Network Toolkit for exploiting Internet Protocol version 6 Security Vulnerabilities

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

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

***Abstract*—This work has developed a network toolkit in Python to automatically carry out several network attacks and security vulnerability testing when operating IPv6. Specifically, the program can be used to launch specified practical types of attacks according to the user's direction such as DoS (Denial of Service), dumping servers, spoofing or bypassing the firewall. The essence of these attacks is based on the inherent vulnerabilities of Extension headers, and protocols such as ICMPv6 (Internet Control Message Protocol), DHCPv6 (Dynamic Host Configuration) Protocol, which are the most important protocols in the IPv6 network operation. In addition, the analysis and illustrations have been presented, which can help network analysts to have a clearer understanding of the potential dangers that could arise from implementation of IPv6. From there, they can propose appropriate solutions to eliminate or mitigate damage when attacks occur.**

***Index Terms*—IPv6, ICMPv6, Toolkit, Network security, Attack, Python, Kali Linux, Scapy**

I. Introduction

IPv4 recently reached its limit in terms of the number of addresses. There are also many limitations that cannot be completely overcome when working with protocols under IPv4. Therefore, IPv6 has been and is being widely deployed in the infrastructure facilities of many internet service providers (ISP) and also content delivery networks (CDN). As a matter of course, the novelty of IPv6 can pose difficulties for security systems as well as network analysts in detecting and preventing potential security vulnerabilities or attacks. The potential threats have prompted the development of tools that can simulate network attacks. From there, network experts can analyze, acquire experiences, and take countermeasures to protect the system from service disruption or loss of information data or unscheduled costs.

To run this toolkit, it is necessary to have Python 3.x and the latest version of Scapy library (obtained from GitHub link [1]). Scapy is a Python-based packet manipulation library that allows users to create, send, and capture network packets. It provides a simple and powerful interface to interact with different layers of network protocols, including Ethernet, IPv6, ICMPv6, DNS (Domain Name System), TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Kali Linux, which is a popular and powerful Linux-based operating system, is used for sending packets created by this Python network toolkit and capturing messages due to specific purposes.

There are several IPv6 toolkits available that are specifically designed to help with IPv6 network testing and analysis such as Thc-IPv6 [2], Chiron [3] and Ipv6-toolkit [4]. These tools have some limitations. For example, when performing flooding attacks, Thc-ipv6 only has a low packet sending rate per second. This leads to the fact that it does not really disable victims or make them unresponsive in many cases. Chiron only runs on the Python 2.7 platform, so it is not widely used due to lacking many language features, community support, and compatibility with modern systems today. IPv6-toolkit is probably too complicated in the way of inserting parameters, which requires users to thoroughly research each detail if they want to use it properly.

The newly designed network toolkit is planned to fix some weaknesses of above mentioned tools and provide some additional functionalities in particular cases. Specifically, this designed toolkit can launch DoS attack with the maximum rate 70000 packets/second (compared to 20000 packets/second with the same type and size in case of Thc-IPv6). Moreover, user can easily observe the possible responses and attacking results shown in the terminal when running the toolkit. This feature is not available at all other mentioned toolkits even though they are more complicated for user to launch. The toolkit is still being further developed and its current source code is available at https://github.com/vafekt/IPv6-toolkit.

II. Basic Description of the developed Network Toolkit

At present, this network toolkit is comprised of the following modules, which can be used for specific purposes such as sending, sniffing, detecting and attacking:

- **ping.py**: A tool that sends ICMPv6 Echo Request messages from specified source to destination. Users are able to define how many packets to send, the size of data, option to send the file and the Ping Flood attack.
- **routing_header.py**: A tool that sends ICMPv6 Echo Request messages from specified source through defined intermediate hops to the destination. The Flood attack is included to slow down every hosts on the way to the destination.

