ISADT users (i.e., network administrators, cyber security analysts, etc.) and other applications must only interact with this MN, which is, at the same time, in charge of applying different AI-based techniques to find patterns in the usage of ports.

DNs, external applications, and users communicate with the MN by means of a Representational State Transfer (REST) Application Programming Interface (API), which provides a service based on REST-style architecture [16]. The communication with the API is done by the use of secure

methods by which specific DNs can only submit information of their associated VLANs and specific external applications can only access specific information, as it will be explained in Section 3.2.3.

Finally, the architecture incorporates a Web Interface that can be used by the network administrator to define the VLANs to be scanned and the existing DNs. This interface includes a Visualization Module that can be used to inspect the state of VLANs at any timestamp.

3.1. Distributed Nodes

As it has already been stated, DNs are located in the different isolated networks and are in charge of scanning the devices present in them and collecting all their information.

When a specific device is going to be scanned by a DN, the information that is gathered from it is the following: its IP and MAC addresses, its operating system, its list of ports in use (and their status), and, when possible, the list of associated services running on each port and their versions.

To gather this information, the first characteristic to identify for each device is its IP address. Then, the MAC address associated to the IP address is obtained by making an Address Resolution Protocol (ARP) query to the local ARP table. The rest of the information can be discovered by the use of the Nmap tool [3], as described in Section 2. Therefore, Nmap tool is used to retrieve the operating system and the list of open ports with their associated services and versions when possible from the scanned devices. However, ISADT provides several advantages with respect to Nmap, as it will be discussed in Section 4.

3.1.1. Discovery of network assets

Regarding the discovery of network assets, there are three main problems to be solved:

1. A new device might be connected to the network at any instance of time. As any new device might suppose a risk, it is essential to register it as soon as it is connected to the network.
2. An asset has already been registered, but its associated characteristics (i.e., its open ports) change over the time.
3. A device has been protected against scans and is not possible to obtain its assets by means of Nmap or other similar tools.

To solve the first problem, DNs capture the Dynamic Host Configuration Protocol (DHCP) [17] traffic on each of the local networks that are monitoring. When a DHCP-ACK packet is detected, the IP address is stored, its associated host is immediately scanned with Nmap, and the resulting information is sent to the MN.

However, this approach does not provide a solution for those devices which do not make DHCP requests, i.e. when a static IP address has been assigned or when the device has been re-connected to the network and the dynamic IP address that had received before has not expired yet. To solve this, DNs query their ARP cache tables in very short time frames (e.g. every second) to check whether there are IP addresses that have not been detected before. In that case, DNs will scan that host with Nmap and send the information to the MN.

To solve the second problem (i.e., keeping the discovered information up to date), DNs perform scans of the networks in regular time frames whose length can be configured by the network administrator. This scan process starts by sending Internet Control Message Protocol (ICMP) [18] 'Echo request' messages to all the range of IP addresses of the network. When the tool obtains an ICMP 'Echo reply' message from an IP address, it is scanned with Nmap to discover or update that information.

To cope with the third problem in which network devices have been protected and their assets cannot be detected by Nmap and/or they do not response to ICMP 'Echo request' messages, DNs are also configured to intercept all network traffic. As the process of analysing this traffic is done at the same time as the detection of unusual peaks, it is described in section 3.1.2.

Finally, when a scan of any of the previous types is done, the information is sent to the MN as a *snapshot*, including the timestamp at which the scan started and the timestamp at which it finished. If, for any reason, the MN cannot be reached (i.e. temporal lost of connectivity), the information is temporally stored in a light embedded database of the DN, and will reattempt to send the information periodically.

3.1.2. Traffic analysis

Traffic analysis consists of analysing all IP packets that go through the VLAN that is being scanned by a DN. This analysis is performed to achieve two objectives: i) to detect IP addresses and/or ports in use that have not been detected before; and ii) to detect unusual traffic peaks on each port. With this purpose, the algorithm based on sliding windows that is described

in Algorithm 1 has been designed, being the parameter n the number of sliding windows to be used.

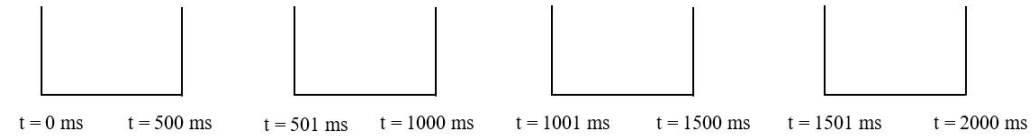
Algorithm 1 sliding_window(n)

```
prev_window = 0
while true do
    Wait until new network packet is received
    timestamp = timestamp of the packet in miliseconds
    current_window = (timestamp / 500) % n
    Include the IP, port and size of the packet in current_window
    for all windows between prev_window and current_window do
        for all IP and port combinations in window do
            if IP and port are present in, at least, 75% of windows then
                Set this IP and port as "in use"
                Add packet size to accumulated size for this IP and port in the last minute
                Remove any data older than 2 hours
                if number of packets ≥ 20 then
                    μ = mean of all sizes per minute for this IP and port
                    σ = standard deviation of all sizes per minute for this IP and port
                    if size of the last minute ≥ μ + 2σ then
                        Raise new traffic peak notification
                    end if
                end if
            end if
        end for
    end for
    prev_window = current_window
end while
```

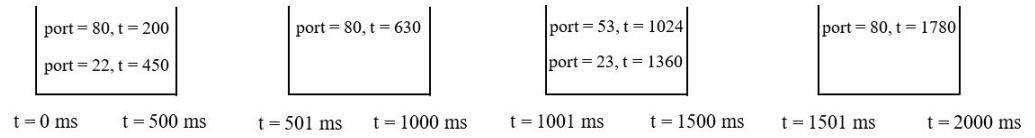
The algorithm consists of defining n time windows between two timestamps with a length of 500 miliseconds (ms). Figure 2(a) shows an example with $n = 4$ windows, starting at time $t = 0$.

