



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

APPLICATION FOR PERFORMING MAN-IN-THE-MIDDLE IPV6 ATTACKS

APLIKACE PRO PROVEDENÍ MAN-IN-THE-MIDDLE IPV6 ÚTOKŮ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Branislav Kadlec

SUPERVISOR

VEDOUCÍ PRÁCE

Ing. Viet Anh Phan

BRNO 2024

Bachelor's Thesis

Bachelor's study program **Information Security**

Department of Telecommunications

Student: Branislav Kadlec

ID: 241045

**Year of
study:** 3

Academic year: 2023/24

TITLE OF THESIS:

Application for performing man-in-the-middle IPv6 attacks

INSTRUCTION:

Study from the literature the issue of development in the python programming language and the functioning of selected communication protocols: especially IPv6, ICMPv6, DHCPv6. Also familiarize yourself with the well-known security problems of these protocols and the issue of man-in-the-middle attack tools. As part of the bachelor's thesis, create a lab scenario and application in Python that will enable man-in-the-middle IPv6 attacks to be carried out within the virtual network environment. Based on the results, analyze the performance and advantages of this application compared to current applications. The solution will also include a description of the test, ways to perform it, and the possible threats on the given system.

RECOMMENDED LITERATURE:

[1] W. Liu, P. Ren, Y. Zhao, D. Sun and K. Liu, "Study on attacking and defending techniques in IPv6 networks," 2015 IEEE International Conference on Digital Signal Processing (DSP), Singapore, 2015, pp. 48-53, doi: 10.1109/ICDSP.2015.7251328.

[2] JEŘÁBEK, J. Pokročilé komunikační techniky. Skriptum FEKT Vysoké učení technické v Brně, 2020. s. 1-180.

**Date of project
specification:** 5.2.2024

**Deadline for
submission:** 28.5.2024

Supervisor: Ing. Viet Anh Phan

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

WARNING:

The author of the Bachelor's Thesis claims that by creating this thesis he/she did not infringe the rights of third persons and the personal and/or property rights of third persons were not subjected to derogatory treatment. The author is fully aware of the legal consequences of an infringement of provisions as per Section 11 and following of Act No 121/2000 Coll. on copyright and rights related to copyright and on amendments to some other laws (the Copyright Act) in the wording of subsequent directives including the possible criminal consequences as resulting from provisions of Part 2, Chapter VI, Article 4 of Criminal Code 40/2009 Coll.

ABSTRACT

This thesis presents the development of a Python application designed to execute Man-in-the-Middle (MITM) attacks within a virtual IPv6 network. Motivated by a deep interest in information security, networking, and programming, this research aims to create a versatile tool that integrates various attack methods into a single, cohesive solution.

The objectives include the development of Python code utilizing the Scapy library, a thorough understanding of IPv6, ICMPv6, and DHCPv6 protocols, and the creation of an application that focuses on three primary attack vectors: a fake DNS server, a fake DHCP server, and a fake default gateway. The evaluation criteria will assess the performance and advantages of the application compared to existing specialized tools. Methodologically, the Scapy library is employed, and a virtual network environment is meticulously designed for comprehensive testing. Ethical considerations emphasize user responsibility in the utilization of such tools, drawing analogies with dual-purpose tools like knives.

The scope of the thesis encompasses theoretical foundations, application design, virtual network setup, testing methodologies, and result analysis. The aim is to contribute valuable insights into MITM attacks while providing a versatile tool for security practitioners. The research explores the intersection of Python programming, networking protocols, and cybersecurity, offering a thorough investigation into the dynamic field of Man-in-the-Middle attacks.

KEYWORDS

Man-in-the-middle, IPv6, ICMPv6, DHCPv6, DNS, Python, Scapy, default gateway, DHCPv6 server, DNS server

Author's Declaration

Author: Branislav Kadlec
Author's ID: 241045
Paper type: Semestral Project
Academic year: 2023/24
Topic: Applications for man-in-the-middle IPv6 attacks

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno

.....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to those who have contributed to the completion of this bachelor's thesis. This journey would not have been possible without the support, guidance, and encouragement from various individuals.

First and foremost, I am deeply thankful to my thesis advisor, Ing. Phan Viet Anh, for their unwavering support and valuable insights throughout the research process. Their expertise and constructive feedback were instrumental in shaping the direction of this study.

I extend my heartfelt appreciation to the faculty members of the Department of Telecommunications for their academic guidance and for providing an environment conducive to intellectual growth. The discussions and interactions with them have been enriching and have significantly contributed to the depth of my understanding.

I am indebted to my family for their constant encouragement and belief in my abilities. Their understanding and patience during the demanding periods of this thesis were invaluable.

I would like to express my gratitude to my friends and peers who provided moral support and shared their perspectives, creating a collaborative and stimulating academic atmosphere.

Lastly, I acknowledge the resources and facilities provided by Brno University of Technology that have been crucial in conducting this research.

This thesis stands as a testament to the collective efforts of those who have been part of this academic journey. Thank you all for your invaluable contributions.

Contents

Introduction	15
Aim of the thesis	17
1 Theory	19
1.1 Internet Protocol version 6	19
1.1.1 IPv6 address structure	19
1.1.2 Domain Name System	21
1.2 Internet Control Message Protocol version 6	22
1.2.1 Neighbour Discovery Protocol	23
1.3 Dynamic Host Configuration Protocol for IPv6	24
1.4 Man-in-the-middle Attack	25
2 Thesis Results	29
2.1 Development tools	29
2.1.1 Python	29
2.1.2 Scapy	29
2.1.3 EVE-NG	29
2.2 Scenario	30
2.2.1 Device Set-up	33
2.3 Application	34
2.3.1 Gateway impersonation mode	34
2.3.2 Rogue DHCPv6 server	37
2.3.3 Rogue DNS server	38
2.4 Virtual testing	40
2.4.1 Default gateway testing	40
2.4.2 DHCP server testing	42
2.4.3 DNS server testing	43
2.5 Testing on physical devices	45
2.5.1 Gateway impersonation mode	46
2.5.2 DHCP server mode	48
2.5.3 DNS server mode	51
2.6 Future Plans	53
Conclusion	55
Bibliography	57

Symbols and abbreviations	59
List of appendices	61
A MITM-tools Application	63

List of Figures

2.1	Simulated Topology	31
2.2	Parameters for the gateway impersonation mode	34
2.3	Development diagram of the gateway impersonation mode	36
2.4	Parameters for the rogue DHCP mode	37
2.5	Parameters for the rogue DNS mode	38
2.6	Development diagram of the DNS mode	40
2.7	Flow of a packet in the network	41
2.8	The flow of a packet through a spoofed	42
2.9	A rogue DHCPv6 server connected to the topology	42
2.10	A victim sending a DNS query	44
2.11	A victim sending a DNS query after it has been manipulated	44
2.12	Topology of the physical network	45
2.13	The Victims' device communicating with a virtual device	46
2.14	Sent Router Advertisement packets, with manipulated Router lifetimes	47
2.15	The Victim's configuration before and after and attack	47
2.16	Captured packet on the Attackers machine	48
2.17	A screen capture of the application in use	49
2.18	Captured DHCP communication on the Attackers machine	49
2.19	Configured address on Victims machine	50
2.20	A screen capture of the application providing a DNS response	51
2.21	DNS query from the victim on the attackers device for the domain vut.cz	52

Introduction

This thesis is devoted to the development of a Python application for executing Man-in-the-Middle (MITM) attacks within a virtual IPv6 network. Driven by a passion for informational security, networking, and programming, the author seeks to create a versatile tool that combines various attack methods into a unified solution.

The core objectives include exploring Python development using the Scapy library, understanding IPv6, ICMPv6, and DHCPv6 protocols, and creating an application focusing on three key attack vectors: a fake DNS server, a fake DHCPv6 server, and a fake default gateway. The research extends to evaluating the application's performance compared to existing specialized tools.

Utilizing the Scapy library, the application is designed for a virtual network environment, featuring inside and outside networks for comprehensive testing. Ethical considerations emphasize user responsibility, drawing parallels with tools that have dual purposes, such as knives.

Chapters will delve into theoretical foundations, application design, virtual network setup, testing methodologies, and result analysis. The aim is to contribute insights into MITM attacks, offering a versatile tool for security practitioners.

Aim of the thesis

The main goal of this work is to create a python application that allows man-in-the-middle IPv6 attacks within a virtual network environment. This application should be versatile and adaptive to many different situations, types of networks and devices, but also easy to use and customize for professional users and beginners alike. We want to emphasise the automation and efficiency of this tool for a truly seamless experience with the combination of good readability and comprehensiveness of the output data. These standards should allow any user to choose our tool over, or in combination with already available tools, which will be compared with our result in the thesis. With the combination of good documentation, this tool should be at the same level with currently used tools and could have the potential to be used widely around the world.

1 Theory

1.1 Internet Protocol version 6

The Internet Protocol version 6 (IPv6) enables devices within digital networks to route packets of information to their correct destinations. This forms the foundation for connecting the entire world to the World Wide Web. The IPv6 protocol is the latest iteration of the Internet Protocol, designed to replace its predecessor, IPv4, due to its vastly superior address space among other enhancements [1]. This version was created to address the challenge of the internet's expansion beyond IPv4's capabilities. IPv4 allows for packets to be split into multiple fragments, accommodating the limited transmission sizes of network segments. However, with modern transmission line capacities, packet fragmentation was replaced with a system of **extension headers**. This system permits the stacking of headers one after another, potentially increasing a packet's size up to 4 GB, known as Jumbo Packets. The header length is also fixed by eliminating IPv4's variable-length options field, resulting in a 40-byte header. This is twice the size of the IPv4 header, which is a significant improvement given the considerably longer addresses. The address length has been quadrupled, allowing many more devices to connect to the internet. IPv6 addresses can be categorized into two main groups and further divided into numerous types[2].

1.1.1 IPv6 address structure

Every IPv6 address is 128 bits in length. This results in 2^{128} possible combinations, or approximately $3 \cdot 10^{38}$. This vast number allows each existing device to have a unique address. To maintain the organization of the address space, addresses consist of two complementary parts. The initial bits of any address identify the specific network. Each network can also contain smaller sub-networks, which inherit the network portion of the address from the parent network. The remaining part of the address is used to identify individual devices within the network. The maximum size of any network is determined by the length of this part. This also implies that the lengths of these two parts are variable. To track this, a number is always appended to any address called the **prefix**. A prefix simply represents the number of bits from the beginning of the address that define the network. Devices with identical prefix bits belong to the same network.

Unicast Addresses

Unicast addresses are addresses that identify a single network interface on a node. They serve as a unique identifier for communication with a specific device. The most

common types of unicast addresses include the **Global Unicast address**, which functions similarly to a postal address. This unique address refers to a single network interface of a node on the global network. All global IPv6 addresses begin with a 48 b global prefix and are then subnetted using a 16-bit Subnet ID. This structure helps reduce the size of the global IPv6 routing table. **Link-local addresses** are another type of unicast address used by devices for direct communication. Every device automatically configures a link-local address, which always starts with the prefix *FE80::/10* and is completed with the MAC address of the interface. This address allows a device to request a global address, inform connected devices of its presence, or communicate without a router. Routers do not forward packets with link-local source or destination addresses to other links.

