Cyber Security Attacks to the Network Infrastructure

Duarte Simões Matias
duarte.matias@tecnico.ulisboa.pt
Supervisor: Professor Rui Valadas
Instituto Superior Técnico, Lisboa, Portugal

October 2022

Abstract

Cybersecurity has been evolving at an incredible speed in the last two decades, with the number of software vulnerabilities reported daily increasing at a fast pace. Creating secure software is so important that most companies changed from a DevOps model to a DevSecOps model where security experts are incorporated into the software development teams.

But while securing the software itself is important, securing the underlying layers that run that software is also crucial. One such layer is the so-called network infrastructure, comprised of devices like routers and switches. The devices tend to be running protocols that were created decades ago, some of them having their first specifications as old as the Internet itself, meaning they were created in an era where security was not a primary concern. While many of these have had updates over the past years to bring them up to standards, many entities continue utilizing obsolete versions of them.

While some small and simple tools exist as proof-of-concept for these attacks, there is no single tool out there that combines all kinds of attacks that can be launched against the network infrastructure.

In this MSc dissertation, we built a tool that provides a variety of attacks to test the security of the network infrastructure. The tool is built utilizing the Python programming language and one of its libraries, Scapy. It runs in a Docker container and therefore is easy to install and deploy in a variety of environments.

The tool supports attacks against protocols related to the OSI model layer 2 (ARP, Switches, STP, VLAN, DHCP), IPv4 routing protocols (RIP, OSPF, BGP), and others like DNS and ICMP, while also maintaining extensibility for extra attacks and protocols to be easily added. This MSc dissertation was supported by Instituto de Telecomunicações

Keywords: Python, Scapy, Network Infrastructure Attacks

1. Introduction

Nowadays society is progressively more and more dependent on Internet access to function properly. Be it in healthcare, finance, or entertainment, every business has a certain degree of reliance on the Internet to operate smoothly. As such, the need to secure systems and networks from unwanted access by external individuals has also increased, with security researchers having a crucial role in finding and reporting exploits that can lead to security breaches.

However, most entities still focus on exploiting software vulnerabilities present on endpoint machines, such as desktops and servers, and forget that there's another point of failure in the enormous organism that is the Internet: the network infrastructure, devices like routers and switches, which carry information from endpoint to endpoint, or services like DNS and DHCP, that simplify the overall experience of utilizing the Internet. Devices like these are vital for the Internet to function smoothly.

While a lot of different tools exist to find and exploit software vulnerabilities in endpoints, the same cannot be said for the network infrastructure. In fact, the list of tools to test protocol vulnerabilities for this infrastructure is comparatively shorter, with each tool performing a much smaller number of tests than their endpoint counterparts.

Tools like arpspoof and macof, part of the dsniff suite [10], which only perform one type of attack, or yersinia [3], which contains related layer 2 attacks on STP and VLANs, or even ettercap [1], which is utilized in DHCP attacks and all forms of MitM attacks performed in a local network, are examples of tools that can test network infrastructure.

However, such tools are few and far between, and by looking at tools to perform route injection on various routing protocols one can conclude the list is even shorter, featuring names like vRIN [4], a route injection tool that performs the most basic of route injections, or OSV [7], an automated network testing tool that runs several fixed vulnerability checks on an OSPF network.

Moreover, these tools often offer a limited num-

ber of attack options, making it hard to launch customized attacks, and often run into problems when an uncommon option is needed to perform the attack.

This MSc dissertation aims to compile a list of known vulnerabilities for the protocols running on devices considered part of the network infrastructure, and then create a tool that allows the infrastructure to be tested against all these vulnerabilities while allowing the user to chain different attacks. The created tool must provide an extensible interface for extra attacks to be added in the future, and any attack must be coded in a way where a user can extract the attack code from the program files and be able to run it with minimal modifications.

1.1. Contributions

The contributions of this project include the development of a tool in Python, leveraging the Scapy library, for performing extensive testing in the network infrastructure. The tool provides freedom for the user to decide what tests to perform and in what order they are executed, giving the user control over all the details for each test in a user-friendly interface. The user also has the ability to write customizable tests for their target topologies and import their configurations. Our tool provides an extensive repertoire of tests across different protocols to test various services in the network. The attack list includes attacks on protocols like ARP, STP, VLANs, DTP, DHCP, RIP, OSPF, BGP, DNS, and ICMP, as well as the layer 2 switches' CAM tables.