DOI 10.13164/eeict.2023.125

- **destination_header.py**: A tool that sends SYN or ICMPv6 packet with several options of Destination Option header such as 1x Destination Option header; 1x Destination Option header with hidden data; 3x Destination Option headers; 100x Destination Option headers and 200x Destination Option headers for checking abilities to bypass the firewall.
- **fragment_header.py**: A tool that sends SYN or ICMPv6 packet with several options of Fragment header including 1x atomic fragment (with same id and different id); 3x atomic fragment (with same id and different id); 100x atomic fragment (with same id and different id); tiny fragments and overlapping fragments. It is used for checking abilities to bypass the firewall.
- **redirect.py**: A tool that sends Redirect message to a victim host, causing it to update its routing table with a new, incorrect route to a destination host. The flooding option is also included to cancel the legitimate routing permanently.
- **mldv2_query.py**: A tool that sends several types of Multicast Listener Discovery version 2 (MLDv2) Query message to discover multicast listeners in the network or pretend to be the multicast router. It also has the option of flooding attack to make the target unresponsive.
- **mldv2_report.py**: A tool that sends Multicast Listener Discovery version 2 (MLDv2) Report message to inform routers about the multicast groups that have active listeners on a network segment, add or delete multicast listeners. DoS is included as option flooding attack.
- **covert_channel.py**: A tool that performs hidden communication between two specified entities by using Covert Channel in Extension headers. It has option to encrypt the file with AES (Advanced Encryption Standard) or DES (Data Encryption Standard).
- **detect_alive.py**: A tool that detects active hosts on the attached local link, including their IPv6 addresses and link-layer addresses.
- **detect_new.py**: A tool that detects new hosts joining the attached local link based on Duplicate Address Detection (DAD) process.
- **neighbor_solicitation.py**: A tool that sends Neighbor Solicitation message to specified target for resolving link-layer address and status of that victim, together with option to flood it.
- **neighbor_advertisement.py**: A tool that answers every Neighbor Solicitation message with a falsified Neighbor Advertisement message for spoofing address resolution.
- **router_solicitation.py**: A tool that sends arbitrary Router Solicitation message to specified target, with option to flood.
- **router_advertisement.py**: A tool that sends arbitrary Router Advertisement message to specified target with an aim to spoof attack, changing the default router, creating bogus IPv6 prefix on the link or poisoning routing entries. The option to flood is included to cause DoS on the target.
- **prevent_autoconfiguration.py**: A tool that prevents every host on the local link from auto-configurating its global IPv6 addresses. Therefore, these hosts cannot communicate with the external network.
- **implant_mtu.py**: A tool that implant the Maximum Transmission Unit (MTU) to a specified target, so the victim can only transfer data up to this defined MTU.
- **smurf.py**: A tool that triggers smurf attack to flood a specified target.
- **flood_dhcpv6.py**: A tool that kills all DHCPv6 servers with enormous falsified DHCPv6 Solicit messages and option Rapid Commit.
- **dhcpv6_server.py**: A tool that operates as a fake DHCPv6 server.

To launch the program properly, users must access to the directory of the toolkit, choose one of the tools and run it as root. When starting the module, it is necessary to insert specific parameters (users must define at least the network interface to use). At any time when needing help, users can run the particular tool with **-help** switch for more information, which is shown in Fig. 1 for the case of tool **router_advertisement.py**.

```
┌──(root㉿kali)-[/home/kali/IPv6-generator]
└─# python3 router_advertisement.py --help
usage: router_advertisement.py [-h] [-smac SOURCE_MAC] [-sip SOURCE_IP]
                               [-dip DESTINATION_IP] [-M] [-O] [-H] [-A]
                               [-prf {High,Medium,Low,Reserved}] [-lft ROUTER_LIFETIME]
                               [-rcht REACHABLE_TIME] [-rtrt RETRANS_TIMER]
                               [-prefix PREFIX_INFO] [-rmac ROUTER_MAC] [-mtu MTU]
                               [-dns DNS [DNS ...]] [-p] [-f]
                               [interface]

Sending arbitrary Router Advertisement message to the specified target, which can be used
for Router Advertisement spoofing attack, changing default router, creating bogus IPv6
prefix on the link or flooding the target.

positional arguments:
  interface             the network interface to use from the sender

options:
  -h, --help            show this help message and exit
  -smac SOURCE_MAC      the MAC address of sender (resolved from the interface if
                        skipping)
  -sip SOURCE_IP        the IPv6 address of sender (resolved from the interface if
                        skipping)
  -dip DESTINATION_IP   the IPv6 address of destination (ff02::1 if skipping)
  -M                    the M flag (Managed Address Configuration)
  -O                    the O flag (Other Configuration)
  -H                    the H flag (Home Agent)
  -A                    the A flag (Address Configuration)
  -prf {High,Medium,Low,Reserved}
                        the preference level of default router (set to Medium if skipping)
  -lft ROUTER_LIFETIME  the router lifetime in seconds (set to 1800s if skipping)
  -rcht REACHABLE_TIME  the router lifetime in milliseconds (set to 30000ms if skipping)
  -rtrt RETRANS_TIMER   the retransmission timer in milliseconds (it is set to 0ms when
                        skipping)
  -prefix PREFIX_INFO   the prefix information of router (not included when skipping)
  -rmac ROUTER_MAC      the MAC address of the desired router
  -mtu MTU              the MTU on the link to router
  -dns DNS [DNS ...]    the IPv6 address of DNS server (separated by space if more than 1)
  -p                    send the RA messages periodically every 5 seconds
  -f                    flood the target
```

Fig. 1: Manual page of router_advertisement.py tool.