Multicast Addresses

A Multicast address allows a device to send a packet to a group of recipients. In IPv6, a multicast address is identified by the first 8 bits being set to 1. Consequently, every IPv6 multicast address begins with *ff*. The IPv6 protocol does not include a broadcast address, so to send a packet to all nodes, we use the **All-Nodes Address**. If this address is used as the destination, it indicates that the packet is meant to be received by all nodes, prompting each node that receives it to reply. Additionally, the **All-Routers Address** enables a device to send a packet to all routers on the network. It is used as the destination address for Router Solicitation and is typically defined as *ff02::2*.

Given the vast size of the internet today, it is often necessary to connect to another network. To communicate outside its network, one router within the network must be connected to the internet and serve as the **default gateway**. A gateway functions as a point of entry into another network, often leading to a change in addressing and the use of different networking technologies. Essentially, a router directs data packets between networks with distinct network prefixes. Each computer's networking software maintains a routing table that determines which interface to use for transmission and which router within the network should forward specific sets of addresses. When none of these forwarding rules match a particular destination address, the default gateway is selected as the fallback router. The default gateway can be configured using the route command to set the node's routing table and default route.

Enterprise network systems often consist of multiple internal network segments. When a device wants to communicate with a host on the public internet, it sends the data packet to the default gateway of its network segment. This router also

has a default route configured to a device on an adjacent network, moving one step closer to the public network.

There are three different ways a device can be assigned an address: stateless address assignment, stateful address assignment, and manual assignment.

Stateless auto configuration

Stateless Address Auto-configuration (SLAAC) means that no internal state is required in the device or any other device in the network. The significant bits are taken from the network address, and the rest is generated either randomly—where there is a minimal risk of duplication due to the vast IPv6 address space—or using the EUI-64 identifier, where a MAC address is used as the basis. This process is deterministic, meaning it produces the same address if the same MAC address is provided. This fact can enable tracking a specific device throughout the internet.

Statefull assignment

Stateful assignment relies on the internal state of a device, typically a server in the network using the DHCPv6 protocol. The server keeps track of the addresses in use by devices in the network. This method ensures a unique address for each device, although it incurs a small amount of network traffic.

Manual assignment

The most reliable but also the most time-consuming method is manual assignment. It requires the system administrator to designate and assign an address to every connected device. This method demands significantly more effort compared to the other two, and for that reason, it is rarely used aside from small and controlled networks.

1.1.2 Domain Name System

The Domain Name System, commonly referred to as DNS, is a crucial and fundamental component of internet infrastructure. It functions as a distributed and hierarchical system for translating human-readable domain names, such as "www.vut.cz," into the numerical IP addresses used by network devices to locate and communicate with each other. This translation process is essential because computers operate using IP addresses, while users prefer the convenience of domain names for accessing websites, email services, and other online resources [3].

DNS operates as a global network of interconnected servers, organized into a hierarchical structure. The system includes various types of DNS servers, with root servers at the top of the hierarchy, followed by top-level domain (TLD) servers, and authoritative name servers for individual domains. When a user enters a domain name into a web browser or another networked application, their device sends a DNS query to a DNS resolver. This resolver, often provided by the internet service provider (ISP) or configured on the user's device, is responsible for locating the appropriate DNS server to resolve the requested domain.

Once the resolver identifies the necessary DNS server, it sends a request for the corresponding IP address. The DNS server processes the request, either by providing the IP address directly or referring the resolver to another DNS server if necessary. The resolver then caches the response for future use, reducing the need to repeatedly contact DNS servers for frequently accessed domains.

DNS serves as a fundamental enabler of internet navigation, ensuring that users can access websites and services without the need to remember long IP addresses. It plays a pivotal role in internet security by authenticating and validating domain name ownership through DNSSEC (Domain Name System Security Extensions) and helping to detect and mitigate malicious activities, such as DNS spoofing or cache poisoning. Overall, DNS is an integral part of the digital ecosystem, making it possible for individuals and organizations to access and interact with the vast array of resources available on the internet.

Multicast DNS

Multicast DNS (mDNS) is a networking protocol designed to facilitate the automatic discovery of devices and services on a local network without needing a centralized Domain Name System (DNS) server. Developed as part of the Zero Configuration Networking (Zeroconf) initiative, mDNS enables devices to announce their presence and respond to queries within a local network using multicast IP addresses. This protocol simplifies the process of identifying and connecting to devices such as printers, cameras, and other networked services without requiring manual configuration or traditional DNS servers. mDNS is particularly useful in scenarios lacking dedicated DNS infrastructure, promoting seamless and hassle-free communication between devices in a local network environment.

1.2 Internet Control Message Protocol version 6

The Internet Control Message Protocol version 6 (ICMPv6) is a supporting protocol for the internet protocol. It is used to transfer control messages such as errors,

success notifications, and pings. ICMPv6 helps in discovering devices, establishing initial connections, and alerting nodes about redirections in case of network failures. Similar to the Internet Protocol, ICMPv6 was adapted into its version 6 to work alongside IPv6, ensuring stable communication at the network layer. ICMPv6 messages are divided into two categories: informational messages and error messages. For this thesis, we will focus solely on the former. The most basic type of informational message is the Echo message. Echo messages, often known as pings, are used to verify connectivity between two devices. A ping consists of two separate messages: an echo request and an echo reply. The request is sent to a device to check connectivity, and the reply is sent back to the original sender to confirm successful communication.

1.2.1 Neighbour Discovery Protocol

ICMPv6 also uses a **Neighbour Discovery Protocol** to discover nodes on its own network.

Router Solicitation

To establish a connection of a device with the router in the network. Firstly, when a device connects to a IPv6 network, it sends out a **Router Solicitation** packet. This means it does not have a router to send its traffic to and is requesting configuration information.

Router Advertisement

Any available router responds with a **Router Advertisement** packet to provide a default gateway, a default hop limit and other configuration details. Routers can also send out Router Advertisements automatically after a configured amount of time. A Router Advertisement packet is also used to discover nodes that are directly linked to our device.

Neighbor Solicitation

A **Neighbor Solicitation** packet is used to ask neighbouring devices for their physical and link layer addresses.

Neighbor Advertisement

If a node receives such a message, it replies with a so called **Neighbor Advertisement** message to provide its information and the requesting node can note its information into its memory.

Redirect Message

Lastly, we will need to look at the **Redirect Message**. Such a message is sent by a router to notify devices that it is no longer available for routing traffic. It also needs to provide an address to a replacement router so devices can have an alternative to successfully route their traffic.

1.3 Dynamic Host Configuration Protocol for IPv6

Dynamic Host Configuration Protocol for IPv6 (DHCPv6) is a widely used network protocol designed to automate the process of assigning IPv6 addresses and network configuration parameters to devices connected to the same IPv6 network. DHCPv6 plays a crucial role in efficiently managing and distributing IPv6 addresses within a network, ensuring that devices can easily and accurately communicate in the IPv6 environment.

DHCPv6 closely resembles the operation of DHCP for IPv4 but has been customized to meet the specific requirements of the IPv6 protocol. When a device, referred to as a DHCPv6 client, connects to an IPv6 network, it typically requires an IPv6 address and other essential network configuration parameters. These parameters include subnet prefixes, DNS server information, default gateway details, and additional network settings. A DHCPv6 server in the network provides these parameters. DHCPv6 servers are responsible for managing and distributing these parameters to the devices that request them.

The DHCPv6 process as described in [4] begins when a DHCPv6 client sends a DHCPv6 *SOLICIT* message to the network in search of a DHCPv6 server. The message also contains fields called options which represent different network parameters the client is requesting. In response a DHCPv6 server within the network responds with an *ADVERTISE* message where it provides the client with an IPv6 address and any other necessary configuration data in the same options fields. Alternatively, it can provide the client with an error message, for example when all addresses are depleted. The client sends a *REQUEST* message where it can ask for more options or changes to the assigned configuration. The server then responds with a *CONFIRM* message which signals to the client that the configuration was successfully completed and the process is over. It's important to note that DHCPv6 can operate in different modes, offering flexibility to network administrators. These modes include stateful and stateless modes as mentioned above.

One of the main advantages of DHCPv6 is its role in simplifying network administration and management. It significantly reduces the need for manual configuration of individual devices and ensures the efficient allocation of IPv6 addresses, which is

essential for addressing the growing number of internet-connected devices and the transition to IPv6. It helps to ensure that devices across IPv6 networks can effortlessly and dynamically obtain the resources they need to participate in the exchange of data and services.

1.4 Man-in-the-middle Attack

A man-in-the-middle (MITM) attack ¹ is a cybersecurity threat where an attacker secretly intercepts and possibly alters the communication between two parties, typically without their knowledge or consent. In this type of attack, the attacker positions themselves between the two legitimate parties, effectively becoming a middleman who can eavesdrop on the data being exchanged. This can occur in various forms of communication, including internet traffic, emails, or wireless connections [5].

The attacker can then either passively monitor the communication or actively manipulate it, potentially stealing sensitive information, injecting malicious content, or causing disruptions. To carry out a man-in-the-middle attack, the attacker often exploits vulnerabilities in the communication channel or uses techniques like Address Resolution Protocol (ARP) spoofing in IPv4, DNS poisoning, or SSL/TLS (Secure Sockets Layer/Transport Layer Security) interception.

Default gateway spoofing

Default gateway spoofing is a technique used in man-in-the-middle (MITM) attacks within IPv6 networks. In IPv6, the default gateway is a crucial component responsible for routing traffic between devices on the local network and external networks, including the internet. When an attacker impersonates the default gateway in an IPv6 network, they gain control over the network traffic that passes through this gateway, effectively becoming the central point through which all data flows.

By spoofing the IPv6 default gateway, the attacker can intercept or manipulate all communication between devices on the local network and external destinations. This becomes particularly effective for executing MITM attacks within the IPv6 local network.

Default gateway spoofing in IPv6 often involves techniques like Neighbor Discovery Protocol (NDP) spoofing. In this scenario, the attacker sends deceptive NDP messages to network devices, rerouting their traffic through the attacker-controlled

¹Also known as a monster-in-the-middle, machine-in-the-middle, meddler-in-the-middle, manipulator-in-the-middle, person-in-the-middle (PITM), or adversary-in-the-middle (AITM) attack.

gateway. This enables the attacker to intercept all unencrypted data and potentially inject malicious content into web pages or software updates.

To mitigate the risks associated with default gateway spoofing in IPv6 networks, network administrators can implement several security measures. These include monitoring NDP caches, configuring network devices to use static NDP entries, or deploying intrusion detection systems to detect and prevent such attacks. Additionally, the use of encryption for network traffic, particularly through protocols like VPNs, or establishing secure communication channels, adds an extra layer of protection against MITM attacks, including those involving default gateway spoofing in IPv6 environments.

Despite these measures, a significant number of networks are not properly configured or do not use any of these techniques, leaving them vulnerable to such attacks[6].

Impersonating a DHCPv6 server

Another method used in conjunction with default gateway spoofing is the spoofing of a DHCPv6 server. DHCPv6 is responsible for assigning network configuration information, such as IP addresses, subnet prefixes, and DNS server details, to devices within the network. When an attacker successfully spoofs a DHCPv6 server, they can distribute false or malicious configuration parameters to unsuspecting devices.

By impersonating a DHCPv6 server, the attacker can manipulate the IPv6 addressing scheme and provide rogue DNS server information. This allows them to direct network traffic through their own malicious infrastructure, leading to the interception of sensitive data, DNS manipulation, and further facilitation of future attacks.

