# NETSTALDI: A Modular Distributed Architecture for Incremental Network Discovery

Matteo Paier[1,2,*], Mattia Pizzolitto[1], Gian Luca Foresti[1] and Marino Miculan[1,3]

[1]University of Udine - Dept. of Mathematics, Computer Science and Physics, Italy
[2]IMT Alti Studi Lucca, Italy
[3]Ca' Foscari University of Venice - Dept. of Environmental Sciences, Informatics and Statistics, Italy

**Abstract**

Maintaining a comprehensive understanding of a network's connected devices is fundamental for robust security. This knowledge is crucial for tasks like vulnerability assessments, identifying potential attack surfaces, and network mapping. In this paper, we propose NETSTALDI, a distributed architecture specifically designed for incremental and secure network discovery. Our architecture leverages non-intrusive network scanning techniques based on standard TCP/IP protocols, eliminating the need for monitoring agents on individual devices. This approach prioritizes scalability, modularity, and resilience, making it well-suited to handle large and dynamic network environments.

We have implemented a prototype system based on this architecture, utilizing established open source tools. This system has been successfully tested on a large, real-world network. The results are promising: the tool efficiently scans the entire network within a few hours, and the intuitive GUI allows administrators to interactively explore the generated network map to identify and address potential vulnerabilities and misconfigurations.

**Keywords**

Network security, Network discovery, Network mapping, Vulnerability assessment

## 1. Introduction

*Network discovery*, the practice of finding and analysing connected devices in a network, is crucial for various management tasks. It plays a vital role in security assessments, network mapping, and administration [1, 2]. Through systematic network probing and analysis of responses, network administrators gain invaluable insights into the network landscape. This includes understanding the connected devices, the services they offer, and their potential vulnerabilities. Having a complete and updated inventory of devices is essential for identifying potential attack surfaces that malicious actors might exploit. Additionally, mapping running services helps uncover unauthorized or unnecessary applications, which can increase the attack surface and introduce misconfigurations. This visibility empowers IT teams to manage their network health, optimizing performance for smoother operations, strengthening security postures against cyberthreats, and ensuring compliance with industry regulations. Tools automatizing device discovery and vulnerability assessments free up valuable time and resources.

Network discovery can be achieved using specialized software tools, which act as cartographers, providing a comprehensive view of the network landscape, from identifying connected devices and services to revealing potential security vulnerabilities and performance bottlenecks.

Often these tools leverage specific protocols like the Simple Network Management Protocol. While it provides a detailed view of network devices (e.g., routers, switches), it may not comprehensively cover endpoints like personal computers and servers. Additionally, it necessitates the installation of specific client software on each monitored device. Alternatively, network scanning can be achieved by sending carefully crafted data packets to specific IP addresses and ports. Analyzing the responses and device behaviours helps build a detailed picture of the network's layout and configuration. This agentless

*ITASEC 2024: The Italian Conference on CyberSecurity, April 08–12, 2024, Salerno, IT*
*Corresponding author.
matteo.paier@imtlucca.it (M. Paier); pizzolitto.mattia@spes.uniud.it (M. Pizzolitto); gianluca.foresti@uniud.it (G. L. Foresti); marino.miculan@uniud.it (M. Miculan)
0009-0000-7588-7169 (M. Paier); 0000-0002-8425-6892 (G. L. Foresti); 0000-0003-0755-3444 (M. Miculan)

approach, employed by tools like Nmap [3], is less intrusive as it does not require software installation on devices. However, the visibility gained may be incomplete and dependent on the scanner's perspective, that is, its position in the network to analyse [4]. Moreover, most open source tools do not allow for an incremental building of the network map, that is, they lack support for "merging" different scans of the network into a coherent picture.

To overcome these issues, in this paper we propose a distributed architecture for incremental network discovery. Our architecture, called NETSTALDI, leverages network scanning techniques based on standard TCP/IP protocols, eliminating the need for intrusive monitoring tools on individual devices. This design prioritizes scalability, modularity, and resilience. We have implemented a prototype system based on this architecture, utilizing established tools like Nmap. This system has been successfully tested on a large, real-world network exceeding 55,000 hosts. The results are promising: the tool efficiently scans the entire network within a few hours, and the intuitive GUI allows administrators to interactively explore the generated network map, looking for vulnerabilities and misconfigurations.

The rest of the paper is structured as follows. In Section 2 we recall the basic techniques for network discovery and scanning. We present our distributed architecture in Section 3, detailing design requirements and choices. In Section 4 we describe the implementation of a prototype, according to the proposed architecture, and present some results on a real-word large network. Finally, conclusions and direction for future work are in Section 5.

