# ptnetinspector: A Practical IPv6 Scanner for Network Reconnaissance

1st Viet Anh Phan
*Department of Telecommunications*
*Brno University of Technology*
616 00 Brno, Czech Republic
243760@vut.cz

2nd Jan Jeřábek
*Department of Telecommunications*
*Brno University of Technology*
616 00 Brno, Czech Republic
jan.jerabek@vut.cz

3rd Roman Kümmel
*CEO of HACKER Consulting Ltd.*
Czech Republic
r.kummel@hacker-consulting.cz

***Abstract*—The rapid expansion of IPv6 has introduced new challenges in network reconnaissance and vulnerability assessment. In this paper, we present ptnetinspector, a dedicated IPv6 scanner for local environments. While existing IPv6 tools rely on traditional methods that often miss active nodes or specific address types, ptnetinspector employs modern scanning strategies to uncover more IPv6 addresses (including SLAAC, DHCPv6, and manual assignments) and accurately map node roles. Our evaluation shows that ptnetinspector not only discovers additional active addresses and hosts but also identifies vulnerabilities unique to IPv6 protocol implementations, providing a robust foundation for precise network reconnaissance and proactive security in modern IPv6 environments.**

***Index Terms*—IPv6, Reconnaissance, Vulnerability, Scanning, Network Security**

## I. Introduction

The transition to IPv6 expands the address space while introducing distinct addressing methods like SLAAC, DHCPv6, and manual configuration, alongside protocols such as Neighbor Discovery Protocol (NDP) and Extension Headers. Hosts in IPv6 environments behave differently than in IPv4, rendering traditional scanning techniques ineffective due to sparse address distribution, multicast reliance, and NDP-driven address resolution. IPv6 also introduces new vulnerabilities, including insecure SLAAC implementations vulnerable to rogue router advertisements, NDP spoofing, extension header manipulation for evasion, and risks in dual-stack systems from misconfigured transition mechanisms. These changes demand specialized tools and strategies for accurate network mapping, host discovery, and vulnerability assessment in IPv6 environments.

Although tools like Nmap [2] have been adapted to IPv6 (using the `-6` flag), they generally depend on pre-defined target lists and can overlook active IPv6 addresses on hosts with multiple assignments. Similarly, the THC-IPv6 toolkit [3] and fi6s [4] are geared toward specific attack vectors or rapid scanning, but they do not offer a comprehensive solution for complete network reconnaissance. Furthermore, while the SI6 Networks IPv6 Toolkit [5] excels in diagnosing connectivity and configuration issues, it lacks integrated mechanisms to enumerate every IPv6 address advertised by a host.

This work is supported by Ministry of the Interior of the Czech Republic under Grant VK01030019.

To overcome these shortcomings, we introduce **ptnetinspector**[1], a dedicated IPv6 scanner designed specifically for local network penetration testing. This tool is part of **Penterep**[2] [1], a platform for complex penetration testing. Using an innovative approach, **ptnetinspector** can systematically discover all IPv6 addresses associated with a host, accurately map network topology and node roles, perform detailed service detection and perform targeted vulnerability assessments. This unified methodology provides security professionals with a holistic tool that exceeds the capabilities of existing solutions.

## II. Related work

### A. Nmap (IPv6 Mode)

Nmap's `-6` mode uses IPv6-specific scripts (e.g., `targets-ipv6-multicast-slaac`, `-invalid-dst`) to send ICMPv6 Router Solicitation, Echo Requests, and malformed packets to multicast addresses (e.g., `ff02::1`), triggering SLAAC based address disclosures, host activity confirmation, and vulnerability exposure via error messages. Dictionary-based probes and Multicast Listener Discovery map nodes and multicast groups, enabling detailed IPv6 topology analysis.

### B. THC-IPv6 Toolkit

The `alive6` tool actively probes multicast addresses (`ff02::1`) with ICMPv6 Echo Requests and Neighbor Solicitations to enumerate active hosts. Passive detection via `detect-new-ip6` identifies real-time node activation through unsolicited Router/Neighbor Advertisements, mapping topology and detecting IPv6 configuration flaws.

### C. SI6 Networks' IPv6 Toolkit

`scan6` combines ICMPv6 Echo Requests, Neighbor Solicitations, and multicast queries to elicit host responses. Analysis of Neighbor Advertisements and Router Solicitations maps network structure, identifies misconfigurations, and exposes vulnerabilities in IPv6 protocol implementations.