To safeguard against DHCPv6 server spoofing, network administrators should implement security measures like DHCPv6 snooping, which validates the legitimacy of DHCPv6 servers, and utilize technologies such as Router Advertisement Guard (RA-Guard) to prevent unauthorized Neighbor Discovery (ND) and DHCPv6 server announcements. Regular monitoring of DHCPv6 server activities and the use of secure communication practices help protect against potential MITM attacks, including those involving DHCPv6 server spoofing[7].

This method of attack is harder to detect, making it an effective way to compromise a network.

Rogue DNS server

Impersonating a Domain Name System (DNS) server is a prevalent and concerning tactic utilized in sophisticated man-in-the-middle (MITM) attacks, particularly

within IPv6 networks. DNS is an essential component of the internet that translates human-readable domain names into IP addresses, facilitating web browsing and communication between devices. When an attacker impersonates a DNS server, they manipulate the DNS resolution process, intercepting and altering DNS queries and responses or simply creating their own. This enables them to redirect legitimate domain requests to malicious IP addresses, potentially leading to a wide range of malicious activities.

In a DNS server impersonation attack, the attacker typically establishes a rogue DNS server or modifies the DNS queries and responses flowing through the network. One common method is DNS cache poisoning, where the attacker injects forged DNS data into the cache of a DNS resolver. As a result, the compromised DNS resolver may return malicious IP addresses when queried for a particular domain, causing users to unknowingly connect to malicious websites or servers.

Another technique that attackers employ is DNS tunneling, which allows them to divert DNS traffic through their controlled DNS server. In such cases, the attacker can use covert channels within DNS requests and responses to exfiltrate data, bypass network security measures, or maintain persistent access to a compromised network.

Mitigating DNS server impersonation attacks is critical for maintaining the integrity and security of network communication. Network administrators can implement a variety of DNS security measures to counteract these threats. DNSSEC (Domain Name System Security Extensions) is a critical tool that provides cryptographic authentication of DNS data, ensuring the authenticity and integrity of DNS responses. Additionally, using trusted and reputable DNS servers with strong security practices can help reduce the risk of DNS server impersonation. Employing anomaly detection and monitoring tools for DNS traffic can provide real-time insights into unusual activities and promptly identify and respond to potential attacks.

DNS server impersonation in MITM attacks poses a significant threat to the security and privacy of network communications. Implementing robust security practices, including DNSSEC, careful DNS server selection, and active monitoring, is crucial to safeguard against such attacks and ensure the reliability of DNS resolution in the face of potential manipulation. [8].

2 Thesis Results

2.1 Development tools

For our application we will be using a number of available tools to speed up and ease up development.

2.1.1 Python

Python is a flexible and high-level coding language renowned for its straightforwardness and clarity. It is extensively utilized for numerous purposes such as web development, data analytics, machine learning, and many others. Python's vast array of libraries and strong community backing make it a favored option for both novices and seasoned programmers. Its neat and succinct syntax, coupled with its cross-platform interoperability, renders it an effective instrument for a broad spectrum of programming endeavors. Its adaptability is the primary factor for our selection, given the multitude of open-source networking tools built on Python. [9].

2.1.2 Scapy

Scapy is a robust Python library employed for network packet manipulation and analysis. It enables us to generate, transmit, intercept, and examine network packets at a granular level, rendering it an indispensable tool for network engineers, security experts, and researchers. Scapy offers the capability to design custom packets and carry out various network-related activities, such as network scanning, protocol testing, and packet inspection. Its adaptability and extensibility make it a widely utilized tool for handling network protocols and packet-level operations in Python.

2.1.3 EVE-NG

Emulated Virtual Environment Next Generation (EVE-NG) is a robust network emulation platform tailored for IT professionals, network engineers, and students to simulate intricate networking scenarios. It offers a virtualized environment where users can create and link virtual routers, switches, firewalls, and other network devices to mirror real-world network setups. EVE-NG supports a broad spectrum of virtualization technologies, including Cisco, Juniper, and other vendor equipment, enabling users to test and validate network designs, troubleshoot issues, and gain practical experience without the necessity for physical hardware. This adaptable platform is renowned for its user-friendliness, flexibility, and capability to operate

on standard hardware, making it an invaluable resource for learning, testing, and honing networking skills.

To implement these complex scenarios, we utilize VMware Workstation Player 13, a robust and widely accessible virtualization software package. VMware Workstation Player 13 is renowned for its ability to efficiently simulate entire computer operating systems. In our setup, we use this software to create and manage virtual machines representing different roles within the network.

Specifically, we simulate a Windows 10 virtual machine as the victim device. Windows 10 was chosen due to its widespread use and relevance in real-world scenarios, which provides a realistic environment for testing the application's effectiveness. The victim machine runs standard user applications and network services, making it an ideal target for MITM attacks. By employing a Windows 10 virtual machine, we can assess the application's performance and stealthiness in intercepting and manipulating data traffic within a typical user environment.

On the other hand, the attacker is represented by a Manjaro Linux virtual machine. Manjaro Linux is selected for its extensive toolset and ease of customization, which are essential for developing and running the MITM attack application. The Manjaro virtual machine is configured with the necessary network tools and libraries, such as Scapy, to perform sophisticated network attacks. Using Linux as the attacker's operating system provides a flexible and powerful platform to implement and test various attack vectors.

2.2 Scenario

The final program should be very universal, meaning it is supposed to operate in many different situations. Impersonating a Domain Name System (DNS) server is a prevalent and concerning tactic utilized in sophisticated MITM attacks, particularly within IPv6 networks. DNS is an essential component of the internet that translates human-readable domain names into IP addresses, facilitating web browsing and communication between devices. When an attacker impersonates a DNS server, they manipulate the DNS resolution process, intercepting and altering DNS queries and responses or simply creating their own. This enables them to redirect legitimate domain requests to malicious IP addresses, potentially leading to a wide range of malicious activities.

In a DNS server impersonation attack, the attacker typically establishes a rogue DNS server or modifies the DNS queries and responses flowing through the network. For instance, DNS cache poisoning is a often used method, where the attacker injects forged DNS data into the cache of a DNS resolver. As a result, the compromised DNS

resolver may return malicious IP addresses when queried for a particular domain, causing users to unknowingly connect to malicious websites or servers.

Another technique that attackers employ is DNS tunneling, which allows them to divert DNS traffic through their controlled DNS server. In such cases, the attacker can use covert channels within DNS requests and responses to exfiltrate data, bypass network security measures, or maintain persistent access to a compromised network.

Mitigating DNS server impersonation attacks is critical for maintaining the integrity and security of network communication. Network administrators can implement a variety of DNS security measures to counteract these threats. DNSSEC is a critical tool that provides cryptographic authentication of DNS data, ensuring the authenticity and integrity of DNS responses. Additionally, using trusted and reputable DNS servers with strong security practices can help reduce the risk of DNS server impersonation. Employing anomaly detection and monitoring tools for DNS traffic can provide real-time insights into unusual activities and promptly identify and respond to potential attacks.

DNS server impersonation in MITM attacks poses a significant threat to the security and privacy of network communications. Implementing robust security practices, including DNSSEC, careful DNS server selection, and active monitoring, is crucial to safeguard against such attacks and ensure the reliability of DNS resolution in the face of potential manipulation. Here is a description of the hypothetical scenario for the purpose of this thesis.

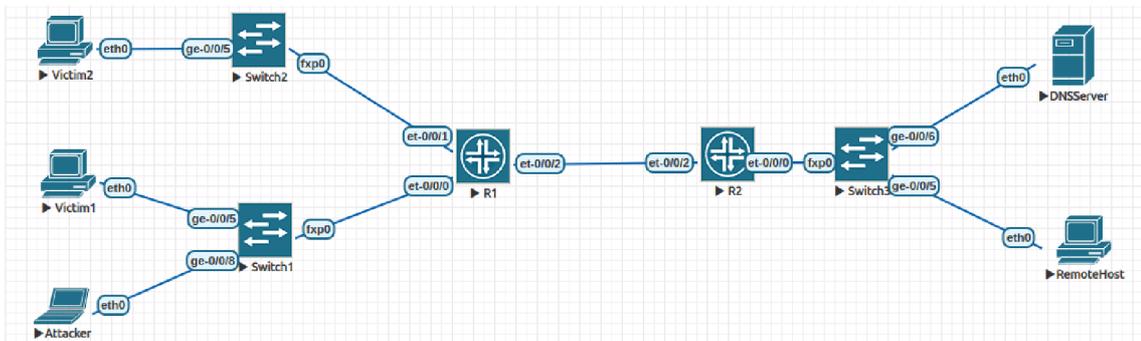


Fig. 2.1: Simulated Topology

The simulated network topology, meticulously constructed within the EVE-NG virtual environment, serves as a critical testing ground for our program, primarily due to the lack of physical devices. While some components of the topology are realized as physical devices for testing purposes, the primary focus of evaluation is within the controlled and isolated EVE-NG environment, safeguarded against external connections.

To simplify the replication of simulated scenarios in the real world, stateless configuration is employed for the victim computers, ensuring that any addresses will work with the application. This configuration ensures that the victims acquire their addresses in a manner consistent with real-world network behavior. The routers within the topology are virtual instances of the Juniper vJunos Evolved router¹, providing a solid foundation for network emulation. Similarly, the switches are virtual representations of the Juniper vJunos Switch, faithfully reproducing the functionalities of actual network switches.

This intricately designed test network aims to simulate a diverse array of cyber threat scenarios and assess the resilience of network infrastructures. At the core of this network is the main network, comprising a switch (S1) intricately connected to a router (R1). Notably, the attacker’s laptop and the first victim (Victim 1) are also linked to the S1 switch, showcasing the adaptability of the simulated environment in accommodating diverse network configurations.

The design choice to represent victims as singular devices, while aimed at resource optimization, incorporates scalability for more intricate testing scenarios. Router S1 extends its connectivity by linking to another switch (S2), which, in turn, integrates another Victim PC (Victim 2). This expansion adds complexity to the potential threat landscape, enabling a more thorough evaluation of the program’s efficacy.

R1 assumes the pivotal role of a default gateway for the entire network, facilitating seamless communication and data flow between the various components. The topology is intelligently divided into two interconnected networks, with routers R1 and R2 establishing a connection between them. To add further complexity to the setup, another instance of switch S2 is connected to R2, hosting yet another PC called Remote Host used to check connectivity to any remote host. This multi-layered network structure mirrors the intricacies of real-world scenarios, allowing for a more comprehensive evaluation of the program’s performance.

Within this elaborate topology, a DNS server has been strategically incorporated into switch S2, serving as a centralized entity that manages domain name resolution for both interconnected networks. This addition enhances the realism of the simulated environment, mirroring the crucial role of DNS servers in actual network architectures.

In summary, the intricately designed network topology within the EVE-NG simulated environment emulates a diverse array of cyber threat scenarios. The interconnected routers, switches, and victim computers create a dynamic environment that allows for a thorough evaluation of network resilience and the effectiveness of

¹Additional testing on the Mikrotik Cloud Hosted Router was also successful

security measures. This comprehensive platform provides valuable insights into the program’s capabilities and its ability to mitigate various cybersecurity challenges.

2.2.1 Device Set-up

The victim machines are operating with a standard installation of Windows 10 (22H2) Home edition. This baseline configuration ensures that no special settings, configurations, or firewall rules are necessary for the MITM attack to be successful. By utilizing a default installation, we can simulate a typical user environment, thereby providing a realistic assessment of the application’s efficacy in intercepting and manipulating data traffic.