1.2. Implemented Attack List

Our developed tool includes all the attacks listed below.

- Layer 2
  - ARP
    - ARP Spoofing
  - Switches
    - CAM Overflow
  - STP
    - Root Bridge Hijack
    - Conf BPDU DoS
    - TCN BPDU DoS
    - Eternal Root Election
    - Root Bridge Disappearance
  - VLANs
    - Double Tagging
    - DTP Negotiation attack
    - PVLAN Proxy
  - DHCP
    - DHCP Starvation
    - DHCP Spoofing
- Routing Protocols
  - RIP
    - RIP Route Injection
    - RIP Request DoS
  - OSPF
    - Remote False Adjacency
    - Disguised LSA
    - Single Path Injection
    - Max Age LSA attack
    - Seq++ attack
    - Max Seq# attack
  - BGP
    - BGP Route Injection or BGP Hijack
- Other Protocols
  - ICMP
    - ICMP Flooding
    - ICMP Redirection
  - DNS
    - DNS Spoofing

1.3. Testing Environment

To test the implemented attacks we use the GNS3 network simulator [2] in conjunction with a set of Cisco routers, namely the C3725 and C7200 routers. To capture and inspect packets circulating in the test topologies, we utilize the Wireshark packet capture tool.

1.4. Structure

This article explores the theory behind three of the implemented attacks, namely VLAN Double Tagging (Section 2), OSPF Remote False Adjacency (Section 3), and ICMP Redirection (Section 4). These attacks cover the most important concepts utilized in our tool, such as the concept of chaining, importing attack configurations from configuration files, and modifying operative system settings to permit some attack to function properly. In the end we present our conclusions and further work.

2. Attacks to layer 2 - VLAN Double Tagging

2.1. Attack Description

One key concept of the developed tool is attack chaining. A chain is a sequence of attacks. These attacks can either be launched in succession or be used to modify the behavior of each other. The latter represents our concept of "Middleware", a function that modifies how the next attack behaves. The

VLAN Double Tagging is categorized as “Middleware” as it inserts certain headers into a packet which modifies the way the packet will be routed in the network. The VLAN Double Tagging attack allows an attacker to perform VLAN hopping, that is, sending packets from its VLAN to another one without requiring the frame to be sent to a multilayer switch. A VLAN, or Virtual LAN, is a form of network isolation utilized in local networks to separate traffic from different machines as if further segregating traffic within a subnetwork. Machines on one VLAN can’t communicate with another VLAN without utilizing a router. VLAN Double Tagging permits an attacker to circumvent the need for a router for sending packets to another VLAN. In order to perform the attack, frames must be sent with two distinct 802.1Q headers (responsible for carrying VLAN information), one with the VLAN ID of the attacker (the ‘outer tag’) and another one with the VLAN ID of the destination machine (the ‘inner tag’). When the frame arrives at the first switch, the switch strips the outer tag (as VLAN information coming from the user should be ignored) and then forwards the frame to the next switch while keeping the inner tag. When the frame arrives on the last switch before being delivered to the target, the switch will read the inner tag and forward the frame to the target’s VLAN, leading to the frame being delivered to the victim. The major downside of this attack rests on the fact that it’s a unidirectional process: while the attacker can deliver packets to the target, the inverse is impossible as the target has no way of double tagging the response. In any case, it can still be used to perform DoS attacks that only require unidirectional communication. VLAN Double Tagging can be used as a way to launch a DoS attack onto a host that would otherwise be unreachable, such as launching an ICMP flooding attack against a server residing on an isolated VLAN or performing a DHCP starvation attack against hosts on a different VLAN.

2.2. Attack Code

The VLAN Double Tagging attack is defined in the ”vlan.py” file, under the ”layer2” directory, by the ”vlan_double_tagging” function. The function code is show in Figure 1.

The code is written in Python and utilizes the Scapy library to create and manipulate the different layers that constitute the packets. In this case, we see the creation of a Dot3 header, two Dot1Q headers for VLAN information, and an IP header.

Figure 1: Function code for the VLAN Double Tagging attack

As a middleware, the function’s objective is to insert the necessary headers in the packet which will then be further modified. As such, the function modifies the ”pkt” key in the ”args” variable, setting up the necessary headers with the information to carry the attack, and returning the ”args” variable with the updated packet. The ”pkt” key can then be accessed by another attack function executed after, and will contain the layers created in during the execution of this function.