When a packet arrives, its timestamp is checked, and it is placed in the corresponding window. Figure 2(b) shows an example after detecting the following six packets for the same IP address:

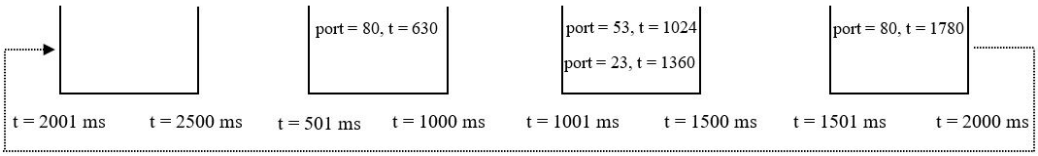
1. Port number 80, with timestamp 200 ms.
2. Port number 22, with timestamp 450 ms.
3. Port number 80, with timestamp 630 ms.
4. Port number 53, with timestamp 1024 ms.
5. Port number 23, with timestamp 1360 ms.
6. Port number 80, with timestamp 1780 ms.



(a) Initialization of the algorithm.



(b) Windows status after detecting six packets.



(c) Windows status after the first overflow.

Figure 2: Sliding windows algorithm with $n = 4$.

When the timestamp of a new packet is greater than the upper limit of the last window, an *overflow* happens. In this situation, the algorithm acts as a circular list and re-uses the first window as the next one, re-defining its time limits, as shown in Figure 2(c), after detecting a packet whose timestamp is greater than 2000 ms.

When an overflow happens, the content of the window that is going to be replaced needs to be removed from it. The packets that are going to be removed are considered as new assets if they had not been discovered before. Many applications (the TCP-based ones) open ephemeral ports to maintain the communication and these ports should not be considered as actual assets. For this purpose, a port is not considered as ephemeral if it is present in, at least, the 75% of the windows at the removal time. In this situation, the port is considered as a new asset and sent to the MN. In the example of Figure 2, port 80 is present in three of the four windows when it is going to be removed, so it will be communicated to the MN, while port 22 is only in the first window, is not considered as a new asset.

Then, according to the other functionality of the algorithm, the sizes of the packets that are going to be removed are temporally aggregated in one

minute groups. By storing these, what is actually stored is the throughput of the ports in bits per minute (bpm). Only the information of the last two hours is stored. For each port, the mean, μ, and the standard deviation, σ, of its throughput along the time are calculated.

When there are at least 20 packets for a specific port (so that the sample is significant) and a new packet is received, it is checked whether the throughput of the last minute is greater than $\mu + 2\sigma$. In that case, it is considered as a having a traffic peak in that minute and a notification of a traffic peak is sent to the MN.

3.2. Main Node

The MN is the global coordinator of the system and in charge of storing all the information that is collected by the DNs. Moreover, any external application and users must retrieve the information through this node. A copy of the software of the agent running in every is also stored in this MN and, when a new DN is going to be deployed, it must connect to the MN to download a trusted copy of the software.

Therefore, the functionalities of the MN make of it the most critical one, since: i) it contains a lot of information that might be exploited with malicious purposes; and ii) there are different users, services, and applications that must be granted with read and/or write privileges. Because of this, the architecture of this node is composed of several layers and modules, which ensures information security and provides specific users access to specific information.

Finally, as it will be explained in section 3.2.1, the MN Software is in charge of analysing the ports in use in each device that have been detected, finding patterns about the ports that are usually used together and raising alarms in the case a port is not being used, while a set of ports that are usually used together with it are being used.

3.2.1. Main Node Software and association rules

The Main Node Software is in charge of carrying out any data processing that is needed before storing the information in the relational database. Specifically, it takes any information given by the Access Interface and stores it in Relational Database.

Moreover, this software is in charge of discovering patterns in the utilization of ports. Specifically, the Apriori algorithm [19] is used to find association rules between the ports. This algorithm identifies assets (ports in this case)

which are frequently used together. For doing this, it starts by counting the frequency of each port. Then, it counts the frequency of each pair of ports, and the frequency of each triplet of ports.

When doing this process, if a tuple of ports does not obtain a frequency greater than a specified parameter, supersets of that set (any set that already includes the ports in that tuple), is not considered to make the algorithm more efficient. This parameter threshold is known as support and in the proposed architecture it has been set to 0.5.

Another parameter of the algorithm is the confidence, which indicates the relative support of the whole rule with respect to the support of its antecedent. In the proposed architecture, the confidence has been set to 0.97. The resulting rules will look like the one in Equation 1.

$$\{(80, \text{open}), (22, \text{open})\} \Rightarrow \{(23, \text{filtered}), (8080, \text{open})\} \quad (1)$$

By keeping track of the ports discovered through each scanning iteration, the algorithm generates a rule-based Machine Learning model that computes the likelihood of appearance of an specific set of ports given a machine's status (set of ports in a device).

After receiving a scanning snapshot, the MN software looks whether the ports with highest probability to appear running in each host are in fact there. In negative case, an alarm is generated, stored and prompted in the web interface. Moreover, to keep an updated stamp of the actual situation, the association rules are derived after a configurable number of receptions of scanning snapshots.

Therefore, using the rule in Equation 1, if in a new snapshot, a device has the ports 80 and 22 with status set as "open", but the port 8080 is not used and the port 23 has "open" in its status, two alerts will be generated: one to indicate that the port 8080 is not being used, and it would be very likely to be in use, and to indicate that the status of the port 23 is not the expected one.

3.2.2. Relational database

The Relational Database is in charge of storing the information about the networks to be scanned, which is directly provided by the system administrator. In this regard, the relational database firstly stores information about the appropriate credentials. Then, the system administrator registers the information about the DNs that will compose the whole system, as well

Table 1: Available endpoints in the REST API.

Resource	**GET**	**POST**
/vlans	Returns a list of all VLANs, including their associated snapshots.	Creates a new snapshot for a specific VLAN.
/vlans/snapshot-id	Returns all the information gathered for a specific VLAN and snapshot.	Method not allowed (405).
/configurations	Returns the configuration of a specific Distributed Node.	Method not allowed (405).
/presence_check	Returns whether or not the MN has stored information from a specific IP and MAC.	Method not allowed (405).

as the network IP address and mask of the VLANs to be scanned by each specific DN. When this information is stored, the relational database will also store all the information about each scan that is sent by the DNs.