The attacker machine, on the other hand, is configured with a fresh installation of Manjaro Linux 23.0 on a x86 architecture and using the Plasma desktop manager option. Manjaro Linux is selected for its robust performance, flexibility, and comprehensive toolset, which are essential for conducting sophisticated network attacks. The attacker machine must have Python installed, along with all required dependencies necessary for the script to run correctly. These dependencies include network libraries such as Scapy, which facilitate packet manipulation and network traffic analysis. The application makes use of specifically of the firewall tool **ip6tables** and the **tee** command, without which it can not properly function. The network infrastructure includes Juniper² and Mikrotik³ routers, which are configured in a hybrid IPv4 and IPv6 mode. However, it is important to note that no IPv4 traffic is generated on the network, focusing the attack solely on IPv6 traffic. This configuration is particularly relevant given the increasing adoption of IPv6 and the unique security challenges it presents. By focusing on IPv6, we can explore vulnerabilities specific to this protocol and demonstrate the application’s capability to exploit them. Additionally, the DHCPv6 servers on the network are disabled, resulting in the use of stateless addresses by the machines. By disabling DHCPv6 and relying on SLAAC, we can assess how effectively the application can perform MITM attacks in environments that use stateless address configuration and compare our results with a test performed with a stateful configuration. The use of standard installations and configurations for both the victim and attacker machines, along with the hybrid IPv4/IPv6 network setup and the reliance on stateless addresses, provides a comprehensive and realistic testing environment. This setup ensures that the MITM attack application can be evaluated under typical network conditions, highlighting its practical applicability and effectiveness in real-world scenarios.

²Juniper vJunos Swtich 23.1R1 and vJunosEvolved Router 23.1R1

³Mikrotik Cloud Hosted Router 7.14.3

2.3 Application

The terminal application is built on the Python library called *click* which makes the process of developing terminal command applications very easy. We do not need any complicated functions to parse command options and arguments. Click can handle it easily and pass the arguments down to the main functions. The application consists of 3 modes of operation the user chooses when starting the process. The modes are chosen when starting the application by the first argument right after the name of the script. An example command can look like this:

```
sudo python mitm.py gateway -t fe80::4 -gw fe80::1 -i eth0
```

Additional parameters can be listed by typing `--help` after any mode command.

2.3.1 Gateway impersonation mode

```
CrumblyBread > python mitm.py gateway --help
Usage: mitm.py gateway [OPTIONS]

Options:
  -i, --interface TEXT      Interface of the network
  -T, --TargetsFile TEXT    File with ll addresses of targets
  -t, --target TEXT         Target IPv6 address
  -gw, --gateway TEXT       The default gateway to impersonate
  -p, -pcap                 Write the traffic into a pcap file
  --help                    Show this message and exit.
CrumblyBread > █
```

Fig. 2.2: Parameters for the gateway impersonation mode

In this mode we can provide a single target by using the parameter `-t` or multiple targets by providing a path to a file where link-local addresses are listed on each line to the `-T` option. The `-i` option points to the interface we desire to use for the attack. The application can not run correctly without this option being specified. Lastly, the `-gw` option needs to be provided with the address of the real link-local address of the default gateway as it is not yet able to request it autonomously. Lastly the `-p` option starts generates a pcap file with all captured traffic. After these parameters are provided, the application starts by configuring itself with necessary information and then automatically starts the attack.

In this operational mode, the application attempts to mimic a default gateway while concealing router advertisement packets from the legitimate default gateway. It then redirects any traffic involving a victim to the genuine gateway, ensuring proper routing. This is achieved by repeatedly sending router advertisement (RA) packets into the network, using the link-local and MAC addresses of the original

default gateway as source addresses in the Ethernet and IPv6 packet layers, respectively. A crucial aspect is the manipulation of the *router lifetime* parameter within the Neighbor Discovery (ND) router advertisement layer, setting it to a value of 0. This configuration indicates that any device receiving this packet, in accordance with RFC 4861 section 4.2, will not designate this address as its gateway. Subsequently, a similar packet, featuring the attacker's device address as the source, is sent, but with the *router lifetime* adjusted to its maximum value. These packets are transmitted at regular intervals of five seconds, ensuring the continuous injection of these deceptive advertisements into the network. An alternative method involves monitoring the network for the genuine gateway's advertisement before sending the deceptive packets. This approach ensures that targeted devices, upon receiving the manipulated advertisement, direct their traffic to our specified IP address. We then intercept and forward this traffic to the legitimate gateway. This methodology ensures seamless routing while enabling the attacker to manipulate the data traffic without detection. To achieve comprehensive bidirectional traffic capture, we generated random virtual MAC and link-local addresses using the EUI-64 process. This innovative solution allowed us to create multiple virtual interfaces that serve as virtual Ethernet ports, each configured with unique identifiers. These virtual interfaces were instrumental in capturing all traffic directed to and from the target devices, thereby ensuring that no data packets were missed during the interception process. The deployment of virtual addresses provided a dual advantage. Firstly, it enabled the circumvention of the inherent limitations associated with incomplete traffic capture, thereby allowing for the interception of both incoming and outgoing communications. Secondly, and equally important, it facilitated the differentiation of traffic originating from various devices within the network. This differentiation was crucial for accurately monitoring and manipulating the network traffic specific to each target device, thereby eliminating any potential for confusion or overlap in the intercepted data. Furthermore, this approach adheres to a systematic methodology to ensure robustness and reliability. By continuously sending router advertisements with carefully manipulated parameters and by establishing virtual interfaces with unique identifiers, the application maintains a high level of control over network traffic. This sophisticated level of control is essential for the successful execution of Man-in-the-Middle attacks, as it allows for precise manipulation and redirection of data flows without alerting the target devices or the legitimate network infrastructure. In conclusion, the strategic use of virtual MAC and link-local addresses significantly enhances the application's capacity to perform detailed traffic analysis and manipulation. By ensuring that bidirectional traffic capture is comprehensive and that traffic from different devices can be precisely distinguished, the methodology employed not only improves the robustness of the MITM attacks but also

contributes to a more rigorous and systematic approach to network traffic interception and analysis. Consequently, the generation and utilization of virtual addresses stand as a pivotal enhancement, bolstering the effectiveness and precision of our overall strategy in performing Man-in-the-Middle attacks within a virtual network environment.

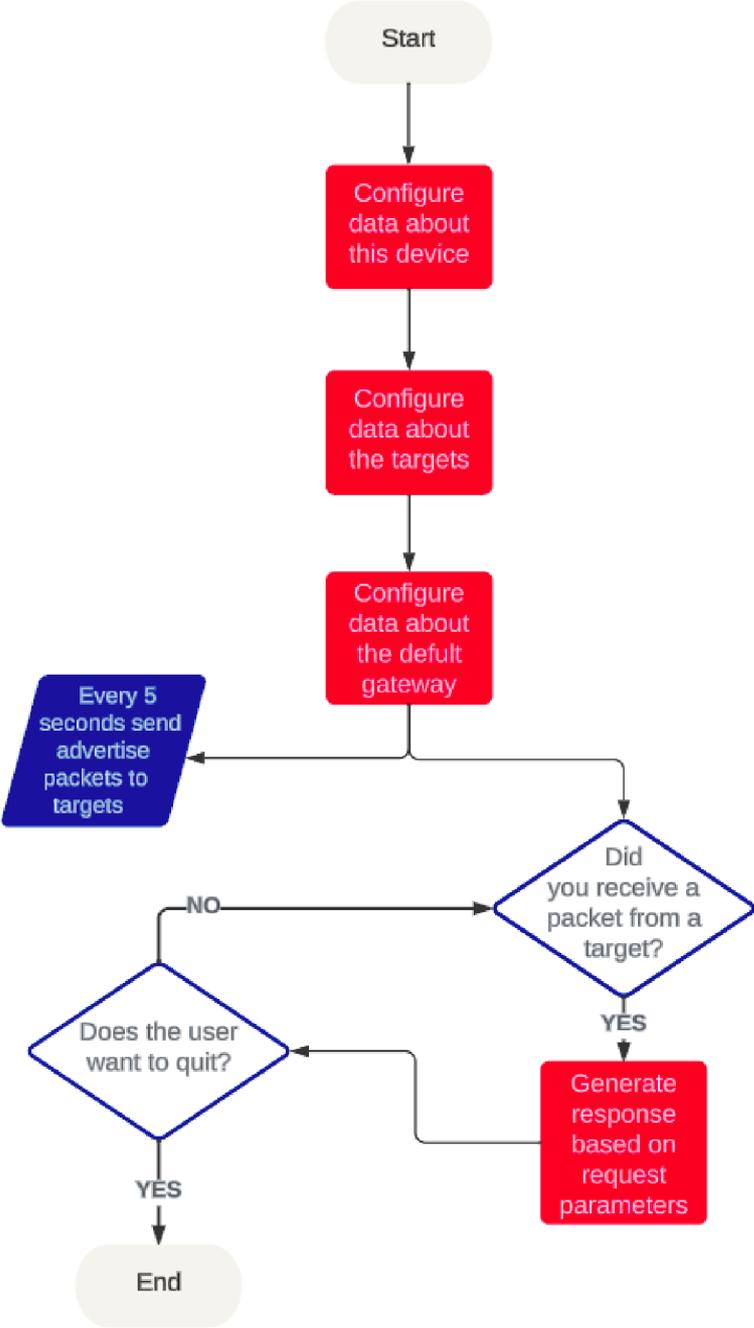


Fig. 2.3: Development diagram of the gateway impersonation mode

As this diagram shows, the application adopts the guise of a default gateway, steering incoming traffic to the legitimate gateway by sending deceptive packets with manipulated *router lifetime* values. The development diagram highlights the need to refine the packet injection process for efficiency and enhance bidirectional traffic interception techniques to bolster the overall effectiveness of this deceptive maneuver.

2.3.2 Rogue DHCPv6 server

```
CrumblyBread > python mitm.py dhcp --help
Usage: mitm.py dhcp [OPTIONS]

Options:
  -i, --interface TEXT      Interface you want to connect to
  -T, --targetsFile PATH    Filepath to list of targets to give addresses
  -n, --networkAddress TEXT Network address of pool
  -p, --prefix TEXT        Prefix length of the pool
  -dns TEXT                 Address of DNS server
  --help                   Show this message and exit.
CrumblyBread > █
```

Fig. 2.4: Parameters for the rogue DHCP mode

As the Figure 2.4 shows, we need to specify the interface on which we want to listen for DHCPv6 requests to the **-i** parameter. The application needs an interface for proper function. Also a file of link-local addresses of desired targets separated on each line. The file of targets is passed to the **-T** parameter. For proper initialization of the server, an address pool needs to be defined. This is done by providing the network address of the pool to the **-n** parameter, and the desired prefix to the **-p** parameter. Also, a better way of defining the address space is planned to be implemented in the future as this method is not ideal. The last thing this mode can offer is a DNS server in the **-dns** field, that we would have configured if requested. By default the server will use the public google DNS server.