[1]https://github.com/Penterep/ptnetinspector.git
[2]https://www.penterep.com/

### D. Fi6s Scanner

fi6s rapidly discovers hosts via ICMPv6 Echo Requests, Neighbor Solicitations, and UDP probes to high ports, exploiting SLAAC patterns. Targets multicast (`ff02::1`) and link-local addresses, minimizing traffic while bypassing IPv6's address-space scalability challenges.

### E. Academic Research on IPv6 Network Reconnaissance

Recent studies have advanced the state-of-the-art in IPv6 reconnaissance by addressing not only address discovery but also topology mapping. For instance, the 6Tree framework (2020) leverages entropy-based techniques to dynamically uncover active hosts within vast IPv6 subnets, demonstrating significant improvements in efficiency over traditional methods [6]. In parallel, approaches such as 6GCVAE (2020) employ deep learning models to generate candidate target addresses, thereby enhancing the precision of scanning activities and reducing false negatives [7].

Existing methodologies for IPv6 reconnaissance have advanced techniques for partial address discovery and scanning efficiency, but remain insufficient to achieve complete visibility of active IPv6 addresses, and identifying protocol-specific vulnerabilities. While machine learning-based approaches improve candidate generation, they struggle with computational overhead, dependency on training data quality, and adapting to evolving network behaviors.

## III. Tool Description

This section offers an overview of current design and sets the stage for a detailed discussion of the innovative features integrated into **ptnetinspector**. In the subsequent sections, we will delve into its software architecture, diverse scanning modes, intuitive usage, and robust implementation strategies, all of which contribute to its enhanced capabilities in exhaustive IPv6 address discovery, node mapping, and vulnerability assessment.

### A. Software architecture

The **ptnetinspector** tool is a command-line application developed in Python that must be run as the **root** user (via `sudo`) under Linux. The user provides the desired scanning parameters (for example, `-t 802.1x/a/a+/p`, which will be described in Subsection B) along with the network interface and output options. The tool supports four primary scan modes: 802.1x, passive, active, and aggressive. These modes may be used individually or combined for complex scanning scenarios. Fig. 1 shows the overall architecture.

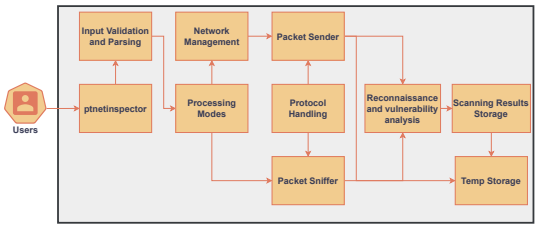


Fig. 1: High-level software architecture of **ptnetinspector**

**Input Validation and Parsing** is the first component executed when the tool is run. It validates all command-line parameters and ensures that mandatory options (such as scan type `-t` and interface `-i`) are present. The parser interprets complex mode strings (e.g., `802.1x/a/a+/p`) and converts them into an internal configuration that guides the subsequent scanning operations.

After validation, the **Network Management** module checks and configures the system's network settings. It verifies that the specified interface is active, retrieves the associated IP and MAC addresses, and sets up required network rules. For example, it updates IPv6 tables so that packets can be sent without interruption. This module ensures that the environment is properly prepared for both sending and capturing packets.

This module determines which scanning modes have been requested by the user (802.1x, active, passive, aggressive, or a combination). It coordinates the overall scanning strategy by enabling the correct functions in the **Packet Sender** and **Packet Sniffer** modules. The **Processing Modes** module acts as the controller, selecting the scanning workflow based on the chosen parameters.

The **Packet Sender** module is responsible for constructing and transmitting packets according to the corresponding scanning modes, which will be discussed later in the next section. The **Packet Sender** uses the **Protocol Handling** module to build each packet with the correct protocol fields and options.

Running concurrently with the **Packet Sender**, the **Packet Sniffer** module captures all incoming network traffic. It operates in promiscuous mode to record responses from remote hosts. The captured data includes packet headers and payloads, which are stored for further analysis. This module ensures that no response is missed during the scanning process.

The **Protocol Handling** module is the core utility used by both the sender and sniffer. When sending packets, it assembles packets by setting proper header fields, options, and protocol-specific information. When receiving packets, it parses the captured data, extracting meaningful information such as source and destination addresses, protocol types, and other relevant fields. This module is essential for both constructing accurate probes and interpreting the responses during the vulnerability analysis phase.