3.2.3. REST API

All the reading and writing operations which involve the MN, must be done by means of calls to the REST API module, which provides a secure access to the information.

This API incorporates several secure endpoints to send and retrieve information. The API has been secured throughout a token-based system, in which each actor owns an API key which will grant specific permissions to access specific endpoints and methods. Token validation is performed before the access to any of the functionalities, thus being necessary to be passed as a parameter to any endpoint.

Taking everything into consideration, Table 1 shows the endpoints (associated to their GET or POST method) that have been defined .

3.2.4. Web Interface and Visualization Module

The Web Interface is a web service that is used to administrate the tool. When a new node is registered, the Access Interface assigns a new token to

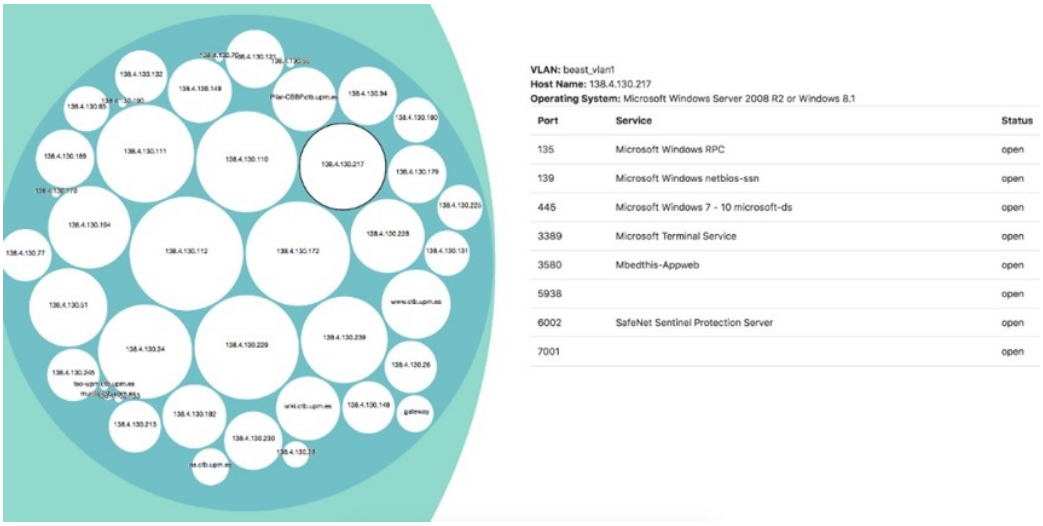


Figure 3: Viusalization of a specific VLAN and the information about one of its assets at a specific timestamp.

it and this token is shown in this web interface so that the node starts to operate.

In addition, the system administrator may grant access to specific endpoints to other applications through the Web Interface. Similarly as in the case of the DNs, a token for these applications is shown and it must be used by them when making a request to the REST API.

The Web Interface includes a Visualization Module that allows the system administrator to visualize each VLAN and its detected assets and alerts at a specific timestamp. In order to do this, the user must define a date and time and the tool will show a circle-pack diagram in which each circle represents a VLAN (its identifier is shown on it).

If the user clicks on a specific VLAN, the visualizer focuses on that network and the hostname of each device is displayed. Moreover, if the user clicks on a specific device, the information about this device that is stored in the tool is displayed on the right part of the diagram. This is shown in Figure 3.

3.2.5. Access Interface

The Access Interface acts as an intermediate layer between the REST API and the other components of the MN. It is in charge of coordinating the flow of information in the MN and also regulates the information that

can be read from or written in the Relational Database or processed by the Main Node Software. In this sense, it provides a secure access to the API by means of a token system.

When a new agent is registered in the Access Interface, it generates a new token that must be used by the agent when it is going to use the REST API. This token is generated using the information in the TPM of the MN, as it will be explained in the next section, and may change if the node may be compromised so that the agents cannot store new information or retrieve the existing one.

3.3. Trusted Platform Module

A TPM is a hardware device designed to provide hardware-based security functions to ensure platform integrity and securely storing keys. It is part of a computer's Trusted Computing Base (TCB), i.e. the collection of software and hardware components that enforce its security policy.

TPM's role in TCB is to ensure its trustworthiness by performing a measured boot, a method of booting in which each component in the boot sequence checks the next component before passing control to it. After each individual checking process, each component sends a hash to the TPM, which is stored in its Platform Configuration Registers (PCRs).

Moreover, in order to enable the host system to perform configuration checks during every power-on state, TPM must reflect any changes occurring in the system's integrity during its lifetime. It does it by updating the correspondent PCRs based on a hash of the data used to modify a specific component, derived by Equation 2.

$$\mathrm{PCR}_i = \mathrm{SHA}(\mathrm{Concatenation}(\mathrm{PCR}_i, \mathrm{SHA}(\text{modification data}))) \qquad (2)$$

By keeping a record of both the original PCR state and the digest resulting from the hashing of the modification data, the system is able to verify the expected new value of the PCR at power-up (and would produce an error or disable specific functionalities if it does not match with the stored PCR).

The proposed method to secure the MN is based on the ability to access PCRs on execution time, being able to monitor whether a modification of the system's integrity took place or not.

3.3.1. Proposed method to secure data

As previously mentioned, the methodology to secure the information contained in the MN involves periodically accessing PCRs in order to monitor the system's integrity. With this purpose, the MN keeps track of potential hardware changes taking place in the PCRs 0 to 7, as these ones can reveal any hardware configuration in the system [20].

After accessing these PCRs, the application will generate its own hash based on their information. This hash will then be concatenated to the tokens associated to every credential present on the application's database.

Since these new tokens are perfectly separable into its two main components, the application is able to verify the system's integrity each time it retrieves the state of the PCRs from the host system. By hashing the newly obtained PCRs and comparing them to the PCR-hash present in its tokens, in case of mismatch between them, the system will be informed of their modification, prompting the system administration of this occurrence and disabling the correspondent tokens (and, hence, the services that use those tokens) until the occurrence is solved or verified.