The primary function of a DHCPv6 server is to provide DHCPv6 addresses to newly connected devices. For the server to be discoverable by these devices, it must be advertised across the network. This is accomplished through the transmission of modified router advertisement messages. To signal the availability of address configuration in the network, the *Managed address configuration* flag is set to 1, in accordance with RFC 4861 section 4.2 [10]. Additionally, the *Other configuration* flag is set, indicating the server's capability to provide other configuration parameters. The transmission of these packets is initiated from the legitimate link-local address, necessitating the configuration of the *router lifetime* field to 0 to avoid

presenting our device as a default route. This ensures that new devices connecting to the network dispatch DHCPv6 solicit messages to the *All DHCP Relay Agents and Servers* multicast address `ff02::1:2`, as specified in RFC 8415 section 7.1 [4]. These solicit messages can be responded to with an advertisement message. The packet manipulation library Scapy is utilized for crafting these responses. Although Scapy provides an Answering Machine for crafting replies to requests, it currently lacks the capability to delegate IP addresses in the DHCPv6 process. Consequently, this functionality needs to be implemented from scratch. The modification of the original Scapy answering machine is required to facilitate responses to the basic DHCPv6 process outlined in the Theory section on DHCPv6. The implementation, while basic, enables the assignment of addresses and processing of any options attached by the client to their packets. However, it currently supports only the most fundamental interactions, with each address eligible for assignment only once.

2.3.3 Rogue DNS server

```
CrumblyBread > python mitm.py dns --help
Usage: mitm.py dns [OPTIONS]

Options:
  -i, --interface TEXT      Interface you want to connect to
  -T, --TargetsFile PATH   Filepath to list of targets to provide dns
  -dns, --DnsFile PATH     Filepath to a dns translation file
  -joker                    Redirect all queries to the first address in DNS
                           File
  --help                    Show this message and exit.
CrumblyBread > █
```

Fig. 2.5: Parameters for the rogue DNS mode

The parameter that this mode requires to function is a DNS file, where each line is composed of an IPv4 address, then an IPv6 address separated by a space, then a domain name to be matched with these addresses separated by yet another space. This correctly configured file needs to be passed to the **-dns** argument. A text file of link-local addresses of targets passed into the **-T** parameter and an interface to listen for DNS requests in the **-i** parameter. Additionally the optional **-joker** flag can be attached to the command, which will cause the server to reply to every single incoming request with the first address in the DNS file.

In the role of a DNS server impersonator, the initiation of communication with targeted devices requires a systematic approach. The first step involves notifying devices of the impersonator's existence through the strategic dissemination of a Router Advertisement to all designated targets. This notification process is designed with

subtlety in mind, with the *router lifetime* intentionally set to 0 to avoid promoting the imposter device as the default route. A critical facet of this communication is the incorporation of a *Recursive DNS Server Option* layer, meticulously configured in strict accordance with RFC 6106 section 5.1 [11]. As a result, nodes traversing the network will seamlessly designate this server as their IPv6 DNS server, unwittingly paving the way for subsequent interception and manipulation. Upon successful integration into the network, all incoming DNS requests are intercepted and subjected to a meticulous processing mechanism orchestrated by Scapy. The configuration of this mechanism is highly customizable, involving the specification of user-defined parameters and the provisioning of a DNS file. This intricate process ensures the covert interception and manipulation of DNS requests within the network, allowing the Python application to operate seamlessly as a DNS server impersonator. The nuanced configuration parameters and meticulous adherence to RFC standards underscore the sophistication and effectiveness of the DNS manipulation aspect of the application, enhancing its overall capabilities in the realm of Man-in-the-Middle attacks. The figure 2.6 outlines the key steps involved in the DNS server impersonation process, including the router advertisement, Recursive DNS Server Option configuration, interception and processing of DNS requests, and the subsequent response mechanism. The flow provides a visual representation of the communication sequence between the targeted devices and the impersonator device during the DNS manipulation process. The DNS file serves as a comprehensive guide, housing domains earmarked for translation and their corresponding IPv4 addresses. These addresses are curated meticulously to respond to type A requests. Simultaneously, the DNS file contains IPv6 addresses, strategically assigned for responding to type AAAA requests. In instances where a suitable address cannot be determined from the DNS file, the DNS request seamlessly transitions to the next stage. The next stage involves the judicious forwarding of DNS requests to a legitimate DNS server, ensuring a comprehensive approach to address resolution. This intricate process ensures covert interception and manipulation of DNS requests within the network. The nuanced configuration parameters and meticulous adherence to RFC standards underscore the sophistication and effectiveness of the DNS manipulation aspect of the application, thereby enhancing its overall capabilities in the realm of Man-in-the-Middle attacks.

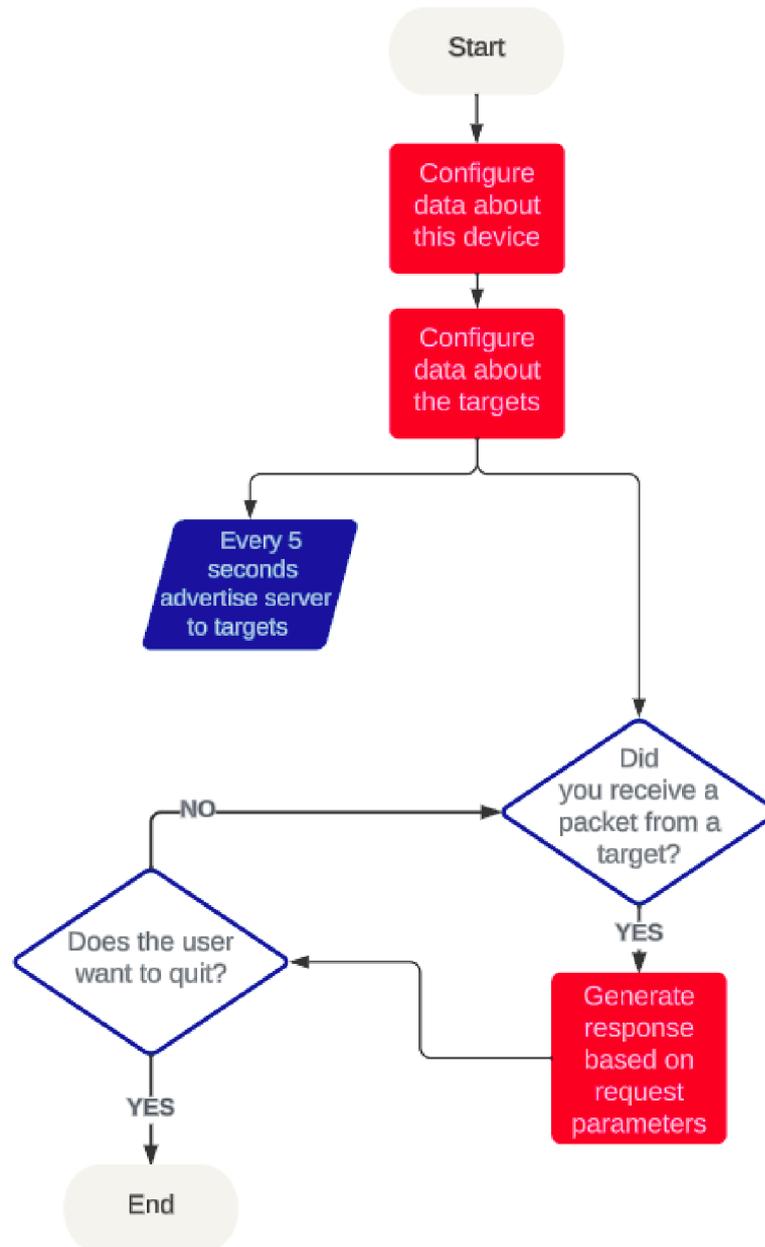


Fig. 2.6: Development diagram of the DNS mode

2.4 Virtual testing

2.4.1 Default gateway testing

Before the attack, a Windows device typically receives its network configuration via DHCP or manual settings, establishing the IP address, subnet mask, default gateway, and DNS servers correctly. The device relies on these legitimate settings

to communicate effectively within the network. With using normal traffic flow, the device routes all traffic through the default gateway, and DNS queries are resolved by trusted DNS servers. Users experience uninterrupted network services, with web pages and applications functioning as expected. Standard security measures include active antivirus software like Windows Defender, configured firewalls, and regular system updates. These measures protect the device against known threats and ensure overall network security.

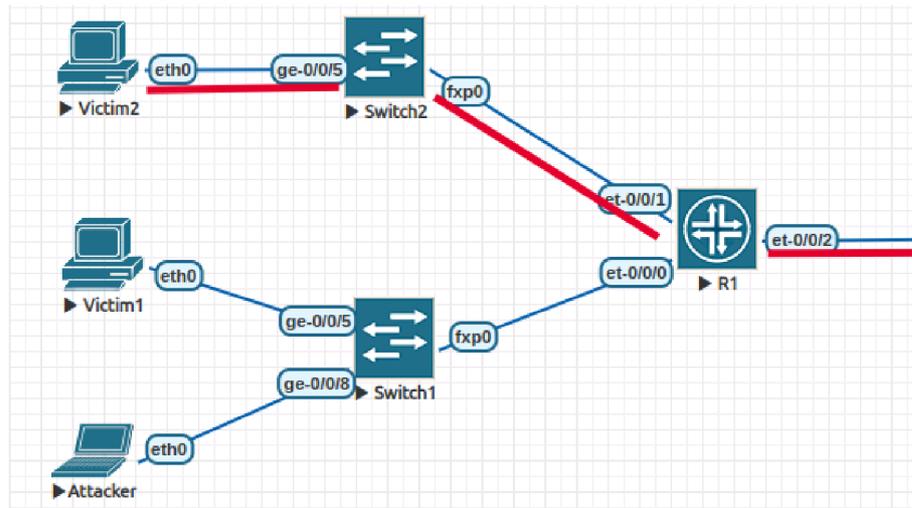


Fig. 2.7: Flow of a packet in the network

Following a default gateway spoofing attack, the Windows device is tricked into routing traffic through the attacker's device. This is achieved by the attacker sending a malicious Router Advertisement packet, spoofing the default gateway and possibly altering DNS server settings. With the attacker's device now acting as the default gateway, all traffic is intercepted. DNS queries are redirected to the attacker's DNS server, which can respond with malicious IP addresses. Despite this interception, the user's experience might remain seemingly normal initially. The immediate observable change for users may be minimal. Web browsing and other network activities will appear unaffected due to the attacker's efforts to provide legitimate-looking responses. However, subtle issues, such as occasional "page not found" errors, can arise if DNS responses are misconfigured. These errors were particularly evident during controlled testing when unassigned addresses were used in DNS responses. The attack poses significant security risks. Sensitive data, including login credentials and personal information, can be captured. Standard antivirus and firewall protections will not detect the attack, as the traffic itself appears legitimate but is rerouted through a malicious intermediary.

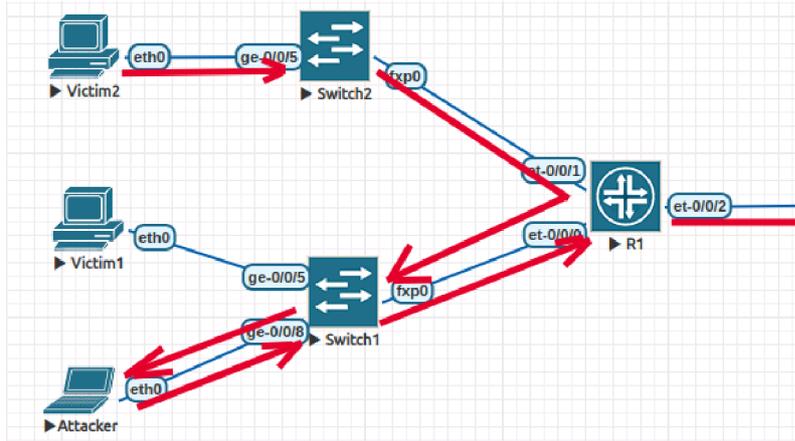


Fig. 2.8: The flow of a packet through a spoofed

2.4.2 DHCP server testing

Under normal conditions, a Windows device obtains its network configuration from a legitimate DHCP server. This configuration includes the IP address, subnet mask, default gateway, and DNS server addresses, which are essential for proper network communication. With correct settings from a legitimate DHCP server, the Windows device routes its traffic through the authorized gateway and resolves DNS queries through trusted DNS servers. The network operates smoothly, with web browsing, email, and other services functioning without issues. Standard security measures on a Windows device include active antivirus software, configured firewalls, and regular updates. These measures protect against common threats and ensure the device's overall security.

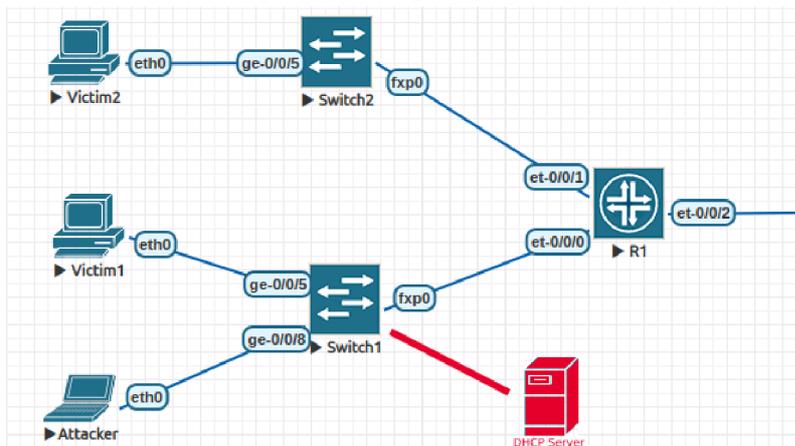


Fig. 2.9: A rogue DHCPv6 server connected to the topology

In a rogue DHCPv6 server attack, the attacker introduces a malicious DHCP server into the network, responding to DHCPv6 requests faster than the legitimate server. The Windows device, receiving incorrect settings, reconfigures its network parameters according to the attacker's instructions. This includes an incorrect default gateway and potentially malicious DNS servers. Once the network configuration is compromised, all traffic from the Windows device is routed through the attacker's device. DNS queries are redirected to the attacker's DNS servers, which can resolve domain names to malicious IP addresses, leading users to phishing sites or other harmful destinations. The immediate impact on the user might be minimal and not easily detectable. Network connectivity remains, and applications may continue to function. However, there are subtle signs, such as occasional connectivity issues, slower response times, or access to unexpected websites. These symptoms arise from the interception and manipulation of traffic by the rogue DHCP server. The attack significantly compromises the device's security. The attacker can capture sensitive information, including login credentials and personal data. Additionally, by manipulating DNS responses, the attacker can redirect users to fraudulent websites to collect information or distribute malware. Standard security measures may not immediately detect this type of attack, as the device operates under seemingly legitimate network settings provided by the rogue DHCP server.

2.4.3 DNS server testing

Under normal circumstances, a Windows device receives DNS server information either through a Router Advertisement packet. Alternatively, network administrators or users might manually configure the DNS server settings to point to specific, trusted DNS servers. These DNS servers are typically part of the network's infrastructure or reputable public DNS services, ensuring accurate and reliable resolution of domain names to IP addresses. With correct DNS settings in place, the Windows device queries legitimate DNS servers to resolve domain names. This seamless translation is essential for enabling users to access websites, email servers, and other internet-based services without interruption. Consequently, network operations, including web browsing, email communication, software updates, and cloud-based services, function smoothly and securely, providing a reliable user experience. Active antivirus software continuously scans for and neutralizes malware, viruses, and other malicious software. Configured firewalls monitor and control incoming and outgoing network traffic based on predetermined security rules, helping to prevent unauthorized access and data breaches. Regular updates to the operating system and installed software ensure that the device is protected against the latest security vulnerabilities and exploits. These updates often include patches for newly

discovered threats and enhancements to existing security features.

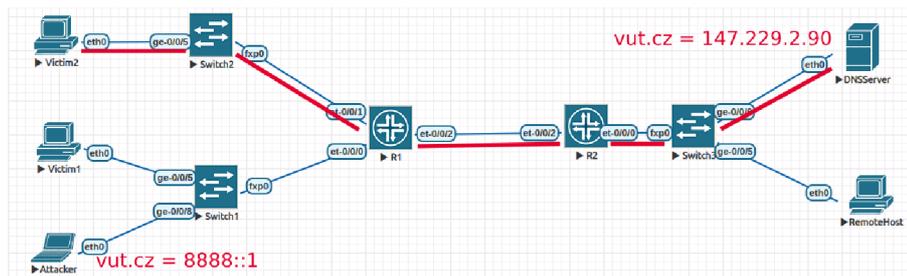


Fig. 2.10: A victim sending a DNS query

In a rogue DNS server attack, the attacker either poisons the DNS cache or provides malicious DNS server settings through DHCP or direct network manipulation. The Windows device unknowingly configures itself to use the rogue DNS server, redirecting its DNS queries to the attacker-controlled server. With the DNS server compromised, all DNS queries from the Windows device are directed to the rogue server. The attacker can resolve domain names to malicious IP addresses, leading users to phishing sites, malware distribution points, or other harmful destinations. The immediate impact on the user might be minimal and not easily detectable. Network connectivity remains, and applications may continue to function normally. However, subtle symptoms such as unexpected website redirects, unusual SSL certificate warnings, or slower response times may occur due to the interception and manipulation of DNS queries by the rogue server. The attack poses significant security risks. The attacker can capture sensitive information, including login credentials and personal data, by redirecting users to fraudulent websites. Additionally, by providing malicious DNS responses, the attacker can facilitate the distribution of malware. Standard security measures may not immediately detect this type of attack, as the DNS traffic itself appears legitimate but is being manipulated by the rogue server.

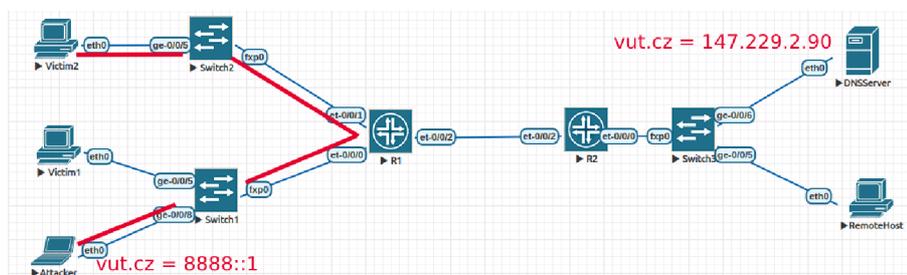


Fig. 2.11: A victim sending a DNS query after it has been manipulated

2.5 Testing on physical devices

For a short period of time, I was able to gain access to physical devices and was able to use them to recreate a section of my simulated topology as seen in Fig. 2.1. The recreated section looked like this:

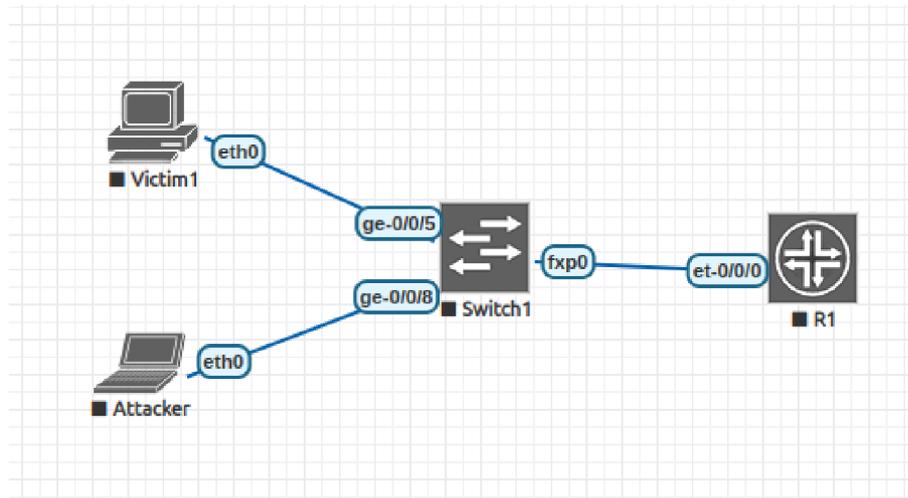


Fig. 2.12: Topology of the physical network

The addresses were set up according to this addressing table:

Table 2.1: Addressing table for physical testing

Device name	link-local address	MAC address
R1	fe80::1	cc:b1:82:82:6b:7a
Attacker	fe80::81c7:95db:d212:fb18	c4:23:60:7c:5f:ec
Victim	fe80::b96c:c817:f160:767d	08:8f:c3:03:71:32

All we need to start the program is to download it with the command

```
git clone https://github.com/CrumblyBread/mitm-tools.git
```

or downloading the archive from this URL

```
https://github.com/CrumblyBread/mitm-tools
```

and navigating into the new folder with the terminal command

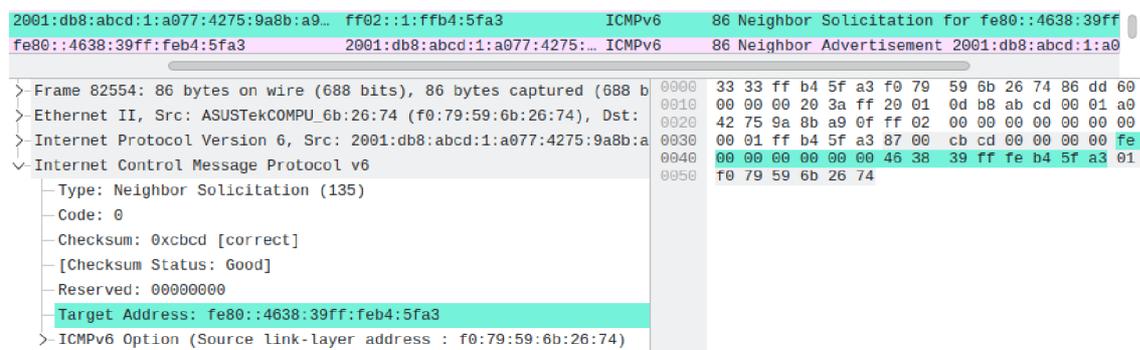
```
cd mitm-tools
```

now the script is ready to use. For installation of required dependencies see the file or get access to it on the aforementioned URL

2.5.1 Gateway impersonation mode

Before initiating the attack, the network operates under standard conditions where any IPv6 traffic generated by the victim device is transmitted directly to the default gateway. In this typical network configuration, the data packets follow a straightforward path from the victim to the designated gateway, without any interception or rerouting. Consequently, the attacker remains unable to access, monitor, or manipulate this traffic, as it bypasses the attacker's device entirely. The gateway impersonation mode was initialised by providing the chosen details from the addressing table to the application. The command used to start it was

```
sudo python mitm.py gateway -t fe80::b96c:c817:f160:767d
-gw fe80::1 -i enp7s0 -p
```



```
2001:db8:abcd:1:a077:4275:9a8b:a9... ff02::1:ffb4:5fa3 ICMPv6 86 Neighbor Solicitation for fe80::4638:39ff
fe80::4638:39ff:feb4:5fa3 2001:db8:abcd:1:a077:4275:... ICMPv6 86 Neighbor Advertisement 2001:db8:abcd:1:a0
> Frame 82554: 86 bytes on wire (688 bits), 86 bytes captured (688 b 0000 33 33 ff b4 5f a3 f0 79 59 6b 26 74 86 dd 60
> Ethernet II, Src: ASUSTekCOMPU_6b:26:74 (f0:79:59:6b:26:74), Dst: 0010 00 00 00 20 3a ff 20 01 0d b8 ab cd 00 01 a0
> Internet Protocol Version 6, Src: 2001:db8:abcd:1:a077:4275:9a8b:a 0020 42 75 9a 8b a9 0f ff 02 00 00 00 00 00 00 00
> Internet Control Message Protocol v6 0030 00 01 ff b4 5f a3 87 00 cb cd 00 00 00 00 fe
Type: Neighbor Solicitation (135) 0040 00 00 00 00 00 46 38 39 ff fe b4 5f a3 01
Code: 0 0050 f0 79 59 6b 26 74
Checksum: 0xcbbc [correct]
[Checksum Status: Good]
Reserved: 00000000
Target Address: fe80::4638:39ff:feb4:5fa3
> ICMPv6 Option (Source link-layer address : f0:79:59:6b:26:74)
```

Fig. 2.13: The Victims' device communicating with a virtual device

and functioned as described above. The application started by configuring its own parameters most importantly a randomly generated MAC address provided by the `get_random_mac()` function in Scapy. And also a link-local address using the EUI-64 process. The first packet it sent out was a Neighbour Solicitation packet requesting the MAC address from the target as displayed in Fig. 2.13. After the target has responded, it sent a similar request to the gateway. After both link-local and MAC addresses were established, the program sent the first Router Advertisement packets to the target. This changed the default route of the *Victim* device to the address of the *Attacker*. After this was verified on the target device, an ICMPv6 Echo Request was initialised on the *Victim* device and sent to the address of its DNS server. The message was correctly sent to the MAC address of the *Attacker*. You can see the captured message and its subsequently changed and forwarded version here in Fig. 2.16

```

16130 1332.7243440... fe80::1          fe80::b96c:c817:f160:767d  ICMPv6    70 Router Advertisement
16131 1332.7643422... fe80::4638:39ff:fed0:b391    fe80::b96c:c817:f160:767d  ICMPv6    70 Router Advertisement

```

```

> Frame 16130: 70 bytes on wire (560 bits), 70 bytes captured
> Ethernet II, Src: CompalInform_03:71:32 (08:8f:c3:03:71:32)
> Internet Protocol Version 6, Src: fe80::1, Dst: fe80::b96c:c817:f160:767d
< Internet Control Message Protocol v6
  -Type: Router Advertisement (134)
  -Code: 0
  -Checksum: 0x9347 [correct]
  -[Checksum Status: Good]
  -Cur hop limit: 0
  -Flags: 0x08, Prf (Default Router Preference): High
  -Router lifetime (s): 0
  -Reachable time (ms): 0
  -Retrans timer (ms): 0

```

Fig. 2.14: Sent Router Advertisement packets, with manipulated Router lifetimes

```

Ethernet adapter Ethernet 7:

Connection-specific DNS Suffix . . :
IPv6 Address. . . . . : 2001:db8:abcd:1:3928:b0f2:a76b:f792
Temporary IPv6 Address. . . . . : 2001:db8:abcd:1:a077:4275:9a8b:a90f
Link-local IPv6 Address . . . . . : fe80::b96c:c817:f160:767d%11
IPv4 Address. . . . . : 192.168.1.69
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::1%11

PS C:\Users\Branislav> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 7:

Connection-specific DNS Suffix . . :
IPv6 Address. . . . . : 2001:db8:abcd:1:3928:b0f2:a76b:f792
Temporary IPv6 Address. . . . . : 2001:db8:abcd:1:a077:4275:9a8b:a90f
Link-local IPv6 Address . . . . . : fe80::b96c:c817:f160:767d%11
IPv4 Address. . . . . : 192.168.1.69
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::4638:39ff:fea9:312f%11

```

Fig. 2.15: The Victim's configuration before and after and attack

After being received the packet is changed according to the script. Meaning the destination mac address is changed to the mac address of the virtual port generated by the attacker, and the IPv6 destination address is also changed with either the link local, or global address of the virtual device. Both the DNS query and the response were captured and documented in a automatically generated pcap file.

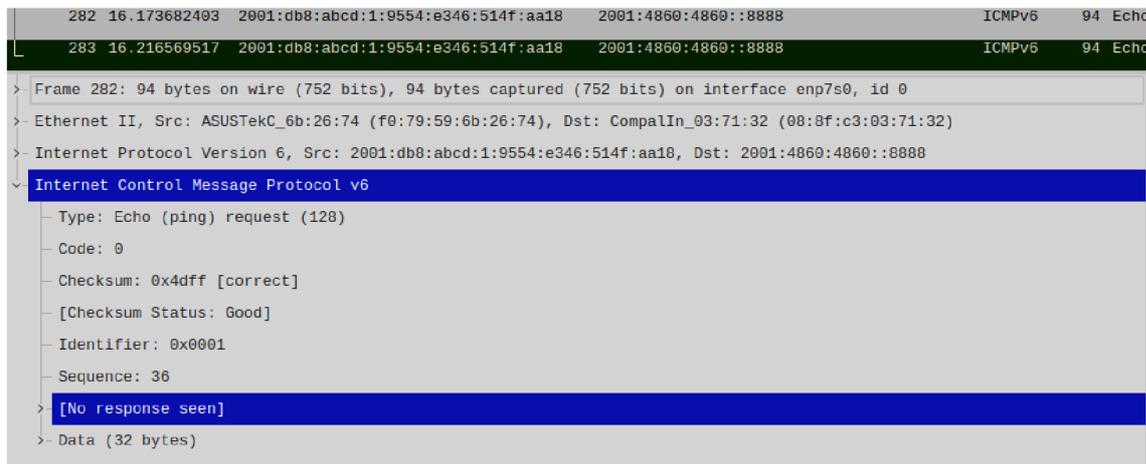


Fig. 2.16: Captured packet on the Attackers machine

2.5.2 DHCP server mode

The DHCP server underwent a meticulous configuration process, requiring details from the addressing table submitted in the *targets.txt* file, along with the creation of a DHCP address pool. The selected pool, denoted as `9999::/116`, was chosen to facilitate easy differentiation of addresses originating from this pool. To initiate the application in DHCP mode with this specific configuration, the following command was executed:

```
sudo python mitm.py dhcp -T targets.txt -n 9999::
-p 116 -i enp7s0
```

The application commenced by gathering its own parameters, including the MAC address and a link-local address. An IPv6 address pool is created from above mentioned parameters, beginning with the address `9999::1` and ending with the address `9999::fffe`. Subsequently, it initiated the process of collecting all MAC addresses corresponding to the link-local addresses specified in the *targets.txt* file. In this particular instance, only one address corresponding to the *Victim* device was provided in the *targets.txt* file.

```

CrumblyBread > sudo python mitm.py dhcp -T targets.txt -n 9999:: -p 116 -i enp7s0
Welcome to MITM-tools!
Device has been configured
Targets have been configured
Required processes have been started
ATTACK HAS BEEN STARTED (press any key then enter to quit)
Sending Router advertisements
Sending Router advertisements
.
Sent 1 packets.
fe80::b96c:c817:f160:767d is requesting an address
.
Sent 1 packets.
fe80::b96c:c817:f160:767d is requesting an address
Sending Router advertisements
Sending Router advertisements
█

```

Fig. 2.17: A screen capture of the application in use

Upon receiving a response from the target device, the program initiated its role as a DHCPv6 configuration server by advertising itself within the network. At this point, any DHCPv6 request transmitted by the target device triggered a corresponding response from the application. To validate the functionality, a command was executed on the target device to request a new DHCPv6 configuration. The resulting packet was accurately transmitted to the *Attacker* device, where the application responded to the request, seamlessly navigating through the entire DHCPv6 process as outlined in RFC-8415[4]. The entirety of this process was captured on the *Attacker* device, providing a comprehensive record of the DHCPv6 interactions, as depicted in Figure 2.18.

No.	Time	Source	Destination	Protocol	Length	Info
165	16.318175495	fe80::b96c:c817:f160:767d	ff02::1:2	DHCPv6	154	Solicit XID: 0xb63318 CID: 000100011de8b6e9f079596b2674
166	16.351784651	fe80::81c7:95db:d212:fb18	fe80::b96c:c817:f160:767d	DHCPv6	146	Advertise XID: 0xb63318 CID: 000100011de8b6e9f079596b2674 IAA: 9999::1
131	17.267125090	fe80::b96c:c817:f160:767d	ff02::1:2	DHCPv6	200	Request XID: 0xb63318 CID: 000100011de8b6e9f079596b2674 IAA: 9999::1
132	17.318383482	fe80::81c7:95db:d212:fb18	fe80::b96c:c817:f160:767d	DHCPv6	146	Reply XID: 0xb63318 CID: 000100011de8b6e9f079596b2674 IAA: 9999::1


```

> Frame 198: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface wlp9s20f3, id 0
> Ethernet II, Src: IntelCor_7c:5f:ec (c4:23:66:7c:5f:ec), Dst: ASUSTek_6b:26:74 (f0:79:59:6b:26:74)
> Internet Protocol Version 6, Src: fe80::81c7:95db:d212:fb18, Dst: fe80::b96c:c817:f160:767d
> User Datagram Protocol, Src Port: 547, Dst Port: 546
< DHCPv6
  Message type: Advertise (2)
  Transaction ID: 0xb63318
  Server Identifier
    Option: Server Identifier (2)
      Length: 14
      DUID: 000100012c3c75cc423667c5fec
      DUID Type: Link-layer address plus time (1)
      Hardware type: Ethernet (1)
      DUID Time: Nov 24, 2023 21:43:08.000000000 CET
      Link-layer address: c4:23:66:7c:5f:ec
  Client Identifier
    Option: Client Identifier (1)
      Length: 14
      DUID: 000100011de8b6e9f079596b2674
      DUID Type: Link-layer address plus time (1)
      Hardware type: Ethernet (1)
      DUID Time: Nov 25, 2015 19:13:52.000000000 CET
      Link-layer address: f0:79:59:6b:26:74
  Identity Association For Non-Temporary Address
    Option: Identity Association for Non-Temporary Address (3)
      Length: 40

```

Fig. 2.18: Captured DHCP communication on the Attacker's machine

The successful configuration was validated on the target device to ensure seamless integration and optimal functionality. The target device aptly utilized all the parameters provided by the application, demonstrating a comprehensive implementation of the configured settings. For detailed insight into the entire configuration process and parameters employed, refer to the comprehensive configuration documentation provided by the application, which is available for examination and verification. After the attack is executed, all IPv6 traffic originating from the victim device is no longer directed to the default gateway. Instead, it is routed to the address generated by the attacker's device. This redirection signifies a successful interception of the network traffic, positioning the attacker at a critical juncture within the communication pathway. As a result, the attacker gains the capability to monitor, record, manipulate, or even completely block the traffic at will.

```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ipconfig /renew6 "Ethernet 7"

Windows IP Configuration

Unknown adapter Lokálne pripojenie 3:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 7:

    Connection-specific DNS Suffix  . :
    IPv6 Address. . . . . : 2001:db8:abcd:1:3928:b0f2:a76b:f792
    IPv6 Address. . . . . : 9999::1
    Temporary IPv6 Address. . . . . : 2001:db8:abcd:1:15a2:2cd2:2d13:e791
    Link-local IPv6 Address . . . . . : fe80::b96c:c817:f160:767d%10
    IPv4 Address. . . . . : 192.168.1.69
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::1%10
                                192.168.1.1

C:\WINDOWS\system32>
```

Fig. 2.19: Configured address on Victims machine

There is also a potential case, where there are more DHCPv6 servers in the network. This creates the potential situation where the other server might see the DHCPv6 Advertise packet and respond before the Attackers device has a chance to do so. In this case there is nothing an attacker can do to take priority.

2.5.3 DNS server mode

Before initiating a DNS MITM attack, the network operates under standard conditions where any DNS queries generated by the victim device are transmitted directly to the legitimate DNS server specified in the network configuration. In this typical setup, DNS queries follow a straightforward path from the victim to the DNS server without any interception or rerouting. Consequently, we are unable to access, monitor, or manipulate these queries, as they bypass our device entirely.

To execute the DNS MITM attack, we configure our device to impersonate the legitimate DNS server. This is achieved by utilizing the details provided in the addressing table to craft and send spoofed DNS responses.

```
sudo python mitm.py dns -T targets.txt -dns dns.txt
-i enp7s0
```

This command configures the necessary parameters for the operation. The targets.txt file is parsed to extract all specified target addresses, which, in this instance, includes only one address. Additionally, the dns.txt file is converted into a Python dictionary, formatted according to the specifications required by the Scapy library. The application needs to know the full MAC, link-local and global unicast addresses of the victim to successfully capture any DNS request.

```
CrumblyBread > sudo python mitm.py dns -T targets.txt -dns dns.txt -i enp7s0
Welcome to MITM-tools!
Device has been configured
Targets have been configured
Required processes have been started
ATTACK HAS BEEN STARTED (press any key then enter to quit)
Sending Router advertisements
Sending Router advertisements
Sending Router advertisements
fe80::b96c:c817:f160:767d sent a Query
Sending Router advertisements
Sending Router advertisements
Sending Router advertisements
q
Quitting
CrumblyBread > █
```

Fig. 2.20: A screen capture of the application providing a DNS response

After the application configured its parameters, it initiated the attack sequence. The first step involved compromising the target device by dispatching a meticulously crafted Router Advertisement packet that included the **Recursive DNS Server** (RDNSS) option. This option contained the address of the attacker's device in the DNS server field. Consequently, the target device now erroneously recognized

the attacker's device as a legitimate DNS server, purportedly provided by the default gateway. This deceptive maneuver ensured that any subsequent DNS queries originating from the target device would be directed to the attacker's device for resolution.

```

2001:db8:abcd:1:a077:4275:9a8b:a9... 2001:db8:abcd:1:8fed:544e:... DNS      86 Standard query 0x0002 AAAA vut.cz
-----
> Frame 1800: 86 bytes on wire (688 bits), 86 bytes captured (688 bi
> Ethernet II, Src: ASUSTekCOMPU_6b:26:74 (f0:79:59:6b:26:74), Dst:
> Internet Protocol Version 6, Src: 2001:db8:abcd:1:a077:4275:9a8b:a
> User Datagram Protocol, Src Port: 57530, Dst Port: 53
Domain Name System (query)
  Transaction ID: 0x0002
  Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    > vut.cz: type AAAA, class IN
0000 08 8f c3 03 71 32 f0 79 59 6b 26 74 86 dd 60
0010 1b 1b 00 20 11 40 20 01 0d b8 ab cd 00 01 a9
0020 42 75 9a 8b a9 0f 20 01 0d b8 ab cd 00 01 8f
0030 54 4e c3 81 a2 42 e0 ba 00 35 00 29 04 b9 00
0040 01 00 00 01 00 00 00 00 00 00 03 76 75 74 02
0050 7a 00 00 1c 00 01

```

Fig. 2.21: DNS query from the victim on the attackers device for the domain vut.cz

Upon intercepting a DNS query, the attacker utilized an Answering-Machine algorithm, which is integrated within the Scapy framework, to generate an appropriate response. This algorithm is designed to process and respond to DNS queries based on a dns file supplied by the user. It also considers various nuances, including the specific options that may be appended to the DNS query. By meticulously crafting responses that adhere to these parameters, the algorithm effectively maintains the illusion of legitimacy, thereby facilitating the continuation of the attack. During the attack, the target device, believing it was communicating with a trusted DNS server, would resolve domain names to IP addresses provided by the attacker. This allowed the attacker to redirect the target device's traffic to malicious sites, intercept sensitive information, or even facilitate further attacks such as phishing or malware distribution. From the perspective of a client operating the victim device, no change in behavior, or appearance was detectable. The device continued to operate as expected, with network connectivity seemingly intact. However when the client's browser attempted to load web pages the browser displayed a "page not found" error. This outcome was observed only during testing, where the address provided in the DNS query response was intentionally left unassigned to monitor the attack's effects without causing actual harm. In a real-world scenario, the attack would be significantly more insidious and difficult to detect. The malicious application would provide legitimate responses to DNS queries, redirecting the client's browser to seemingly authentic, but attacker-controlled, websites. These sites could mimic legitimate ones, capturing sensitive information such as login credentials or injecting malicious content into the client's device.

2.6 Future Plans

Future plans for the development of the MITM-tools include enhancing its functionality and expanding its feature set, improving detection and evasion techniques, and focusing on usability and community engagement. To begin with, the tool will be extended to support additional network protocols beyond IPv6, ICMPv6, and DHCPv6, such as HTTP/HTTPS and DNS over HTTPS (DoH), along with implementing more sophisticated attack techniques like SSL stripping and advanced DNS spoofing. In terms of detection and evasion, advanced stealth enhancements will be developed to bypass modern intrusion detection and prevention systems (IDS/IPS), mimicking legitimate traffic patterns and utilizing encryption to conceal malicious activities. Anti-detection measures will also be integrated to neutralize countermeasures deployed by network security systems. Usability improvements will involve the creation of a user-friendly graphical interface to simplify configuration and deployment, making the tool accessible to users with varying levels of technical expertise. Additionally, automation features and scripting capabilities will be introduced for predefined attack scenarios. Finally, community engagement and collaboration will be emphasized through open-source contributions, fostering community input and collaborative development, and establishing partnerships with academic institutions and industry leaders to drive further research and development in network security.

Conclusion

In concluding this thesis, the journey from the inception to the development of the Python application for MITM attacks marks a significant exploration at the intersection of programming, networking, and cybersecurity. The creation of this application represents a unique endeavor to offer a versatile tool capable of integrating various attack methods into a unified solution.

Throughout the research, the Scapy library's proficiency in crafting and manipulating network packets has been harnessed, coupled with an in-depth understanding of IPv6, ICMPv6, and DHCPv6 protocols. The application's focus on fake DNS, DHCP, and default gateway attacks underscores its potential as a comprehensive toolkit for security professionals and researchers.

As the project evolves, future plans center around refining the application, expanding its capabilities, and addressing challenges identified during testing. Emphasis will be placed on ethical considerations, user responsibility, and collaboration within the cybersecurity community. The envisioned trajectory seeks to keep pace with emerging networking technologies, ensuring the application's relevance in dynamic cybersecurity landscapes.

The commitment to ethical use, ongoing development, and community collaboration aims to establish the application as a valuable resource for educational purposes and responsible penetration testing. The continuous evolution of the tool aligns with the broader goal of contributing to the field of network security and empowering cybersecurity practitioners with effective, responsible, and adaptable solutions.

Bibliography

- [1] Dr. Steve E. Deering and Bob Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, July 2017. URL: <https://www.rfc-editor.org/info/rfc8200>, doi:10.17487/RFC8200.
- [2] Marlon A. Naagas and Anazel P. Gamilla. Denial of service attack: an analysis to ipv6 extension headers security nightmares. *International Journal of Electrical and Computer Engineering (IJECE)*, 2022. URL: <https://api.semanticscholar.org/CorpusID:247456764>.
- [3] Hao Wu, Xianglei Dang, Lidong Wang, and Longtao He. Information fusion-based method for distributed domain name system cache poisoning attack detection and identification. *IET Inf. Secur.*, 10(1):37–44, January 2016.
- [4] Tomek Mrugalski, Marcin Siodelski, Bernie Volz, Andrew Yourtchenko, Michael Richardson, Sheng Jiang, Ted Lemon, and Timothy Winters. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 8415, November 2018. URL: <https://www.rfc-editor.org/info/rfc8415>, doi:10.17487/RFC8415.
- [5] Mohamed Abdallah Elakrat and Jae Cheon Jung. Development of field programmable gate array-based encryption module to mitigate man-in-the-middle attack for nuclear power plant data communication network. *Nucl. Eng. Technol.*, 50(5):780–787, June 2018.
- [6] Liumei Zhang, Yu Han, Yichuan Wang, and Ruiqin Quan. Petri net model of mitm attack based on ndp protocol. In *2022 International Conference on Networking and Network Applications (NaNA)*, pages 402–405, 2022. doi:10.1109/NaNA56854.2022.00074.
- [7] Alexandru Lucian Petrescu, Mohamed Boucadair, and Leaf Y. Yeh. Route problem at relay during dhcpv6 prefix delegation. 2013. URL: <https://api.semanticscholar.org/CorpusID:64545539>.
- [8] Amir Herzberg and Haya Shulman. Retrofitting security into network protocols: The case of DNSSEC. *IEEE Internet Comput.*, 18(1):66–71, January 2014.
- [9] Dave Kuhlman. *A Python Book: Beginning Python, Advanced Python, and Python Exercises*, June 2012. URL: https://www.davekuhlman.org/python_book_01.pdf.
- [10] William A. Simpson, Dr. Thomas Narten, Erik Nordmark, and Hesham Soliman. Neighbor discovery for ip version 6 (ipv6). RFC 4861, September

2007. URL: <https://www.rfc-editor.org/info/rfc4861>, doi:10.17487/RFC4861.

- [11] Syam Madanapalli, Jaehoon Paul Jeong, Soohong Daniel Park, and Luc Be-
loeil. IPv6 Router Advertisement Options for DNS Configuration. RFC
6106, November 2010. URL: <https://www.rfc-editor.org/info/rfc6106>,
doi:10.17487/RFC6106.

Symbols and abbreviations

IPv6	Internet Protocol version 6
MITM	Man-in-the-middle
DHCPv6	The Dynamic Host Configuration Protocol version 6
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
mDNS	Multicast DNS
Zeroconf	Zero Configuration Networking
ARP	Address Resolution Protocol
NDP	Neighbour Discovery Protocol
MAC	Media Access Control
SSL	Secure Sockets Layer
TLS	Transport Layer Security

List of appendices

A MITM-tools Application

63

A MITM-tools Application

```
/.....root of the attached archive
├── mitm.py.....The main MITM-tools script
├── classes.py.....A supporting script containing the used classes
├── dhcpAM.py.....Script containing the DHCPv6 answering algorithm
├── dnsAM.py.....Script containing the DNS answering algorithm
├── targets.txt.....An example of a Targets file
├── dns.txt.....An example of a DNS dictionary
├── requirements.txt.....The list of all needed dependencies
├── README.md.....The documentation for the code
└── LICENSE.....The licence agreement
```