After the scanning and packet capturing phases are completed, the captured data is processed by the **Reconnaissance and Vulnerability Analysis** module. This module reconstructs the network topology, identifies addressing schemes, enumerates services running on discovered hosts, and detects potential vulnerabilities. The analysis integrates both direct responses and inferred data from protocol behaviors, providing a comprehensive security assessment.

For debugging and further analysis, intermediate data such as raw packet captures and preliminary results are stored in a temporary storage area. This module supports troubleshooting

and audit processes by maintaining a log of all temporary data generated during the scan.

Finally, the **Scanning Results Storage** module formats and stores the final output. Depending on the options selected (e.g., JSON output, minimal or detailed terminal output), it generates a report that includes the network topology, discovered addresses, service information, and vulnerability findings. This report is both displayed to the user and saved in temporary storage for later review.

### *B. Scanning modes*

The **ptnetinspector** tool supports four primary scanning modes, each chosen based on the user's intended network assessment objectives as shown in Fig. 2.

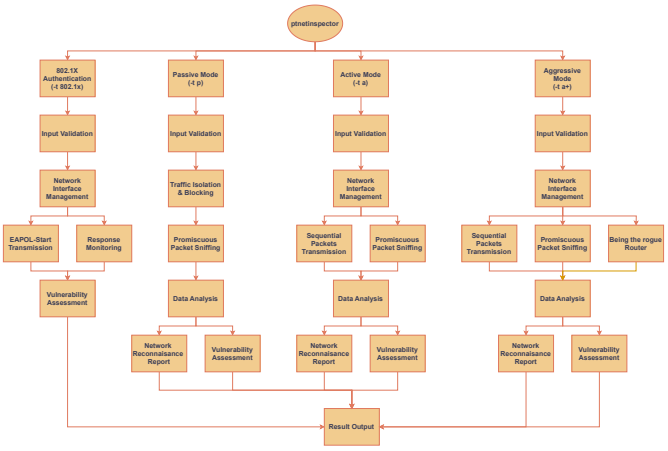


Fig. 2: Scanning modes of **ptnetinspector**

In first mode called 802.1x (using `-t 802.1x`), the tool tests the presence of EAP authentication on the network. The process begins by sending an EAPOL-Start message to trigger the authentication process. The tool then waits for responses from any available network authenticators. A valid response confirms that 802.1x authentication is active. This mode is integrated into the vulnerability assessment by checking whether proper 802.1x security controls are implemented.

Passive mode (using `-t p`) is designed to capture network traffic without introducing any new packets. The tool places the network interface in a listening state and captures all incoming traffic over a specified duration. Since no packets are sent, this method is completely stealthy and non-intrusive. The captured data is later analyzed to reconstruct the network topology, identify active nodes, and assess vulnerabilities. Key advantages of passive mode include:

- Minimal impact on network performance.
- Stealth operation that reduces the risk of detection.
- Ability to capture authentic, real-time network behavior without triggering security alerts.

Active mode (using `-t a`) involves the transmission of a defined sequence of packets to trigger responses from network devices. In this mode, after input validation and network management steps (including updating IPv6 tables), the tool sends sequential probes to discover nodes, build an IPv6 topology, and collect addressing information. Although many tasks mirror those in passive scanning, active mode provides additional data by prompting responses from devices. A detailed algorithm for the packet transmission sequence is provided below.

**Algorithm 1** Active Scanning Process

**procedure** ACTIVESCAN
  INITCONFIG ▷ Setup
  STARTSNIFFER
  SENDMLDV2QUERY ▷ MLD Queries
  SENDMLDV1QUERY
  SENDMCASTECHO ▷ Probing
  SENDICMP6(128, "bad")
  PINGROUTERS
  SENDROUTERSOLICITATION
  QUERYMDNS/LLMNR("PTR") ▷ Discovery
  **if** GOTRESPONSES **then**
    $h \leftarrow$ GETHOSTNAME
    QUERYDNS($h$, "ANY/A/AAAA")
  **end if**
  **for all** $h \in$ Hosts **do** ▷ Addr Derivation
    **if** $h.\text{ll} \wedge \neg h.\text{global}$ **then**
      $g \leftarrow$ MAKEGLOBAL($h$.ll)
      $src \leftarrow$ NEXTADDR
      PING($src$, $g$)
    **end if**
  **end for**
  REPEATSCAN ▷ Secondary phase
  **for all** $g \in$ GlobalAddrs **do** ▷ Spoofing
    $fake \leftarrow$ GENFAKEADDR($g$)
    $src \leftarrow$ PICKADDR($fake$)
    SENDFAKEPING($src$, $g$)
  **end for**
  STOPSNIFFER ▷ Cleanup
  ANALYZE
  REPORT
**end procedure**