4. ISADT Testing and Validation

Asset discovery is a crucial task that needs to be accurately performed. The discovery of assets that are connected to the network of an organization allows to identify not only unauthorized connections, but also to identify their characteristics and potential vulnerabilities that can compromise an organization. Current asset discovery tools offer a set of functionalities that covers most of the necessities of system administrators to potentially protect a network. However, these tools still lack several functionalities that would substantially improve the potential necessities in all the facets around the discovery of assets.

Our proposed solution relies on the creation of a set of innovative components and functionalities that are built on top of the very well-known Nmap tool. While Nmap provides a robust functionality, the tool lacks several features that would help to improve the cyber security of an organization. ISADT has been designed and developed to offer new features and provide a ground-breaking solution that would help to improve the analysis of network organizations.

The current section will deep into the analysis of some of these characteristics from both a qualitative and quantitative point of view.

4.1. New features

As it has been described, ISADT implements a set of new functionalities that enhance the capabilities of the base tool (Nmap) in the task of discovering new assets. Table 2 shows a comparison between ISADT and Nmap for the different implemented capabilities that cannot be quantitative measured.

Table 2: Comparison between ISADT and Nmap for new qualitative-evaluated functionalities.

Feature	**ISADT**	**Nmap**
Execution in a distributed environment over different VLANs.	Implemented distributed architecture that allows automatic execution of the discovery of assets in independent VLANs. Each VLAN can be scheduled differently.	It is necessary to execute Nmap independently in each VLAN.
Automatic VLAN-data centered integration.	Implemented capability of integrating different VLAN data in a single point (main node) as part of its distributed architecture.	No integration capabilities of data from different executions/VLANs.
Use of TPM as a protection system to detect hardware modifications and disable scanning functionalities to protect system integrity.	Integrated mechanism that makes use of TPM (when available – and with optionally enabling) to detect potential hardware changes that can imply a system breach.	Lacks this functionality.
Port/service pattern recognition to identify potential vulnerabilities based on devices not following common patterns.	Implemented functionality based on a machine learning algorithm named A priori that allows to identify tuples of ports/services that typically are open/available together. This functionality allows to identify machines with differences on those patterns to alert about them to be further investigated as potential threats.	Lacks this functionality.
Traffic network peaks automatic analysis (ephemeral port analysis).	Implemented functionality to measure if there are peaks on the network traffic to identify potential threats. ISADT focuses on the ephemeral ports as they are used in a short time for TCP/IP communications.	Lacks this functionality.
Real-time asset detection.	Implemented traffic analysis that allows to detect when a new device is connected to a network, allowing to execute immediate scans over the device without needing to wait for the scheduled analysis.	Lacks this functionality.

4.2. Distributed architecture complexity evaluation

The distributed architecture proposed for our tool is based on the idea that different VLANs can be scanned by applying Nmap to each of those VLANs. Apparently, behind this approach the results between Nmap as a standalone solution and ISADT could be considered as similar, and hence, can be thought as similar results would be provided. However, some considerations need to be taken into account. ISADT provides further capabilities for the recognition of devices that Nmap is not providing. When an Nmap execution is running, devices that are connected and disconnected in the network might not be detected by Nmap. However, ISADT could detect them by using the traffic sniffing capabilities to execute ad-hoc scans over these devices.

Another element to consider are the parameters referred to the complexity of the execution. Temporal and spatial complexity are important elements that need to be evaluated.

The methodology for the evaluation of the tool and its comparison against Nmap is based on the execution of both tools in the same scenario. For this evaluation, we have prepared 6 different execution scenarios:

- Scenario 1 (S1): 1 VLAN with 2 static machines.
- Scenario 2 (S2): 1 VLAN with 5 static machines.
- Scenario 3 (S3): 1 VLAN with 10 static machines.
- Scenario 4 (S4): A total of 3 VLANs were prepared. VLAN1 contains 1 static machine. VLAN 2 contains 3 static machines. VLAN3 contains 2 static machines and 2 DHCP machines.
- Scenario 5 (S5): A total of 3 VLANs were prepared. VLAN1 contains 2 static machines. VLAN 2 contains 6 static machines. VLAN3 contains 4 static machines and 3 DHCP machines.
- Scenario 6 (S6): A total of 3 VLANs were prepared. VLAN1 contains 4 static machines. VLAN 2 contains 10 static machines. VLAN3 contains 6 static machines and 4 DHCP machines.

Each machine has a different configuration (operation system, open ports and services). The full list of machines configurations per scenario can be

found in supplementary materials. The static machines are always connected to the VLAN while the DHCP machines are dynamically connected in a specific moment of the scan process.

Therefore, several experiments were run. For each scenario, both Nmap and ISADT Distributed Node, have been executed 20 times in order to obtain statistically significant results. In each execution the following parameters were measured:

- Time (in seconds): how long takes to the tool to perform the scan of the corresponding VLAN. In scenarios 4, 5 and 6 these parameters are independently measured per VLAN and the results among tools will be compared in those VLANs as different VLANs might take different times due to the network load.
- Maximum virtual memory (in Megabytes): the maximum amount of memory consumed by the process by which the tool was executed.
- Minimum virtual memory (in Megabytes): the minimum amount of memory consumed by the process by which the tool was executed.
- Average virtual memory (in Megabytes): the average amount of memory consumed by the process by which the tool was executed.
- Number of ports/services discovered in each machine: how many (and which ones) ports were discovered in each of the machines. The idea of this measure is, firstly, to measure how many devices were detected; secondly, to compare the results in terms of the number of ports/services found.

To measure the execution times, the *date* command before and after executing each tool has been run (the difference between these values provides the execution time). To measure the memory values, the PID of each process has been obtained and, then, the */proc/<pid>/status* file has been periodically queried. Table 3 shows the mean values of these parameters in the 6 scenarios (S1 to S6) for the 20 executions for each VLAN and configuration using an ISADT Distributed Node, while Table 4 presents these same parameters using the Nmap software.

Figure 4 shows a comparison of the execution times of ISADT (blue) and Nmap (red), according to the number of ports in each scenario and VLAN.

Moreover, a trend line is included for each tool. It can be seen that Nmap takes less time than ISADT when there are few ports (i.e., approximately, less than 7), while ISADT is more faster when there are more ports, what is the most common scenario in real networks. This is due to the parallel nature of ISADT, needing some more time for initialization, but taking advantage of this when the number of ports to be scanned is large enough. Regarding the memory, Nmap usually requires less memory, since it does not require to store information about more threads in memory. However, it can be seen that the memory usage is constant in ISADT, regardless of the number of ports to be discovered.

Table 3: Mean of collected data from a Distributed Node for each scenario and VLAN.

Scenario & VLAN	**Time (s)**	**Max. mem. (MB)**	**Min. mem. (MB)**	**Avg. mem. (MB)**	**Number of ports**
S1 - VLAN1	52.9	11,015.2	520.8	10,334.8	6
S2 - VLAN1	176.2	11,015.2	532.8	10,755.7	17
S3 - VLAN1	187.1	11,016.2	575.2	10,773.5	28
S4 - VLAN1	32.2	11,015.8	569.8	10,141.4	2
S4 - VLAN2	129.5	11,015.4	525.6	10,676.5	8
S4 - VLAN3	51.0	11,015.6	854.6	10,242.3	8
S5 - VLAN1	46.5	11,014.8	567.0	10,310.4	5
S5 - VLAN2	58.9	11,015.4	564.8	10,409.0	9
S5 - VLAN3	130.0	11,015.6	850.8	10,661.9	15
S6 - VLAN1	83.1	11,016.0	767.6	10,689.9	15
S6 - VLAN2	56.5	10,789.0	868.2	10,176.3	9
S6 - VLAN3	178.5	11,014.4	880.0	10,648.9	21

4.3. Validation of the new features

However, more important than the performance analysis is the validation of the new features included in ISADT that Nmap does not include.

This section aims at validating these new features. For that purpose, the set-up of two experiments is detailed and, then, their results are discussed.

4.3.1. Experimental set-up

In the first setup, a controlled environment with virtual machines using Vagrant [21] as Hypervisor, since it provides very simple management of the virtual machines through a command-line interface. Moreover, synthetic traffic has been designed to carefully test all the functionalities of the system (except the Apriori algorithm, since there are few data in the experiment for

Table 4: Mean of collected data from Nmap for each scenario and VLAN.

Scenario & VLAN	Time (s)	Max. mem. (MB)	Min. mem. (MB)	Avg. mem. (MB)	Number of ports
S1 - VLAN1	26.2	8,748.0	1,594.4	7,448.0	5
S2 - VLAN1	231.0	8,753.0	1,600.0	8,598.0	16
S3 - VLAN1	371.7	8,748.0	1,712.8	8,653.6	27
S4 - VLAN1	2.7	1,602.4	1,602.4	1,602.4	1
S4 - VLAN2	102.1	8,748.0	1,711.6	8,407.8	7
S4 - VLAN3	22.7	8,748.0	1,811.2	7,372.3	5
S5 - VLAN1	23.6	8,748.0	1,746.4	7,358.6	4
S5 - VLAN2	39.8	8,748.0	1,764.0	7,900.2	8
S5 - VLAN3	120.2	8,748.0	2,165.8	8,478.0	11
S6 - VLAN1	107.7	8,748.0	2,558.4	8,466.8	14
S6 - VLAN2	45.6	8,748.0	1,632.8	7,983.8	8
S6 - VLAN3	266.9	8,748.0	1,863.6	8,619.6	16

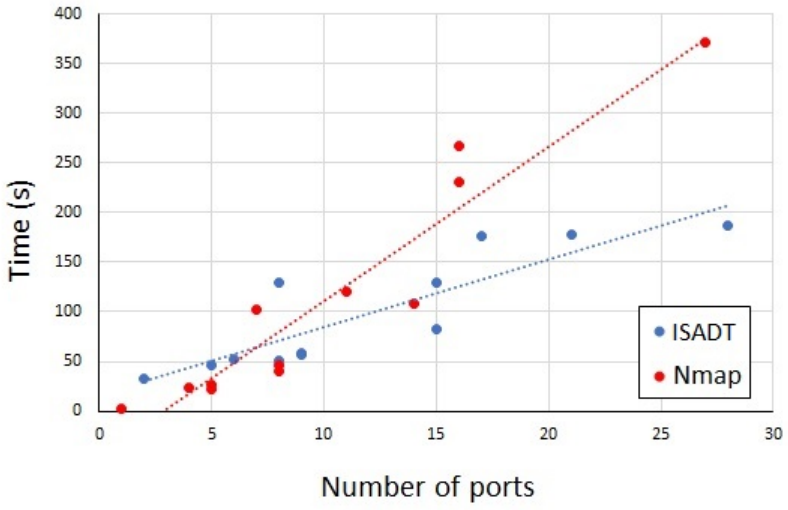


Figure 4: Time comparison.

that purpose, and it is validated in the other experiment). In the second setup, the system has been deployed in the real network of a research laboratory.

4.3.2. Experiment in a controlled environment

In this experiment two VLANs, whose network addresses are 192.168.11.0/24 (VLAN 1) and 10.8.0.0/24 (VLAN 2), have been configured. The infrastructure deployed in the virtual machines is shown in Figure 5. A server that is

connected to both VLANs acts as DN for both of networks. VLAN 1 only has connected the servers that act as MN and DN (and it will not be analysed), while VLAN 2 has connected four hosts and two servers (one of them running the DN). All the hosts and servers run Ubuntu 20.04 LTS and their network adapters have been configured as "internal network" so that they are isolated of the larger host network. To achieve this isolation, Vagrant makes use of the network segmentation concept, which consists on dividing a single network into multiple logical subnetworks, each of which can be configured with its own gateway and routes. Therefore, in internal network mode, the subnets do not use the host as gateway and they can only communicate with the other internal networks by means of static routes. This approach allows the experiment to be carried out in a completely separate and controlled environment.