III. Brief Description of the Testing Environment

The testing is implemented in the virtual machine environment powered by GNS3. There are overall 7 devices including 2 Cisco routers; 1 Ethernet switch; 1 Kali Linux PC with version 2022.4; 1 Windows 10 Home PC and 2 Ubuntu PCs with version 22.04.1 LTS (Long-term Support). The network topology is depicted below in Fig. 2.

Router R1 and R2 are the legitimate default routers on their network segments (`2001:db8:abcd:1::/64` and `2001:db8:abcd:3::/64`, respectively). These two

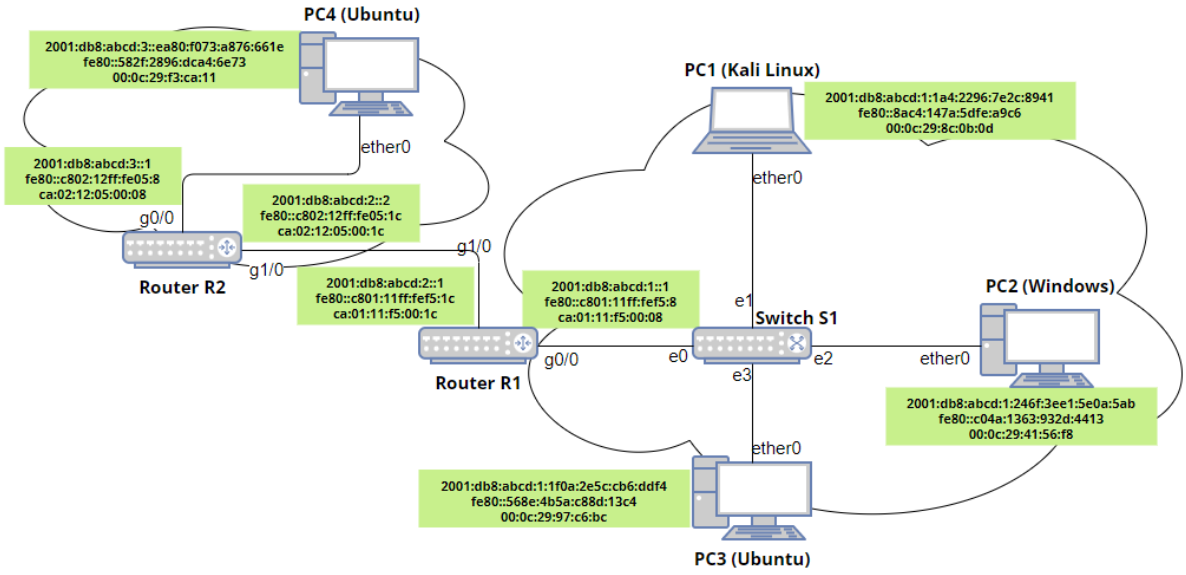

Fig. 2: The testing network topology with address specification (/64 prefix used in all subnets).

routers provide Router Advertisement to clients for address auto-configuration. The PCs from number 1 to number 3 reside on the local link of router R1, while PC4 locates on the network of router R2. All PCs are the clients in the model, and PC1 (Kali Linux) takes on the job of being an attacker in specific situations, which will be mentioned later.

In the scope of the article, two tools including **detect_alive.py** (section IV) and **neighbor_advertisement.py** (section V) from the toolkit are presented briefly to see how they impact victims and/or the network operations.

IV. Brief Description of the detect_alive.py Tool

Before executing any type of attack, it is essential to get information about active hosts on the attached link and about the target. Ping sweep or similar brute force scans is usually applied in IPv4, but conducting this type of scan of all IPv6 addresses is impractical due to the vast address space of IPv6 subnets. Instead, this stage can be carried out by transmitting ICMPv6 Echo Request messages (or similar packets) to the all-nodes multicast address.

However, not all nodes respond to the normal ICMPv6 Echo Request (notably, machines running Windows system often ignore this type of packet). Therefore, the idea of sending an additional type of packet, which is called malformed ICMPv6 Echo Request, is implemented. In this packet, an unknown option of Destination header is inserted (depicted in the yellow frame of Fig. 3) to make hosts unable to process the packet. Then, all hosts should send a Parameter Problem response (shown as the red frame of Fig. 3) back to the attacker. This has been defined in specification RFC 4884 [5]. As a result, information such as IPv6 address and MAC address of victims are exposed.

That is what the tool **detect_alive.py** performs. In the scenario (Fig. 2), it is taken into account that all devices are in active mode. From PC1 (Kali Linux), which is the attacker,

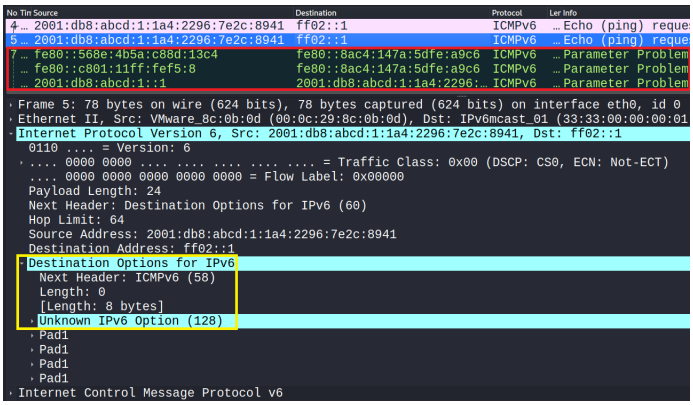

Fig. 3: Illustration of captured malformed ICMPv6 Echo Request packet and its responses.

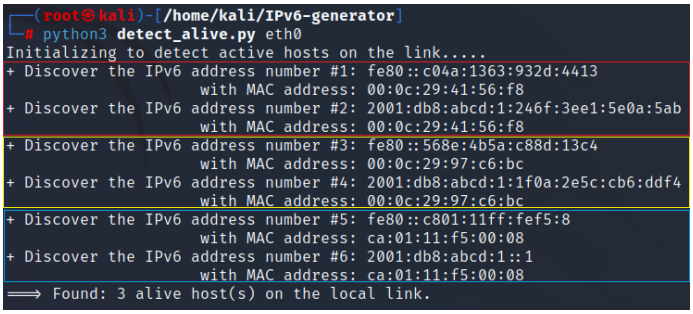

Fig. 4: The operation and result of detect_alive.py tool after running.

the tool is launched as shown in the Fig. 4. The toolkit sends ICMPv6 Echo Request packets using both IPv6 link-local and global source addresses in order to get the link-local and global ones from victims.

After launching the attack, there are 3 hosts found on the local link (Fig. 4). In the local network containing attacker's PC1, there are 3 hosts including Router R1, PC2 (Windows), and PC3 (Ubuntu). The red frame contains address information about PC2, the yellow frame is from PC3, and the blue frame is from router R1. Therefore, the program has operated correctly. Obtained addressing information could be used when using other tools.

V. Brief Description of the neighbor_advertisement.py tool

This tool executes Neighbor Advertisement spoofing attack. Specifically, if the attacker (PC1) responds to a Neighbor Solicitation (NS) message with a falsified Neighbor Advertisement (NA) that contains his own link layer address, he can disrupt the process of resolving link layer addresses.

Since the victim (router R1, PC2 or PC3) accepts this NA packet, it will start sending all data link frames to the MAC address of the attacker. If the attacker goes a step further and uses a falsified NA message to spoof the destination node, he can carry out a man-in-the-middle attack, which allows him to intercept and read all data transmitted between the two parties involved in the conversation, as depicted in Fig. 5. The attack in progress is shown in Fig. 6.

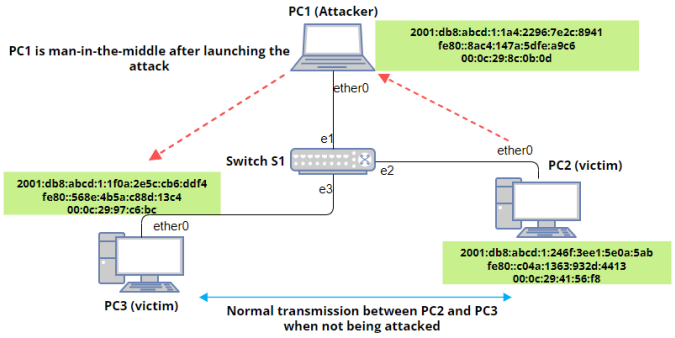

Fig. 5: Illustration of Neighbor Advertisement spoofing attack in the scenario.

```
┌──(root㉿kali)-[/home/kali/IPv6-generator]
└─# python3 neighbor_advertisement.py eth0
Initializing Neighbor Advertisement spoof attack (press Ctrl+C to stop
cess).....
net.ipv6.conf.all.forwarding = 1
.....
.........
+ Spoofing to the host: 2001:db8:abcd:1:246f:3ee1:5e0a:5ab
   as pretending to be: 2001:db8:abcd:1:1f0a:2e5c:cb6:ddf4
```

Fig. 6: The running program of detect_alive.py tool.

In this scenario, when first communicating with PC3, PC2 sends Neighbor Solicitation message for achieving the link layer address of PC3. The attacker PC1 spoofs the PC2, as the result is described in Fig. 6. The attacker then pretends to be PC3. By allowing traffic forwarding (`sysctl -w net.ipv6.conf.all.forwarding=1`) in Kali operating system, the attacker now becomes man-in-the-middle, which means PC2 cannot communicate directly with PC3 but the PC1 (attacker) will receive the data from PC2 and forward it to PC3.

Besides, it is worth noting that this attack can occur in two directions, meaning that PC1 (the attacker) spoofs PC3 in the opposite direction and pretends to be PC2. As a result, the entire information exchange process between the two PCs is completely interfered by the attacker. Similarly, we can use this attack to pretend to be legitimate default gateway of the network and become man-in-the-middle for whole network.

As can be seen from the Listing 1, the neighbor cache of PC2 (Windows) are observed to change as soon as PC2 and PC3 start sending messages to each other. The same thing happens to the neighbor caches of PC3. After the attack happens, the link layer address of PC3 changes from the real one (`00:0c:29:97:c6:bc`) to a new address (`00:0c:29:8c:0b:0d`), which is actually the link layer address of the attacker PC1. This allows the attacker to redirect all IPv6 connections through his machine.

```
# Before the attack
C:\WINDOWS\system32>netsh interface ipv6 show neighbor

Internet Address                          Physical Address
----------------------------------------  -----------------
2001:db8:abcd:1::1                        ca:01:11:f5:00:08
2001:db8:abcd:1:1a4:2296:7e2c:8941        00:0c:29:8c:0b:0d
2001:db8:abcd:1:1f0a:2e5c:cb6:ddf4        00:0c:29:97:c6:bc


# After the attack
C:\WINDOWS\system32>netsh interface ipv6 show neighbor

Internet Address                          Physical Address
----------------------------------------  -----------------
2001:db8:abcd:1::1                        ca:01:11:f5:00:08
2001:db8:abcd:1:1a4:2296:7e2c:8941        00:0c:29:8c:0b:0d
2001:db8:abcd:1:1f0a:2e5c:cb6:ddf4        00:0c:29:8c:0b:0d
```

Listing 1: The neighbor cache in PC2 (Windows) before and after the happening attack (outputs shortened).

Conclusions

The structure of the network toolkit for exploiting IPv6 security vulnerabilities has been briefly described with concrete modules. The toolkit is still under development, yet, it has removed some drawbacks of other available attacking toolkits. Two modules (section IV and V) have been briefly presented to prove the workability of the toolkit. In the future, this toolkit is continuing to be developed with more features (types of attack, related options) and to be more user friendly.

References

[1] P. Biondi, Scapy, [online] Available: http://www.secdev.org/projects/scapy.

[2] M. van Hauser. THC IPv6 Attack Toolkit, [online] Available: http://www.thc.org/thc-ipv6/

[3] Chiron, [online] Available: https://www.secfu.net/tools-scripts/

[4] IPv6-toolkit, [online] Available: https://www.kali.org/tools/ipv6-toolkit/

[5] Extended ICMP to Support Multi-Part Messages, [online] Available: https://www.rfc-editor.org/rfc/rfc4884

[6] J. Jeřábek, *Pokročilé komunikační techniky*. Skriptum FEKT Vysoké učení technické v Brně 2022, s.1-174.

[7] Kurose, J. F., Ross, K. W., Computer networking: a top-down approach. 7th global ed. Essex: Pearson, 2017, 852 s. ISBN 978-1-292-15359-9.