For Active mode, our designed approach follows these steps:

- **Initialization and Setup:** The procedure begins by initializing configuration (setting up parameters and network state) and starting the packet sniffer to capture responses during the scan.
- **MLD Query Phase:** It sends both Multicast Listener queries (version 1 and 2) twice. This helps in discovering IPv6 multicast group information.
- **Probing Phase:** The tool then sends a multicast echo request, followed by an ICMPv6 probe (with option type 128 and a "bad" parameter) to test for malformed or unexpected responses. Next, it pings routers and sends a Router Solicitation to gather network topology data.
- **mDNS/LLMNR Discovery:** A multicast DNS and Link-Local Multicast Name Resolution query for PTR

records are issued. If responses are obtained, the host name is extracted and additional mDNS/LLMNR queries (for ANY, A and AAAA records) are performed to resolve IP addresses.

- **Address Derivation:** For each discovered host that has only a link-local address (without a global address), a global address is derived (based on the EUI-64, and RFC 7217), and a unicast ping is sent to this derived address for testing the availability.
- **Secondary Scan and Spoofing:** The algorithm optionally repeats the scan sequence. Then, for every discovered global address, it generates a fake global source address and sends a spoofed ping. This step helps in testing vulnerabilities related to accepting unauthorized router advertisements or spoofed packets.
- **Cleanup and Reporting:** Finally, the sniffer is stopped, the captured data is analyzed, and a comprehensive report is generated.

Aggressive mode (using `-t a+`) extends Active mode by incorporating a rogue router component. In addition to the standard sequence of active probing packets, the tool also sends fake Router Advertisement (RA) packets. This additional behavior tests whether network devices accept unauthorized RA messages, a vulnerability that can allow an attacker to hijack the role of the default router. Such misconfigurations can lead to traffic redirection, data interception, or bypassing of security controls. Aggressive mode is especially useful for identifying weaknesses in IPv6 RA guard mechanisms and for exposing vulnerabilities related to unauthorized router advertisements.

## IV. Implementation and Evaluation

To validate the effectiveness of the **ptnetinspector** tool against widely used network security tools, including Nmap, THC-IPv6, SI6, and fi6s, we have designed an experimental testbed scenario, as illustrated in Fig. 3. We, as the IPv6 scanner (running on a Kali 2024.4), have connected to a local IPv6 network containing seven other nodes. These nodes and their IPv6 configurations are summarized in Table I. Briefly:

- Node 1 (Cisco CSR 1000v Router): Acts as the default gateway. It advertises the global unicast prefix `2001:a:b:c::/64` via SLAAC. The router itself holds a link-local address (EUI-64 derived) and one global unicast address `2001:a:b:c::1`.
- Nodes 2 (Windows 10), 3 (Windows 11), 4 (Kali 2024.4), 5 (Ubuntu 24.04.1 LTS), 6 (macOS Sonoma), and 7 (RHEL 9.5): Each generates two global addresses via SLAAC. One is a *permanent* address (opaque global unicast address using RFC 7217) and one is a *temporary* address (privacy extensions using RFC 4941). All nodes also hold link-local addresses.
- Scanner (Kali 2024.4): Our scanning system is also equipped with link-local and global IPv6 addresses. It actively probes the network using multiple tools and our proposed method.