2.3. Testing and Validation

Figure 2 shows the topology utilized to test the attack. Both R1 and R2 are c3725 routers fitted with an NM-16ESW module. This module enables the router to behave like a switch, and with IP routing deactivated they only behave like switches (no inter-VLAN routing). R4 behaves like a host, having its ‘f0/0’ interface configured with the IP address 11.11.11.4/24. No IP configuration was performed on the attacker machine. The objective of for this test is to verify if our attacker machine can communicate with R4 who resides on a different VLAN.

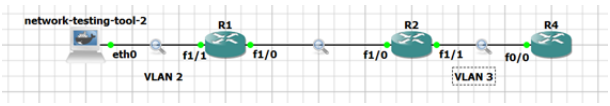

Figure 2: VLAN Double Tagging attack test topology

When launching the tool, we are greeted with the menu shown in figure 3. This menu gives us, the user, the possibility of choosing what attacks to perform, from what interface, and import extra configurations for any attack if needed.

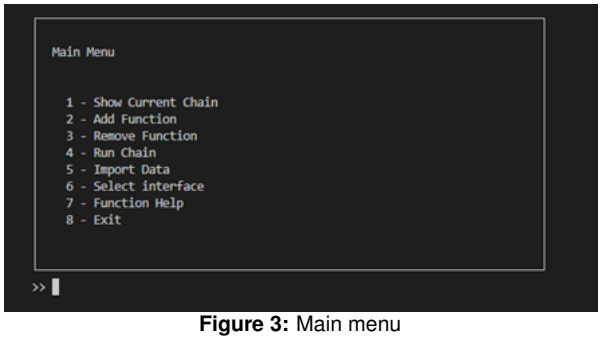

Figure 3: Main menu

To perform the attack, we first select the interface ”eth0” in the ”Select Interface” sub-menu, and then chain two functions: first, by selecting “Middleware”, “L2” and “Wrapper” we can filter for the “Double Tagging” function; and second, selecting “Attack”, “L4” and “Ping Test” we can filter for the “Ping Test” function, which sends a single ICMP Echo Request to the target. We can then launch the attack by selecting the “Run Chain” option, where we are prompted for the missing parameters, i.e., the VLAN tags for outer and inner head-

ers (for the Double Tagging attack) and the destination IP (for the Ping test), and after which we start sending attack packets. In this case, we set the outer VLAN to the value “2”, the inner VLAN to the value “3”, and the destination IP is the IP address of R4, “11.11.11.4”.

The only way we have of detecting whether the attack was successful or not is by attaching a Wireshark probe to the interface on R4. When the attack runs, the packet shown in figure 4 is captured by Wireshark. This packet corresponds to the ICMP request sent by the attacker, stripped of the VLAN-related headers.

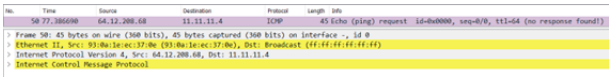

Figure 4: VLAN Double Tagging captured packet at R4’s interface

3. Attacks on OSPF - Remote False Adjacency

3.1. Protocol Description

OSPF, defined in RFC 2328 [8], is an intra-domain link-state routing protocol. OSPF router exchange topology information in the form of LSAs and build a topology from all the collected LSAs, which are utilized to calculate the routing tables.

OSPF routers establish an adjacency relation between themselves, which allows them to exchange their database information and detect link failures, modifying the link’s state to reflect topology changes. This adjacency is established utilizing the Hello protocol. Routers will periodically send Hello messages from their interfaces which contain basic OSPF information, such as the list of connected OSPF routers in the local subnet.

When two non-adjacent routers receive Hello packets from each other, they establish an adjacency relation between themselves and exchange the details of their databases in a process called LSDB synchronization. During this stage, also called DBD exchange, routers exchange the headers of LSAs present in their databases utilizing DBD packets.

When the DBD exchange phase is finished, routers update existing LSAs to reflect the new topology. When an adjacency is established on a transit link, one of the OSPF routers is nominated the DR, which is the router responsible for generating the Network LSA for the newly created transit link.

OSPF LSAs are transmitted inside packets called Link State Updates, or LSUs, that allow more LSAs to be transmitted at once. The LSUs need to be acknowledged by the destination router, which will send an LSAck packet containing the headers of the received LSAs.

OSPF implements a natural fightback mechanism when wrong information is present in an LSA. When a wrong LSA reaches the router which generated it, the router will correct the information in the LSA and flood it so all the domain routers can update its information. This mechanism makes it extremely hard for an attacker to inject false information into an OSPF network, as LSAs are only accepted if they are either created or flooded by an adjacent router.

3.2. Attack Description

The Remote False Adjacency, first described by G. Nakibly et. al. [6], is an OSPF route injection attack that injects LSAs utilizing a phantom router.

In order for an LSA to be accepted, it must be sent by an active OSPF neighbor. This means injection attacks are more complicated than in RIP.

The concept of a phantom router was first explained by E. Jones et. al [5], and describes an OSPF neighbor that isn’t attached to a physical router. In other words, it’s a form of spoofing an adjacency relation with the Designated Router (DR) so that LSAs can be sent to the DR with a source IP address of a machine that doesn’t exist in the network. Configuring a phantom isn’t easy because, if an attacker isn’t in the local subnet, there is no access to the packets sent by the DR destined for the phantom.

The Remote False Adjacency attack utilizes a phantom router to inject LSAs into the network. The point of the attack is to have an attacker configure a phantom router remotely, i.e., from a different subnetwork, and then send all LSAs in unicast to the DR. The difficult part of the attack is the lack of bi-directional communication between the DR and the attacker: packets sent by the attacker have their source IP address spoofed so that, from the DR’s point of view, the packets received come from the local subnetwork. Consequently, the attacker has no access to the packets sent by the DR to the phantom, as those will be destined to an IP address in the DR’s subnet, and not the attacker.

However, there’s a workaround for this. Since most of the adjacency setup process is deterministic, it can be finished without the attacker ever receiving a response from the DR. Every message sent by the attacker must be sent in unicast, and the interface on the DR that receives the packets must be connected to the local subnet where the phantom is created.

The attack begins by having the attacker send a hello packet destined for the DR, containing the DR’s router ID in the neighbor list, with the source IP of the phantom. The authors note that the router ID for the phantom needs to be higher than that of the DR for the attack to be successful (explained below). These hello messages need to be re-sent

periodically to keep the adjacency from being torn down.

The next step is to send the DBD messages. Usually, the DR is the master in the database exchange phase. However, since the phantom's ID is higher than the DR's, the phantom can perform the whole exchange process as the master. This allows the attacker to set a custom value for the sequence number utilized in the DBD messages, which would be impossible to guess if the DR was the master as the attacker doesn't have access to the DBD messages sent by the DR. The authors note that the number of DBD messages can be as high as the attacker wants, but it needs to at least cover the number of headers sent by the DR, meaning the attacker can send more DBD messages than needed for the exchange phase, but never less. The authors also note this number can be easily estimated if the attacker can see OSPF messages in the local network. Certain flags need to be set on the DBD messages: every message must have the master (MS) flag set, the first message needs to have both the I and M bits set, and every message after except for the last one must have the M bit set. Once the last DBD message arrives at the DR the adjacency is considered established, and routers can now exchange LSAs.

The last step of the attack is to send fake LSAs with the source IP address and router ID of the phantom, also in unicast, to the DR. Once delivered, the DR immediately proceeds to flood the LSAs to every connected OSPF router.

There is one extra step that can only be taken if the attacker can read OSPF LSU packets from the directly connected link, and that is acknowledging the newly generated network LSA. The DR will generate a new Network LSA for the local subnet to add the phantom to the list of attached routers. This LSA needs to be acknowledged by all routers, including the phantom, or the DR will tear down the adjacency relation created. Thus, if the attacker can sniff LSUs from the connected network, it must obtain the new Network LSA and send an LSAck on behalf of the phantom.

Implementation of this attack was relatively more complex than the VLAN attack explained before, as the attack requires more steps to perform, large quantities of routing information to be imported, and parallel tasks for periodically sending Hello messages while performing the adjacency setup. However, this is irrelevant from the end user's perspective as the entire process is simplified by having the different menu options execute the distinct steps needed to perform the attack.

This attack can inject all types of OSPF LSAs into the network, meaning the user can manipulate a vulnerable network at will. Our tool includes support for injecting not only all relevant LSA types, from type 1 to type 7, but also multiple LSAs originating from multiple machines at the same time, i.e., in the same LSU. These LSAs can be described and imported by the user utilizing the YAML format, meaning the user can create custom attacks for their specific topologies. Figure 5 shows an example of such a file, which contains definitions for two distinct router LSAs and one network LSA, whose successful injection allows the attacker to create an entire new subnet in the OSPF domain.

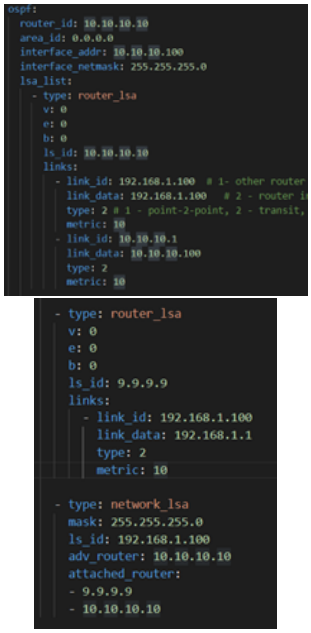

Figure 5: YAML File describing OSPF LSAs

3.3. Testing and Validating

Figure 6 shows the topology utilized to test the Remote False Adjacency attack. All routers are Cisco 7200 devices, fitted with C7200-IO-2FE slots. All routers have OSPF configured to announce their directly connected subnets. R1 is made the DR for both subnets it is connected to utilizing a higher priority value. All router IDs are based on the router number, i.e., R1's OSPF ID is 1.1.1.1, R2's OSPF ID is 2.2.2.2, . . .

The objective of this test is to create a phantom in the 10.10.10.0/24 subnet, connected to Switch3, with IP address 10.10.10.100 and router ID 10.10.10.10, which will then inject a fake route to a subnet 192.168.1.0/24, a transit network with two OSPF routers: the phantom and another nonexistent router. Figure 5 shows the YAML file imported during the attack.

To launch the attack, we start by selecting interface "eth0" in the "Select Interface" sub-menu. We import the routing configurations from the YAML file by selecting "Import Data", "OSPF LSU/LSA",

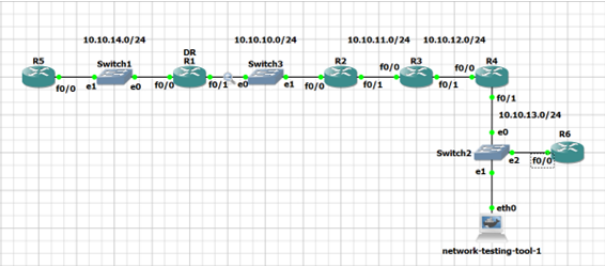

Figure 6: Remote False Adjacency test topology

and “ospf_rfa_test_1.yml”. Then, we add the attack function by selecting “Add Function”, “Attack”, “L3-OSPF”, “Route Poisoning”, and “Remote False Adjacency”. Finally, we launch the attack by selecting “Run Chain” and, when prompted, introducing the values 10.10.10.1 for the “victim_IP” and 1.1.1.1 for the “victim_id”, corresponding to the DR IP address and router ID. Having to know this information beforehand turns the attack from an outsider’s point of view harder, however, this information can be obtained by having the attacker attach itself to a transit link and wait for the Router and Network LSAs of the DR to be naturally refreshed and flooded.

In order to verify if the attack is correctly implemented we can observe the routing table in any domain router. In this case we decided to inspect the table of R1, shown in Figure 7, where we can see the injected route to subnet 192.168.1.0/24

```
R1#sh ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
C        10.10.10.0/24 is directly connected, FastEthernet0/1
L        10.10.10.1/32 is directly connected, FastEthernet0/1
O        10.10.11.0/24 [110/2] via 10.10.10.2, 00:05:26, FastEthernet0/1
O        10.10.12.0/24 [110/3] via 10.10.10.2, 00:05:26, FastEthernet0/1
O        10.10.13.0/24 [110/4] via 10.10.10.2, 00:05:26, FastEthernet0/1
C        10.10.14.0/24 is directly connected, FastEthernet0/0
L        10.10.14.1/32 is directly connected, FastEthernet0/0
O     192.168.1.0/24 [110/11] via 10.10.10.100, 00:04:19, FastEthernet0/1
R1#
```

Figure 7: R1’s Routing table

4. Attacks on other protocols – ICMP Redirection

ICMP is a protocol used by IP devices to transmit operational information between themselves. Defined in RFC 792 [9], ICMP is the protocol behind the ping and traceroute commands, as well as informing a host that a certain destination is unreachable.

Although considered part of the internet layer in the TCP/IP model, ICMP messages are always sent within an IP packet.

ICMP headers have three fixed elements: the ICMP type, code, and checksum. The remainder of the header is dependent on the type and code. Common ICMP messages include Echo Requests (type 8, code 0) and Echo Replies (type 0, code 0) utilized by ping and traceroute commands, and Destination Unreachable (type 3, variable code).

ICMP Redirection is a MitM attack that works with one of the ends residing in the local network. The other machine must be in a different subnetwork, accessible only with the use of a gateway. ICMP Redirect packets are traditionally utilized by routers to notify hosts of existing alternate routes and update their routing information. They are usually sent when a more direct route to a destination exists than the one currently being used.

An ICMP Redirection attack utilizes ICMP Redirect messages to redirect traffic in the local network from the default gateway to the attacker’s machine. To launch this attack, the attacker needs to send an ICMP packet with the following characteristics:

- IP Source is the default gateway IP address
- IP Destination is the victim’s IP address
- ICMP Type 5
- ICMP Code 1
- The gateway address is the IP address of the attacker
- Inner IP Header with the victim’s source IP address
- Inner IP Header with the target’s destination IP address

Before sending the packet, the attacker needs to enable IP forwarding on its machine, disable ICMP redirects, and configure a NAT rule to replace the source and destination IP addresses on packets sent to the attacker from the victim.

When our tool runs the attack, it also writes every packet redirected on a packet capture file, which can later be visualized using appropriate software like WireShark.

The ICMP redirection attack is an example of an attack which needs to perform write operations in a file and several configuration modifications in the operative system for it to successfully run.

5. Conclusions

In this report, we describe our tool for testing attacks on the network infrastructure. This tool includes a wide range of already implemented attacks for different protocols across many layers of the TCP/IP model, while also supplying a developer with an interface to add more attacks and protocols. We have effectively created a framework for a penetration testing tool that is easy to expand, providing a way to chain many different attacks on

different protocols to test the most robust and complex network topologies. The program is also easily deployable in any environment due to its Docker image, which permits professionals and students alike to spend less time installing and more time testing and learning. The tool includes all the attacks listed in the introduction section, as well as the configuration files utilized to test the attacks in their corresponding test topologies.

Further work includes the expansion of the attack list. While we tried to include as many attacks on as many protocols as possible, new attacks are always being published and thus can also be added to the collection. The protocol list can also be expanded to include other network protocols, such as IS-IS for intra-domain routing, GRE, IPSEC, and VPNs for tunneling protocols, or even other protocols such as PPP and IGMP, to name a few.

This also includes the modification of existing attacks. Attacks on protocols that implement security measures like authentication and encryption can be modified to include support for such measures. An example of this is the attacks on OSPF, which can be modified to include an authentication header in the crafted packets.

Some attacks have limitations related to the way they are implemented. Attacks like DHCP spoofing, DNS spoofing, and some OSPF route injection attacks like the Disguised LSA attack can be modified to provide more customization by adding extra options to the created in the attack code.

One more interesting option for further work would be the inclusion of IPv6-compatible attacks. Some attacks described in this report already work for IPv6 if modified, and thus exploring these options would be valuable. This would also include the addition of IPv6-exclusive protocols like NDP, for example.

Finally, the inclusion of “Middleware” functions that provide firewall and IDS evasion would also be beneficial for network testing, as they allow an administrator to better understand the flaws of the current configurations.

References

[1] Ettercap home page. https://www.ettercap-project.org/. Accessed: 9/1/2022.

[2] Gns3 — the software that empowers network professionals. www.gns3.com.

[3] tomac/yersinia: A framework for layer 2 attacks. https://github.com/tomac/yersinia. Accessed: 6/1/2022.

[4] A. Dosztal. vrin - virtual route injector. https://dosztal.com/static/vrin.

[5] O. L. M. E. Jones. Ospf security vulnerabilities, 2016.

[6] D. G. G. Nakibly, A. Kirshon and D. Boneh. Persistent ospf attacks, 2012.

[7] P. Kasemsuwan and V. Visoottiviseth. Osv: Ospf vulnerability checking tool. pages 1–6, 2017.

[8] J. Moy. Ospf version 2.

[9] J. Postel. Internet control message protocol, 1981.

[10] D. Song. dsniff. https://www.monkey.org/ dugsong/dsniff/. Accessed 5/1/2022.