## 2. Background

Due the importance of the problem, many tools and techniques have been developed for network mapping and scanning [5]. Among these tools, those utilizing Simple Network Management Protocol (SNMP) to gather information about the network topology stand out for their ability to delve deep into the inner workings of routers and switches [6, 7].

On one hand, SNMP offers unparalleled clarity: it is able to obtain detailed configuration data, performance metrics, and even granular statistics on individual interfaces and protocols. This comprehensive view allows network administrators to easily troubleshoot issues and optimize network performance. SNMP embraces automation, facilitating the data gathering process. Its widespread support across vendors and models ensures consistent mapping, simplifying management across diverse hardware ecosystems. Additionally, SNMP provides fine-grained control, allowing administrators to focus on specific metrics or devices.

On the other hand, SNMP-based approaches can also present some potential shadows. Security can be a major concern, especially with older versions of the protocol. These lack robust measures, leaving them vulnerable to eavesdropping and manipulation, potentially exposing sensitive network information. Additionally, the configuration of SNMP is a complex process that necessitates technical expertise and familiarity with *Management Information Bases*. This initial learning curve can pose a challenge for some administrators, especially in small to medium size enterprises. Moreover, while the insights SNMP offers are often very valuable, it might not provide a complete picture, as some devices or protocols might not be SNMP-compliant. In fact, in order to get a comprehensive view of the network we have to install an SNMP client (i.e., a miniature agent that gathers vital system information like performance metrics, hardware details, and error logs) on each device we want to observe and manage. This intrusive approach may be not always feasible, leaving blind spots in the network map.

An alternative approach is to rely on the protocols that are implemented by default on any node of the network, that is ICMP packets, UDP datagrams, TCP handshakes, or even raw IP packets. Among others, Nmap is one of the best-known free and open source port scanner [3]. It uses raw IP packets to detect available hosts on a network, the services they are offering (including application name and version), what operating systems they run, and other network characteristics like what type of packet filters and firewalls are in use. Although it was primarily designed to rapidly scan large networks, it is also effective against single hosts.

Nmap provides advanced support for host discovery, i.e. by reducing a list of IP ranges into a list of

active and possibly interesting hosts to analyse. There is also support for very efficient port scanning with granular definition of the states that a port can be in: `open`, `closed`, `filtered`, `unfiltered`, `open|filtered`, `closed|filtered`. Usually the most interesting (from a security perspective) state is the `open` one. Nmap can detect the operating system in use on the target device by using TCP/IP stack fingerprinting and comparing the results to its database of crowd-sourced samples. Finally, Nmap supports the execution of custom Lua scanning scripts using the Nmap API (Nmap Scripting Engine). These scripts are especially useful for both searching and exploiting vulnerabilities.

However, Nmap is a CLI tool, which may pose some difficulties for users not acquainted with this interaction paradigm. To this end, Nmap offers an official GUI, called Zenmap, which has been designed to make Nmap easy for beginners while providing advanced features for experienced users [3]. Zenmap simplifies repetitive tasks by letting you save frequently used scans as profiles and providing a user-friendly interface to build Nmap commands interactively. Moreover, it automatically stores recent scans in a central database, enabling convenient review of past results and comparisons between scans to analyse network changes. While Zenmap is a great tool for smaller networks, its usability suffers on networks exceeding around fifty devices. The interface becomes less responsive and may lag, and the output can be overwhelming and difficult to interpret.

Moreover, Nmap, by design, cannot offer the same level of comprehensive network visibility compared to SNMP-based tools. Finally, none of these tools offer *incremental* network discovery, i.e., the possibility of building a more extensive view of the network by combining the outputs of several scans, possibly executed in different moments and points of the network.

## 3. Netstaldi Architecture

In this section we present the architectural design of Netstaldi, our system for distributed incremental network discovery.

### 3.1. Requirements

The system's functional requirements can be broadly categorized into three key areas: scan functionality, data management, and user interface.

From the scanning functionality point of view, the system shall identify and inventory all active devices connected to the network, including their IP addresses, MAC addresses, and operating systems (whenever possible). Moreover the system shall identify and list the services running on each discovered device, including their port numbers and protocols used. Finally, the system shall build a representation of the network topology, illustrating the connections between devices and subnets.

The data shall be gather efficiently from the scan probes and stored in a central knowledge base. The system shall persistently store all collected scan data, including historical scan results, for future reference and analysis. Also, the system shall allow users to export scan data for further analysis or integration with other tools.

The user interface shall provide three main features: an interface for users to initiate scans on specific network segments or the entire network; a user-friendly interface for visualizing the network topology, including detailed information about discovered devices and services; a mean to filter and search the collected data based on various criteria (e.g., IP address, device type, service name) for efficient navigation and analysis.

Aside from the functional requirements, the non-functional requirements are less evident.

The system's design must prioritize scalability, ensuring it can seamlessly adapt to growing network sizes and complexities. The use of lightweight scan probes that can be easily deployed throughout the network allows for expansion without significant overhead. This also allows for the separation of scanning from data processing and analysis, offering several advantages: reducing the load on individual scanning devices, preventing performance bottlenecks, and enhancing scalability. Moreover, centralized computation provides a single point of control and coordination for complex analysis tasks, improving operational efficiency.

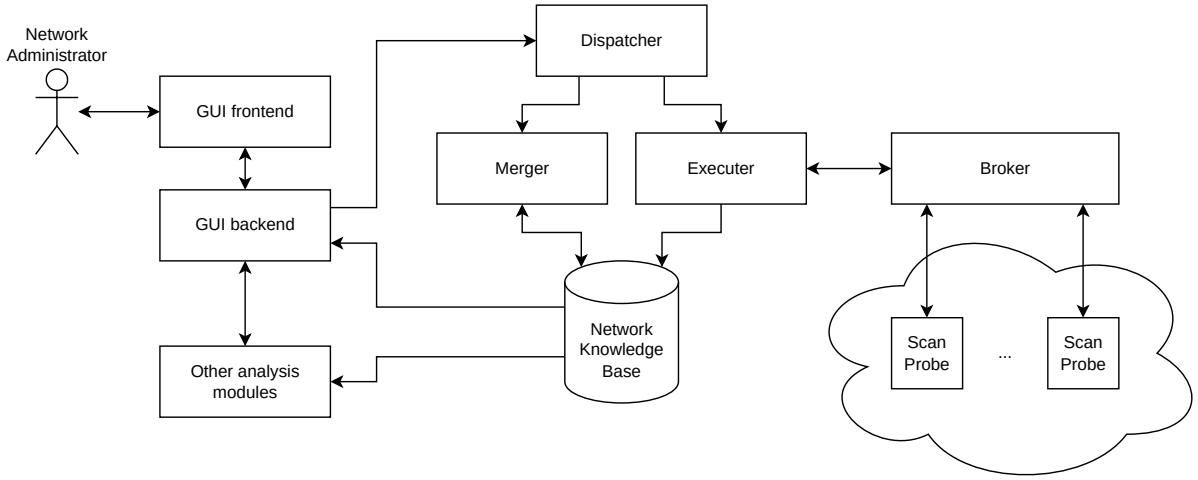



**Figure 1:** NETSTALDI's high level architecture.

The architecture must be modular, where each component has clearly defined interfaces and responsibilities. This decoupled design fosters flexibility, as individual modules can be upgraded, replaced, or fine-tuned independently without impacting the overall system. This modularity simplifies maintenance, promoting agility in addressing evolving requirements. The system's open architecture allows for the seamless integration of additional modules or components as needs evolve. For example, if specialized analysis tools become necessary, they can be easily added due to the system's well-defined communication protocols and data formats. This adaptability ensures the system can grow along with future network monitoring and management requirements.

The system's reliance on network-based scanning eliminates the need for installing and managing software agents on individual end devices. This simplifies deployment, reduces the attack surface of the network, and minimizes the risk of compatibility issues across a diverse set of end devices. Moreover, this architecture facilitates incremental deployment: new scan probes can be added strategically to extend coverage without disrupting existing operations.

## 3.2. Architecture Design

Following the aforementioned requirements, we propose a design for the network mapping system (Figure 1) consisting of six primary modules with different tasks:

**Scan Probes** These are lightweight modules deployed across the network to gather data. They collect information from various segments, providing (possibly partial and different) views of the network infrastructure.

**Dispatcher** This module is responsible for managing the execution of scans. It employs its internal policies to regulate the data collection and processing workflow, ensuring efficient and coordinated operation. This controlled execution helps maintain consistency and avoid conflicts in the gathered network information.

**Executer** This component manages the scan probes. It receives scan requests from the dispatcher, transmits them to the relevant probes (via a suitable message broker), and coordinates their data collection efforts. The outcomes of the scans are stored in a shared *Network Knowledge Base*.

**Network Knowledge Base (NKB)** This component serves as a central repository for all scan data and the resulting network representation. It stores the information persistently, allowing for historical analysis and future reference.

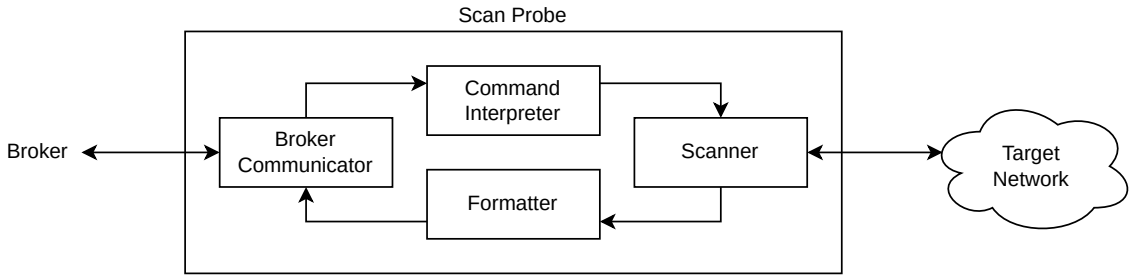



**Figure 2:** Logical architecture of the Scan Probe module.

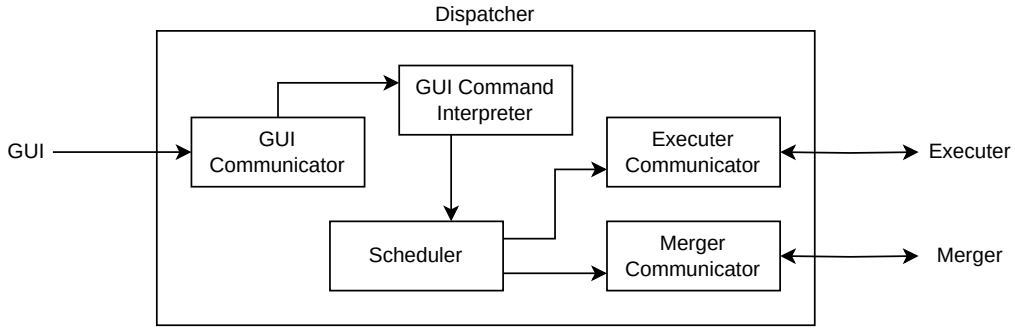



**Figure 3:** Logical architecture of the Dispatcher module.

**Merger** This module integrates the collected data from the probes, stored in the NKB, merging and reconciling it into a consistent representation of the entire network. It harmonizes potentially conflicting information to create a unified picture.

**GUI** The graphical user interface provides a platform for network administrators to initiate and manage scans. It displays the results, allowing users to visualize and analyse the network topology and configuration details.

We describe next some key components in more detail.

The system leverages specialized tools called Scan Probes to gather information about connected devices. These probes are comprised of several key components working together (Figure 2). One component, the Broker Communicator, acts as a communication bridge to the broader system, ensuring the seamless exchange of data and instructions. Another, the Command Interpreter, translates received commands into specific instructions for the probe's internal functionalities. The core element, the Scanner, actively sends out requests to network devices and collects their responses, forming the raw data for further analysis. Finally, the Formatter transforms this data into a standardized format, making it easier for the system to process and analyse the gathered information.

The system's central coordinator is the Dispatcher (Figure 3). It acts as the main controller of the operation, receiving commands from the user interface (GUI) through a dedicated communication module. These commands are then interpreted by the GUI Command Interpreter and passed to the Scheduler, that enforces the execution of scans according to given policies, which might involve prioritizing scans based on administrator privileges or restricting access to specific probes in sensitive network areas. Once the plan is established, the Scheduler sends instructions to the Executer and the Merger via specialized communication modules.

The Executer (Figure 4) acts on the Dispatcher's directives (received through the Dispatcher Communicator module), coordinating the actual scanning processes by sending requests to the scan probes via its Broker Communicator Module. The scan results are received by the Collector module, that stores them in the Knowledge Base, ensuring persistency of the collected data. The Supervisor module, on the other hand, is responsible for keeping track of the active scans and ensuring their successful completion.

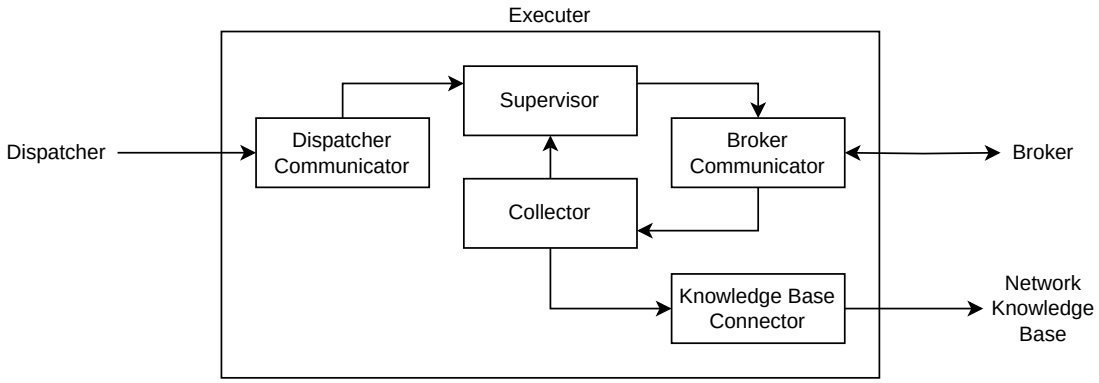



**Figure 4:** Logical architecture of the Executer module.

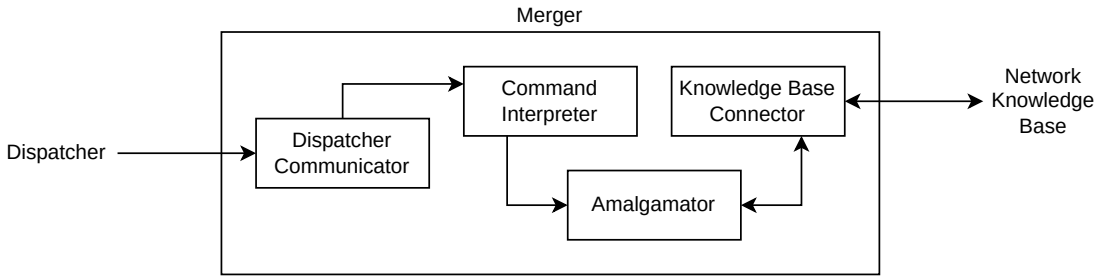



**Figure 5:** Logical architecture of the Merger module.

Finally, the Merger (Figure 5) plays a crucial role in collecting and harmonizing the data gathered by the probes, ultimately presenting a comprehensive picture of the scanned network. The need for this operation arises from the fact that due to the limited communication and incomplete information about non-immediately connected nodes each probe can only build a partial view of the whole network. To achieve this goal, the Merger receives merging commands from the Dispatcher, and instructs the Amalgamator accordingly. This module takes completed scans from the Knowledge Base and assembles them into a coherent view of the entire network.

Amalgamation of these partial views obtained by discovery tools into a coherent representation can be solved using labelled bipartite directed graphs representing the communications between hosts, interfaces, and networks (*round-trip graphs*), as described in [8]. The amalgamation operation is thus a compositional and "structure preserving" merge operation that is defined on these kind of graphs.

## 4. Prototype Implementation and Experimental Tests

In this section we present a prototype implementation of the architecture presented in Section 3.

As a starting point we adopted a single-probe approach to streamline the development process. This allowed us to focus on the core functionalities of network scanning, data parsing, and storage without the added complexity of multi-probe coordination. However, it is important to note that we have already laid the theoretical groundwork for the *Merger* component in [8]: this will allow future scalability enhancements, enabling us to distribute scanning tasks across multiple probes for greater coverage and efficiency in large-scale network environments.

The network probes within the system leverage MQTT as the communication backbone [9]. MQTT's lightweight pub/sub model makes it ideal for data exchange in distributed systems. Each probe subscribes to a command topic on the central MQTT broker, awaiting instructions. The actual network scan is executed by Nmap, which serves as the main worker for performing various network operations (host

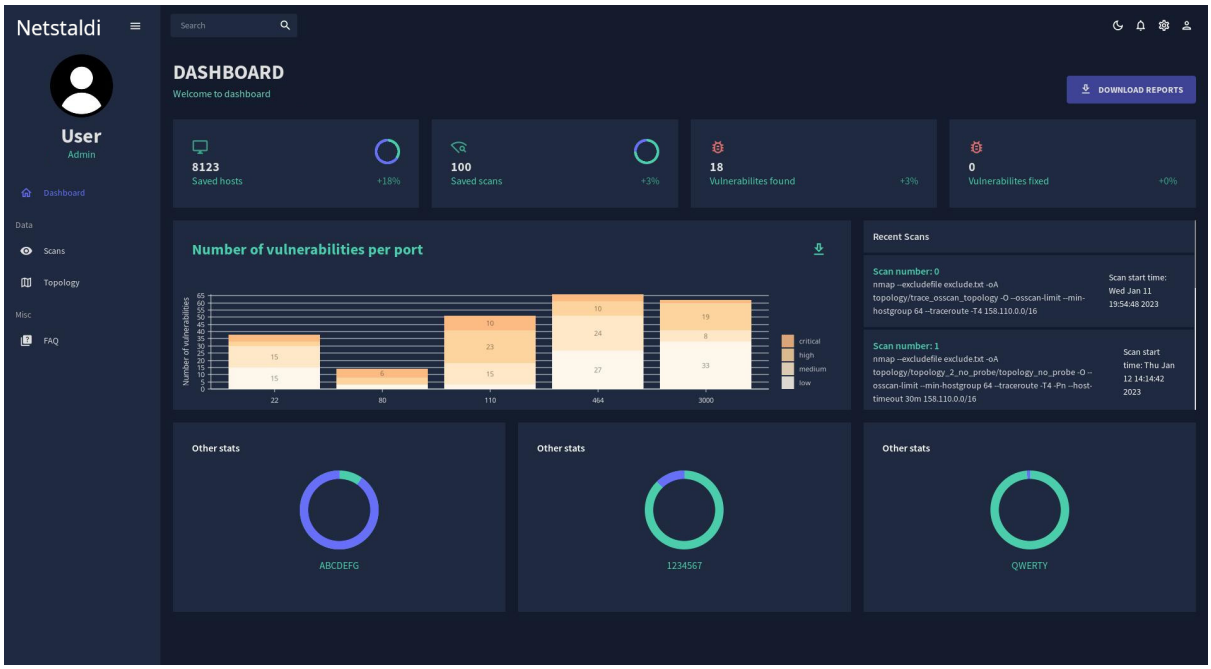



**Figure 6:** NETSTALDI's GUI dashboard.

discovery, port scanning, etc.). Communication, orchestration of Nmap operations, and the processing of Nmap outputs are done by a specific program written in Rust, which provides the necessary speed and efficiency. We decided for Mosquitto as the MQTT broker due to its support for QoS level 2, guaranteeing exactly-once delivery of messages between the network probes and the central system.

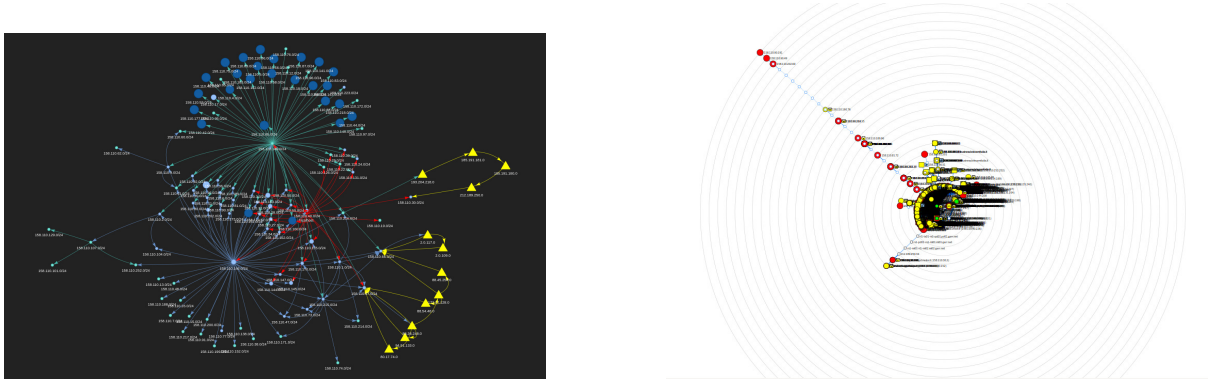
To deliver a modern and intuitive user interface, we opted for the Tauri framework [10]. Tauri enables the development of cross-platform desktop applications using Rust as the backend and web technologies for the frontend. We chose React as our primary frontend library, taking advantage of its component-based architecture for building dynamic and responsive user interfaces. This combination allows us to create a performant, visually appealing, and user-friendly GUI, providing a seamless experience for network administrators (Figure 6).

Also Dispatcher and Executer are implemented in Rust for its performance, concurrency management, and safety features that make it an ideal choice for backend services. The Dispatcher acts as the central coordinator, responsible for receiving and interpreting commands from the GUI, and dispatching instructions to the Executer. To facilitate automated scanning, the Dispatcher supports for `cron` jobs: they interact with the scheduler, allowing administrators to define scan parameters and specify the exact times when scans should be initiated, ensuring regular network monitoring.

To implement our Network Knowledge Base, we have chosen MongoDB, primarily due to its native support for semi-structured data. Network scans can yield varying data formats depending on the scan type and the discovered devices. MongoDB's flexible, document-oriented approach allows us to store these different data structures without needing to adhere to a rigid schema. This adaptability is crucial as the system accommodates new scan types, probes, and different devices, ensuring seamless information storage and retrieval.

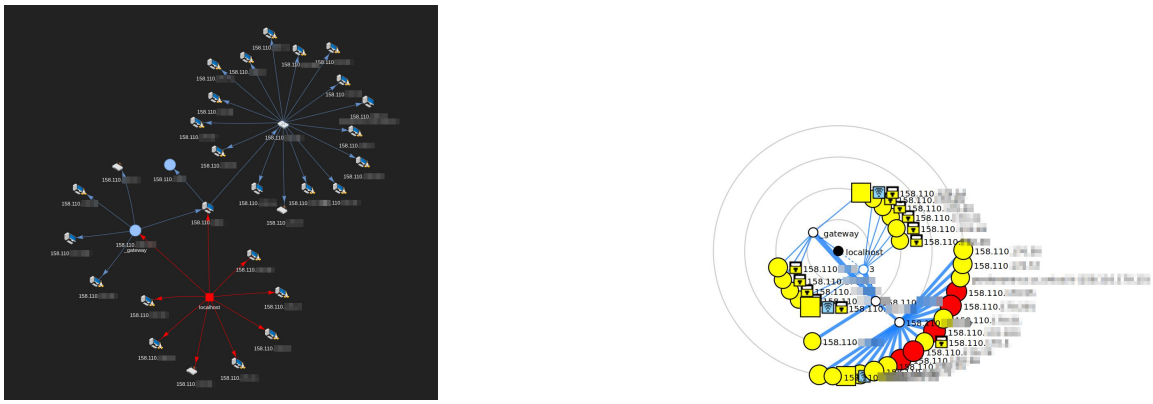
**Validation.** To test the effectiveness and user-friendliness of the proposed system, we opted for a real-world validation approach. This involved running extensive scans on a large real network, namely the network of the University of Udine (which corresponds to 158.110.0.0/16), at different accuracy levels, mirroring end-user interactions and exploring the system's capabilities.

The probing was executed from a whitelisted Linux virtual machine situated within the network itself, ensuring a secure and controlled testing environment. This approach allowed us to gather valuable insights about the system's performance and flexibility when handling real-world network scenarios.



(a) The network rendered in NETSTALDI; (b) the same network rendered with Zenmap.

**Figure 7:** The entire network from the University of Udine rendered with different tools.




(a) The network rendered in NETSTALDI; (b) the same network rendered with Zenmap.

**Figure 8:** A subnet from the University of Udine rendered with different tools.

While a comprehensive scan in this scenario can result in outputs exceeding hundreds of megabytes and take several hours to complete, the system remarkably delivers the results to the network administrator's device for visualization within seconds.

To demonstrate the system's capabilities, a large-scale scan was conducted, encompassing 55,256 hosts over 48 hours. This scan skipped the initial ping check, revealing valuable information about previously unresponsive devices. The analysis of the data revealed that the longest path from the probe to a host was 11 hops, while the average path length was 2.84 hops. This extensive scan generated a substantial XML output, exceeding 2.3 million lines and 150 MB in size. This scan does not include automatic vulnerability discovery, which can be obtained by executing Nmap custom scripts.

As we can see in Figures 7 and 8 our system's outputs offer enhanced clarity compared to Zenmap, particularly in leveraging visual elements like distinct icons and colours; e.g., we highlight in yellow external IPs and paths, pointing out possible security and privacy threats. This improved presentation can significantly aid in troubleshooting network issues and identifying potential vulnerabilities. Moreover, our system promotes efficient and interactive network exploration through its intuitive interface. By simply hovering over the reconstructed network graph, administrators can access important details about individual hosts, including hostname, open ports, and operating system.

## 5. Conclusions

In this work we have presented NETSTALDI, a distributed architecture for incremental network discovery. Our architecture relies on network scanning techniques based on standard protocols from the TCP/IP

stack, and hence it does not need to install monitoring tools on end devices. The architecture is scalable, modular, and resilient. We have implemented a prototype system following this specification, leveraging commonly used tools such as Nmap. We have tested this system on a large, real-world network (with more than 55,000 hosts), yielding promising results: the tool is capable to scan the whole network in few hours, and the GUI allows an administrator to explore interactively the resulting map. As an example, using this tool we have discovered and successfully patched two machines vulnerable to the VMware ESXi CVE-2021-21974.

**Future Work.** As a first development of this work, we intend to complete the Merger module, implementing the amalgamation algorithm presented in [8]. Our architecture is designed to be modular and extensible, allowing us to integrate additional analysis modules for enhanced network monitoring capabilities. One particularly interesting application is the development of a module that analyses, even with incomplete network data, its compliance with predefined security policies. This is achieved by leveraging the expressive power of spatio-temporal logics to specify these policies. For instance, a spatio-temporal security rule to prevent malware spread might state: "once a device connects to a low-security network (e.g., DMZ), it cannot subsequently connect to a higher-security network." Viewing the network as divided into distinct segments suggests a hierarchical structure; to effectively represent this, we propose utilizing *bigraphs*, which are data structures similar to graphs, but specifically designed to model nested network relationships and connections between different nodes [11, 12, 13].

Another future direction involves integrating an SNMP-based approach into our tool. This hybrid architecture would offer a comprehensive view of network information by combining both methods; e.g., it could reveal network devices invisible to network scans, such as redundant switches and routers.

## Acknowledgments

This research has been partially supported by the Department Strategic Project of the University of Udine within the Project on Artificial Intelligence (2020-25), and the project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NGEU. We thank the DISO of the University of Udine for their support in the use of the prototype tool on the real network.

## References

[1] H. Holm, T. Sommestad, J. Almroth, M. Persson, A quantitative evaluation of vulnerability scanning, Information Management & Computer Security 19 (2011) 231–247.

[2] M. Anbar, A. Manasrah, S. Ramadass, A. Altaher, A. Aljmmal, A. Almomani, Investigating study on network scanning techniques, International Journal of Digital Content Technology and its Applications 7 (2013) 312.

[3] G. F. Lyon, Nmap: The network mapper, 2023. Available at https://nmap.org/.

[4] R. A. Alhanani, J. Abouchabaka, An overview of different techniques and algorithms for network topology discovery, in: 2014 Second World Conference on Complex Systems (WCCS), IEEE, 2014, pp. 530–535.

[5] A. De Montigny-Leboeuf, F. Massicotte, Passive network discovery for real time situation awareness, in: Proceedings of the The RTO Information Systems Technology Panel (IST) Symposium on Adaptive Defence in Unclassified Networks, 2004, pp. 288–300.

[6] SolarWinds, Network topology mapper, 2024. Available at https://www.solarwinds.com/network-topology-mapper.

[7] Netdisco Project, Netdisco, 2024. Available at https://netdisco.org/.

[8] M. Miculan, M. Paier, Assembling coherent network topologies using round-trip graphs (short paper), in: G. Castiglione, M. Sciortino (Eds.), Proceedings of the 24th Italian Conference on Theoretical Computer Science (ICTCS'23), volume 3587 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 110–115. URL: https://ceur-ws.org/Vol-3587/6692.pdf.

[9] MQTT Technical Committee, MQTT, 2023. Available at https://docs.oasis-open.org/mqtt/mqtt/.
[10] Tauri, Tauri architecture, 2023. Available at https://tauri.app/v1/references/architecture/.
[11] F. Burco, M. Miculan, M. Peressotti, Towards a formal model for composable container systems, in: C. Hung, T. Cerný, D. Shin, A. Bechini (Eds.), SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, ACM, 2020, pp. 173–175. doi:10.1145/3341105.3374121.
[12] G. Bacci, D. Grohmann, M. Miculan, Bigraphical models for protein and membrane interactions, in: G. Ciobanu (Ed.), Proceedings Third Workshop on Membrane Computing and Biologically Inspired Process Calculi, MeCBIC 2009, volume 11 of *EPTCS*, 2009, pp. 3–18. doi:10.4204/EPTCS.11.1.
[13] D. Grohmann, M. Miculan, Directed bigraphs, in: M. Fiore (Ed.), Proceedings of the 23rd Conference on the Mathematical Foundations of Programming Semantics, MFPS 2007, volume 173 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2007, pp. 121–137. doi:10.1016/J.ENTCS.2007.02.031.