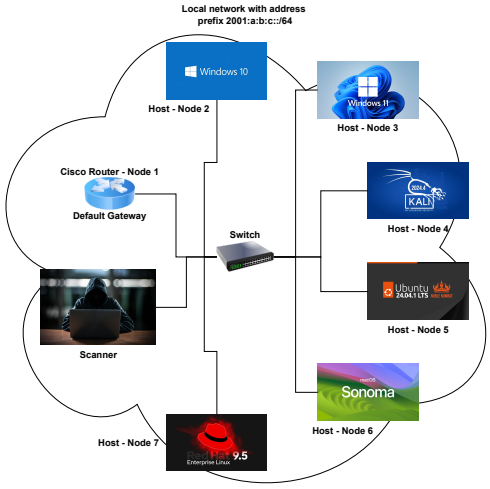


Fig. 3: Network scenario for implementation

TABLE I: Nodes and their IPv6 addressing details in the experimental scenario. LL = Link-Local, GU = Global Unicast

| Node | OS / Device | IPv6 Addresses | Notes |
|---|---|---|---|
| 1 | Cisco CSR 1000v | 1 LL (EUI-64)<br>1 GU: 2001:a:b:c::1 | Default gateway |
| 2 | Windows 10 | 1 LL (RFC 7217)<br>1 GU (RFC 7217)<br>1 GU (RFC 4941) | Host |
| 3 | Windows 11 | 1 LL (RFC 7217)<br>1 GU (RFC 7217)<br>1 GU (RFC 4941) | Host |
| 4 | Kali 2024.4 | 1 LL (RFC 7217)<br>1 GU (RFC 7217)<br>1 GU (RFC 4941) | Host |
| 5 | Ubuntu 24.04.1 | 1 LL (RFC 7217)<br>1 GU (RFC 7217)<br>1 GU (RFC 4941) | Host |
| 6 | macOS Sonoma | 1 LL (RFC 7217)<br>1 GU (RFC 7217)<br>1 GU (RFC 4941) | Host |
| 7 | RHEL 9.5 | 1 LL (RFC 7217)<br>1 GU (RFC 7217)<br>1 GU (RFC 4941) | Host |
| *Scanner* | Kali 2024.4 | 1 LL (RFC 7217)<br>1 GU (RFC 7271) | Active scanning |

The experimental testbed comprises 7 nodes configured with a total of 20 IPv6 addresses. In this experiment, we exclusively utilize the Active mode of our tool, which has proven effective in uncovering all IPv6 addresses, performing node discovery, analyzing network topology, and detecting vulnerabilities. As illustrated in Fig. 4, our evaluation focuses on three key metrics: nodes discovery, IPv6 addresses discovered, and scan time.

IPv6 Addresses Discovered: **ptnetinspector** identifies every address present (20). Nmap is second with 14 total discovered addresses but cannot enumerate permanent global unicast addresses for multiple nodes. THC-IPv6 and SI6 each yield 13 addresses, while missing the link-local address of Node 3 and all permanent global unicast addresses. fi6s detects just 1 easily predictable router address.

Nodes Discovered: **ptnetinspector** and Nmap successfully

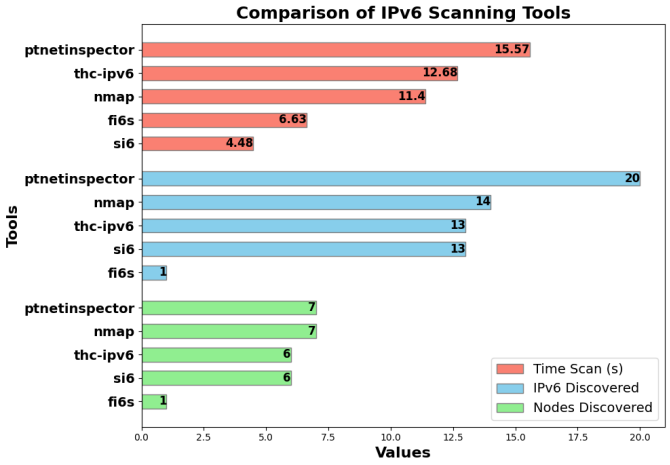


Fig. 4: Results of scanning using different tools

detects all nodes on the network (7), whereas other tools fall short. THC-IPv6 and SI6 each discover 6 nodes, and fi6s only finds 1 node.

Scan Time: The scan time for **ptnetinspector** is the highest at about 15.57 s, reflecting its thorough approach to enumerating all addresses. THC-IPv6 needs 12.68 s, Nmap takes around 11.4 s, while SI6, and fi6s finish faster but discover fewer nodes and addresses. The time difference is acceptable given the more comprehensive coverage provided by **ptnetinspector**. Furthermore, in case of larger network, with higher number of nodes and addresses, the total running time of **ptnetinspector** will not increase significantly.