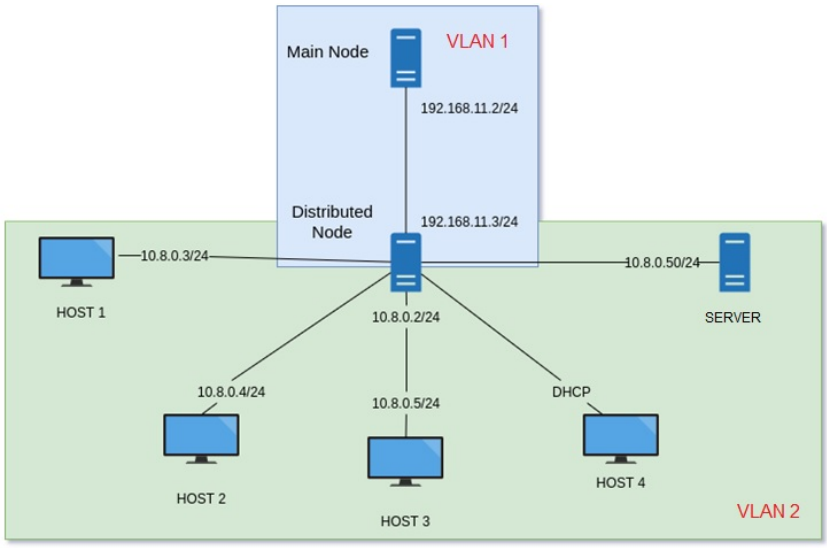


Figure 5: Infrastructure for the experiment in a controlled environment.

Regarding the synthetic traffic, all the packets have a size of $(100 + s)$ bytes, where s is a random component uniformly distributed in $[0, 100]$. Two different types of traffic have been generated:

- Low-intensity traffic: traffic is generated every $(500 + d_1)$ ms, where d_1 is a random component uniformly distributed in $[0, 200]$. Every time that traffic is generated, a burst of 1 to 3 packets is generated.

- High-intensity traffic: traffic is generated every $(5000+d_2)$ ms, where d_2 is a random component uniformly distributed in $[0, 2000]$. Every time that traffic is generated, a burst of 5 to 10 packets is generated.

Table 5 contains the traffic that is generated between each pair of devices and ports (all the source ports are random ports), including the traffic type and the time interval (in hours) in which the traffic is generated.

Table 5: Traffic details between each pair of devices and ports.

Source host	**Dest. host**	**Dest. port**	**Intensity**	**Time int. (h)**
HOST 1	SERVER	1001	High	0:00 - 2:00
HOST 1	SERVER	1002	Low	0:00 - 0:15
HOST 1	HOST 3	1003	Low	1:20 - 3:00
HOST 2	SERVER	2001	High	0:00 - 3:00
HOST 2	SERVER	2002	Low	0:00 - 2:00
HOST 2	HOST 3	2003	Low	1:20 - 3:00
HOST 3	SERVER	3001	High	2:40 - 3:00
HOST 3	SERVER	3002	Low	2:40 - 3:00
HOST 3	SERVER	3003	Low	2:40 - 3:00
HOST 4	SERVER	4001	High	1:00 - 2:30
HOST 4	SERVER	4002	Low	1:00 - 2:30
HOST 4	HOST 3	4003	Low	1:20 - 2:30

The experiment has a duration of 3 hours and the periodic scans are performed every 30 minutes and with the previous traffic it can be tested the discovery methods of periodic scans and traffic analysis (this last one because HOST 3 and HOST 4 begin to send packets to a new port between two scans).

To verify the functionality of the DHCP discovery method, HOST 4 is connected 50 minutes after the experiment starts with 4 open ports during 10 minutes. To verify the functionality of the ARP discovery method, HOST 3 is connected 1 hour and 20 minutes after the experiment starts with a static IP address (though it does not start to send any traffic until 2 hours and 40 minutes after starting the experiment).

4.3.3. Experiment in a real network

Regarding the experiment in a real network, two different VLANs of a research laboratory have been scanned: 192.168.1.0/24 (VLAN 1) and

192.168.50.0/24 (VLAN 2). The infrastructure if the virtual machines is shown in Figure 6. A server that is connected to both VLANs acts as MN and as DN for VLAN 2, while one server presnet in VLAN 1 acts as DN for that network.

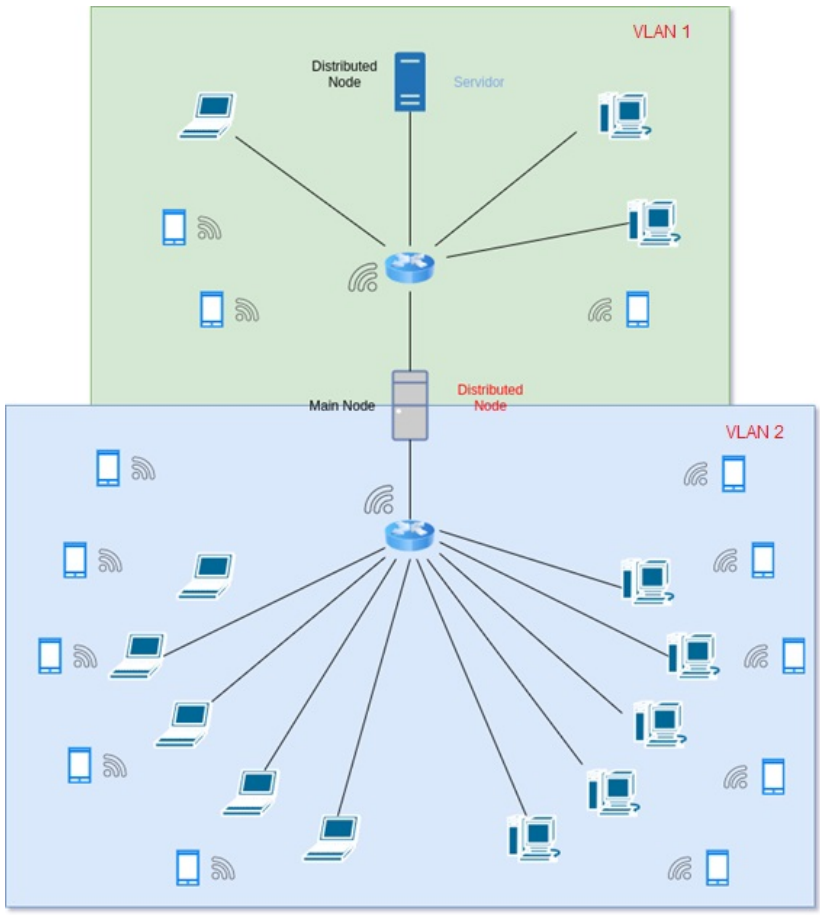


Figure 6: Infrastructure for the experiment in a real network.

In this case, the only knowledge about the elements in the network is that there is one server connected to the VLAN 1, whose IP address is 192.168.1.50, and that the IP addresses of the routers in VLAN 1 and VLAN 2 are 192.168.1.1 and 192.168.50.1, respectively.

Moreover, it is also known that a total of 9 users were connected to the networks during the experiment (divided by both of them). These users were asked to connect and disconnect their smartphones to the network during the

experiment. Besides those users, more of them might be working remotely, but using their desktops in the laboratory remotely. Moreover, they were asked to connect their smartphone to the network as well.

4.4. Validation of the proposed AI-based system

This subsection discusses the results of the two experiments that have been described before. Both of them were successfully completed.

4.4.1. Results of the experiment in a controlled environment

In the case of the experiment in the controlled environment, all the IP addresses of the devices and their ports being used were discovered and their status updated by the regular scans. Moreover, the DHCP discovery method detected HOST 4 after 59 minutes of execution (it was connected at minute 50), and the ARP discovery method detected HOST 3 after 1 hour and 21 minutes of execution (it was connected a minute before).

Regarding the traffic analysis method, hundreds of ports were discovered. They correspond with the random ports that were used to send packets to the server. Furthermore, notifications of traffic peaks were mostly detected on ports 1001, 2001, and 4001 of the server, which correspond with the ports that received high intensity traffic bursts.

4.4.2. Results of the experiment in a real network

With respect to the experiment in the real network, 4 snapshots with regular scans were received from VLAN 1 and 5 of them were received from VLAN 2. All the snapshots from VLAN 1 contained the same 6 IP addresses, with a number of total discovered ports ranging from 28 to 30 in each of them.

Regarding the snapshots received from VLAN 2, all of them contained the same 4 IP addresses. Moreover, the second, the fourth and the fifth snapshots contained an extra IP address too (the same one in all of them).

In the case of the DHCP discovery method, four IP addresses were detected in VLAN 1 and one was detected in VLAN 2. As these addresses do not appear in the snapshots of the regular scans, it seems that they were devices that were connected and, not much time later, disconnected.

Regarding the ARP discovery method, it detected 15 IP addresses, which were very likely devices that had already been connected before and they did not need to obtain a new IP address using DHCP. Furthermore, an average of 24 ports were discovered between each pair of scan snapshots with the method of traffic analysis.

Related to the AI-related techniques, a whole of 454, 19 and 5923 association rules between the ports were generated in each of the set of learnt rules, respectively. One of the rules that were learnt in the last set is shown in Equation 3. Finally, 8 traffic peak notifications were detected in VLAN 1 (most of them related to the router and the server) and 3 in VLAN 2.

$$\begin{aligned} \{(3394, \text{any})\} &\Rightarrow \{(7788, \text{any})\} \\ \{(1990, \text{any}), (18017, \text{any}), (53, \text{any}), (80, \text{any})\} &\Rightarrow \{(5473, \text{any})\} \end{aligned} \tag{3}$$

These results are congruent with the infrastructure that was use in the experiment, validating all the functionalities of the proposed intelligent architecture.

4.5. ISADT security analysis

Finally, in order to make it sure that the tool itself is vulnerability-free, the source code of the tool has been analysed with SonarQube, which is an open-source tool that inspects the code, detecting bugs and vulnerabilities, among others [22]. SonarQube has been used since it provides easy integration with other developing tools that have used to develop the system, such as Git or Jenkins, thus providing a continuous framework to maintain the quality of the code and preserving it from vulnerabilities. The preliminary results of this analysis showed that the code of the tool presented 15 bugs and 19 vulnerabilities.

Regarding the bugs, there were two types: i) not closing resources properly with a finally statement or a try-with-resources; and ii) objects that were used and their value might be null. All these bugs were properly addressed and, in a second round of code analysis, SonarQube reported no bugs.

Regarding the vulnerabilities, there were two types too: i) methods declared as public; and ii) persistent entities that should not be such (entities that were stored in a database, but they were not needed to be stored). Again these vulnerabilities were addressed and in the second round SonarQube did not report vulnerabilities.

Moreover, during the development of the tool the *log4j* vulnerability emerged [23], which was immediately solved (even before running the SonarQube analysis) by updating the library to its latest version. Finally, this tool will be used in any future development of the tool, continuing the effort in enhancing the the security of the developed tool.

5. Discussion

The objective of this work was to propose an intelligent tool for asset discovery in medium and large sized networks. Having a full inventory of network assets (such as devices, their open ports, services running in them, etc.) is very important for any organization. Moreover, in this kind of large networks, the devices and assets that are connected change constantly and, therefore, having an updated snapshot of the devices and the ports and services that they are using is crucial to know any potential open door to malicious software or users.

This inventory can be very helpful to control all the software and sources of vulnerabilities so that network administrators can take any preventive actions before an attack happens.

Moreover, the information discovered by ISADT can be used by other tools and methodologies, like the one presented in [24], to predict future incidents and identify new threat patterns in each single asset. This information can also be used to detect the cyber-security and the privacy risks in a device before an information exchange is done, as presented in [25].

Regarding the obstacles that may emerge when using ISADT, the main limitation that it can be foreseen is the restriction of the use of the Nmap tool in a network or that, being possible to use it, the devices are configured not to reply to these requests. In these situations, the system would not be able to detect the services, but by means of the other modules it could detect the IP addresses in use, as well as the ports being used.

Moreover, given the amount of workload that the DNs need to process, it is essential that they incorporate a Central Processing Unit (CPU) which is able to execute several parallel threads in order for it to work fluently and with no delay. For instance, in the experiment in a real network, the DNs had 8 cores, each of which being able to execute 2 simultaneous threads. Nevertheless, regarding the number of VLANs to be monitored, given that the MN has little workload, it can scale with no problem to any feasible number of VLANs in a large organization.

6. Conclusions

In this paper, a novel approach for the discovery of network assets has been presented. Even if it can be used for any computer network, as shown in the evaluation, it can be more advantageous in medium and large-size

networks. The presented tool implements a distributed architecture which combines active and passive detection methods for network asset discovery. Unlike some of the previous works for asset discovering, the proposed approach is not invasive, since the agents need to be deployed by the network administrator and they will always know where an agent is running.

ISADT contains four new innovative features with regard to the state-of-the-art works: it i) is able to find assets in isolated networks; ii) makes use of a TPM to secure all the information that has been previously found; iii) makes use of artificial intelligence techniques to detect common associations in the usage of ports by means of an Apriori algorithm; and iv) detects anomalous traffic peaks on specific ports in the network using a time series modelling approach.

The asset inventory obtained by ISADT can be further analysed to detect vulnerabilities associated to the services that the tool finds. Besides that, the network administrator can take the actions that they consider when an alarm about a strange combination of used ports or about an anomalous traffic peak is present.

These characteristics have been validated by setting up two experiments: one in a controlled environment and another in a real network.

Moreover, the tool has been evaluated and compared with Nmap, demonstrating that ISADT is able to detect the same ports as Nmap in a shorter period of time, which depends on the number of open ports in the network (the more open ports, the more advantageous that ISADT will be with respect to Nmap). This reduction of time is of a great importance since, if the scanning interval is short and the scan takes a long time, it might not finish until before the time-out for the next scan.

Finally, a qualitative comparison has been performed by specifying the features provided by ISADT that are not available in Nmap, like the capability of scanning isolated networks in a single execution environment, the use of a TPM to secure the information and the implementation of IA-based algorithms to analyse the collected information.

Acknowledgments

The research work presented in this article has been supported by the European Commission under the Horizon 2020 Programme, through funding of the CUREX project (G.A. n 826404). The work has been also supported

by EIT Digital Innovation Actions, through funding of the Obviews project (A.C: 20636-A2002).

References

[1] A. Muñoz, A. Farao, J. R. C. Correia, C. Xenakis, ICITPM: Integrity validation of software in iterative Continuous Integration through the use of Trusted Platform Module (TPM), in: European Symposium on Research in Computer Security, Springer, 2020, pp. 147–165.

[2] S. L. Kinney, Trusted platform module basics: using TPM in embedded systems, Elsevier, 2006.

[3] G. F. Lyon, Nmap network scanning: The official Nmap project guide to network discovery and security scanning, Insecure, 2009.

[4] Bromium Inc.: Hyper-connected web of profit emerges, as global cybercriminal revenues hot $1.5 trillion annually, online; accessed 5 September 2021.
URL `https://www.bromium.com/press-release/hyper-connected-web-of-profit-emerg`

[5] Risk in focus 2020: hot topics for internal auditors, Tech. rep., European Confederation of Institute of Internal Auditing (2019).

[6] G. Nguyen, S. Dlugolinsky, V. Tran, A. Lopez Garcia, Deep Learning for Proactive Network Monitoring and Security Protection, IEEE Access 8 (2020) 19696–19716.

[7] Y. Yuan, D. Lin, A. Mishra, S. Marwaha, R. Alur, B. T. Loo, Quantitative Network Monitoring with NetQRE, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 99–112.

[8] H. J. Abdelnur, R. State, O. Festor, Advanced Network Fingerprinting, in: Recent Advances in Intrusion Detection, Springer Berlin Heidelberg, 2008, pp. 372–389.

[9] J. C. Barroux, Network asset survey tool for gathering data about node equipment, US Patent 6,220,768 (Apr 2001).

[10] Z. Trabelsi, W. El-Hajj, On investigating ARP spoofing security solutions, International Journal of Internet Protocol Technology 5 (1) (2010) 92.

[11] M. Laštovička, M. Husák, L. Sadlek, Network Monitoring and Enumerating Vulnerabilities in Large Heterogeneous Networks, in: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1–6.

[12] P. Calderon, Nmap: network exploration and security auditing cookbook, Packt Publishing Ltd, 2017.

[13] C. Yuan, J. Du, M. Yue, T. Ma, The Design of Large Scale IP Address and Port Scanning Tool, Sensors 20 (16) (2020) 4423.

[14] I. A. I. Diyeb, A. Saif, N. A. Al-Shaibany, Ethical network surveillance using packet sniffing tools: A comparative study, International Journal of Computer Network and Information Security 10 (7) (2018) 12.

[15] M. Brattstrom, P. Morreale, Scalable agentless cloud network monitoring, in: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), IEEE, 2017, pp. 171–176.

[16] J. Navon, F. Fernandez, The Essence of REST Architectural Style, Springer New York, 2011, pp. 21–33.

[17] T. Rooney, Dynamic Host Configuration Protocol (DHCP), 2010, pp. 53–68.

[18] J. Postel, RFC0792: Internet Control Message Protocol, RFC Editor, 1981.

[19] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the 1993 ACM SIGMOD international conference on Management of data, 1993, pp. 207–216.

[20] W. Arthur, D. Challener, K. Goldman, Platform Configuration Registers, Apress, Berkeley, CA, 2015, pp. 151–161.

[21] M. Hashimoto, Vagrant: up and running: create and manage virtualized development environments, O'Reilly Media, 2013.

[22] V. Lenarduzzi, A. Sillitti, D. Taibi, A survey on code analysis tools for software maintenance prediction, in: International Conference in Software Engineering for Defence Applications, Springer, 2018, pp. 165–175.

[23] A. Jones, Security posture: A systematic review of cyber threats and proactive security.

[24] C. Bellas, A. Naskos, G. Kougka, G. Vlahavas, A. Gounaris, A. Vakali, A. Papadopoulos, E. Biliri, N. Bountouni, G. G. Granadillo, A methodology for runtime detection and extraction of threat patterns, SN Computer Science 1 (2020) 1–13.

[25] G. Gonzalez-Granadillo, S. A. Menesidou, D. Papamartzivanos, R. Romeu, D. Navarro-Llobet, C. Okoh, S. Nifakos, C. Xenakis, E. Panaousis, Automated cyber and privacy risk management toolkit, Sensors 21 (16) (2021) 5493.