In addition, we evaluated the discovered IPv6 vulnerabilities (e.g., open services, potential neighbor discovery protocol attacks) and generated a local topology map. While each tool identifies a baseline set of IPv6 exposures, **ptnetinspector** integrates these findings into a comprehensive topology analysis, highlighting the default gateway, ephemeral privacy addresses, and other SLAAC-derived host addresses.

Moreover, **ptnetinspector** also evaluates critical IPv6 and 802.1x vulnerabilities. Our tool performs an in-depth assessment of:

- **802.1x Authentication:** Verifies the presence and proper functioning of EAP-based authentication, ensuring that 802.1x security is active.
- **Router Advertisement (RA) Misconfiguration:** Detects if hosts improperly accept rogue RA packets, which can allow an attacker to impersonate the default router.
- **Neighbor Discovery Protocol (NDP) Issues:** Identifies weaknesses such as spoofed Neighbor Advertisement that could facilitate man-in-the-middle attacks.
- **mDNS and LLMNR Exposure:** Checks for the activation of mDNS and LLMNR services that leaks host information.
- **MLDv1 Acceptance:** Determines if legacy MLDv1 queries are accepted by hosts, which may indicate a security gap.
- **Predictable IPv6 Addresses:** Flags easily guessable addresses (e.g., those generated via EUI-64), which can be exploited by attackers.

Table II summarizes the vulnerability discovery capabilities of **ptnetinspector** versus several popular IPv6 scanning tools.

TABLE II: Vulnerability discovery comparison among IPv6 scanning tools

| Vulnerability | ptnetinspector | Nmap | THC-IPv6 | SI6 | fi6s |
|---|---|---|---|---|---|
| 802.1x Authentication | Yes | Partial | No | No | No |
| RA Misconfiguration | Yes | Partial | Partial | No | No |
| NDP Issues | Yes | Partial | Partial | No | No |
| mDNS Exposure | Yes | Partial | No | No | No |
| LLMNR Exposure | Yes | Partial | No | No | No |
| MLDv1 Acceptance | Yes | Partial | Partial | No | No |
| Predictable Addresses | Yes | Partial | Partial | Partial | No |

As shown in Table II, popular tools such as Nmap provide only partial vulnerability analysis because they focus primarily on enumeration. For instance, Nmap may discover active nodes and open ports but lacks dedicated tests for RA misconfigurations or thorough checks on NDP, mDNS, and LLMNR settings. Similarly, THC-IPv6 and SI6 offer limited insight into these protocol-specific weaknesses, and fi6s often returns only predictable router addresses. In contrast, **ptnetinspector** not only discovers all nodes and addresses but also systematically assesses these critical vulnerabilities.

## Conclusion and Future Directions

In summary, **ptnetinspector** provides a complete solution for IPv6 network scanning and vulnerability assessment. It accurately discovers all active nodes and addresses while performing an extensive analysis of vulnerabilities.

Future work will focus on further refining the protocol handling and vulnerability modules, expanding support to additional IPv6 security tests, and optimizing scan performance. However, even as is, **ptnetinspector** represents a significant advancement in IPv6 network security assessment, providing capabilities that fully meet the needs of modern network environments.

## References


[1] Willi Lazarov, Pavel Seda, Zdenek Martinasek, and Roman Kummel, "Penterep: Comprehensive penetration testing with adaptable interactive checklists," *Computers & Security* 154 (2025): 104399. DOI: 10.1016/j.cose.2025.104399

[2] Nmap Project, "IPv6 Scanning," https://nmap.org/book/port-scanning-ipv6.html.

[3] THC-IPv6 Toolkit, "THC-IPv6," available on Kali Linux, https://www.kali.org/tools/thc-ipv6/.

[4] Kloudle, "Scanning IPv6 with fi6s," https://kloudle.com/academy/scanning-ipv6-with-fi6s/.

[5] SI6 Networks, "IPv6 Toolkit," https://www.si6networks.com/tools/ipv6toolkit/.

[6] Doe, J., & Smith, A. (2020). 6Tree: Dynamic Discovery of Active Hosts in IPv6 Networks Using Entropy-Based Techniques. IEEE Transactions on Network Science and Engineering.

[7] Lee, K., Zhang, Y., & Chen, L. (2020). 6GCVAE: Deep Learning for Enhanced Target Generation in IPv6 Scanning. Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks.