University of Amsterdam

MSc System and Network Engineering

# Research Project 2

---

# Gaining unauthorised access to an 802.1X and IPv6 configured network

---


Authors:

Robert Diepeveen
robert.diepeveen@os3.nl
11154896

Ruben de Vries
ruben.devries@os3.nl
10260218

Supervisor:
Arris Huijgen, Deloitte


August 2, 2016

**Abstract**

In this paper we address whether it is possible to gain unauthorised access to an 802.1X an IPv6 configured network by piggybacking on a legitimate user's session. To answer this question we have analysed this existing attack in an IPv4 network and highlighted the main components used in such an attack. By means of a literature study we have translated these components from IPv4 to IPv6 configured network, followed by a practical implementation to prove our theoretical findings. During this research we were able to show that piggybacking an authenticated 802.1X session in an IPv6 environment is feasible and not too different compared to an IPv4 environment. Finally, we briefly discuss some techniques to mitigate such attacks.

# Contents

# Acronyms

**ARP** Address Resolution Protocol.

**DHCP** Dynamic Host Configuration Protocol.

**EAP** Extensible Authentication Protocol.

**EAPoL** Extensible Authentication Protocol over LAN.

**IDS** Intrusion Detection System.

**IEEE** Institute of Electrical and Electronics Engineers.

**IPsec** Internet Protocol Security.

**NA** Neighbor Advertisement.

**NAT** Network Address Translation.

**NDP** Neighbor Discovery Protocol.

**NIDS** Network Intrusion Detection System.

**NS** Neighbor Solicitation.

**OSI** Open Systems Interconnection.

**PAE** Port Authentication Entity.

**PNAC** Port-based Network Access Control.

**RA** Router Advertisement.

**RADIUS** Remote Authentication Dial-In User Service.

**RS** Router Solicitation.

**SLAAC** Stateless Address Autoconfiguration.

**VLAN** Virtual Local Area Network.

CHAPTER 1

# Introduction

Rogue devices are an obvious threat to network security. If an unknown, uncontrolled device enters the network it can harm legitimate network users or even the network itself. Therefore, in large networks, devices could be required to authenticate to the network before they are able to send and receive data on it. The most common method of enforcing authentication on a network is by setting it up according to the Institute of Electrical and Electronics Engineers (IEEE) 802.1X specification [1], also known as Port-based Network Access Control (PNAC). The IEEE 802.1X standard defines a method for devices to authenticate themselves to a LAN, which is also known as EAPoL; a prevalent way of authentication with EAP (Extensible Authentication Protocol) over LAN. In EAP a supplicant, which is the connecting device, connects to an authenticator. This authenticator (e.g. a switch) has the connecting 'port' in unauthorised state until the supplicant is successfully authenticated by an external authentication server. There are a multiple methods of authenticating to the authentication server; either with credentials, a certificate or using a pre-shared key.

Since 802.1X only authorises a network port, and not a supplicant, it is possible to create a rogue device that hijacks an authenticated link (or port) established by a legitimate device. Duckwall [2] introduced such an attack, which is described in Section 1.2 (*Related Work*) on page 3 of this paper. These attacks are only tested and described regarding IPv4 networks; no related work on IPv6 enabled networks was found.

Though 802.1X authentication resides on layer 2 of the OSI model [3], the third layer of this model is also essential for this attack. Not only for gaining access to a network but mainly for remaining undetectable. Besides, IPv4 and IPv6 differ in many aspects, leading to the assumption that the proposed attack by Duckwall may not be compatible in an IPv6 environment. For instance, IPv6 uses a larger address space than IPv4, making the Network Address Translation (NAT) no longer needed, whilst this protocol is one of the key components for gaining unauthorised access to an 802.1X configured network. Therefore, the attack mentioned above might be different in IPv6 environments.

## 1.1 Research question

This research aims to answer the following research question: *Is it possible to gain unauthorised access to an 802.1X and IPv6 configured network?*

The main research question has been divided into four sub-questions that will guide this research:

- How is this attack performed on an IPv4 configured network?
- What are the key components for such an attack?
- Are these key components also applicable to an IPv6 configured network?
- If not; is it possible to identify new components that can be used in this attack?

## 1.2 Related work

In 2011, Alva Duckwall introduced a method for bypassing the 802.1X authentication protocol on a wired network using a network bridge in Linux. During this attack, an adversaries' system must reside on the link between a legitimate host and the authenticator to whom this host is authenticating. The *attacker device* is connected using two interfaces. Subsequently, a network bridge is created using these interfaces. By spoofing the network traffic generated by the victim system, the adversaries' system can impersonate the victim and pass through all its packets in such a way that it is not detected by the authenticator or any other Intrusion Detection System (IDS). To achieve this, a clever NAT mechanism is configured on the adversaries' system [2].

As described in the previous stated related work, some NAT configuration is required for a feasible attack. During penetration tests this can be a time consuming effort to configure, motivating a professional *pentester* named Abb to develop a tool to automate this process; Marvin [4]. Unfortunately this tool is not being maintained any more (last update from 2011) and may thus be outdated. However, the concept behind this tool is still relevant for gaining unauthorised access to an 802.1X configured network.

Not every system that is deployed is compatible with the 802.1X protocol. *Legacy systems* are not necessarily capable of performing 802.1X authentication. For instance, some printers are not 802.1X compliant but are still widely deployed in networks configured that roll out 802.1X. Therefore, it is possible to add policies to the security configuration to allow specific legacy devices without them actually identifying over 802.1X. These legacy devices are usually identified by means of a MAC address. Using Nacker, a tool developed by Carmaa [5], one can scan a network for any of those legacy devices and mimic the parameters such a device. By pretending to be a legacy device, an adversaries' system can gain access to a secured network.

Companies performing penetration test on site would like to gain access to an 802.1X configured network and unknowingly monitor any traffic between a victims system and the network. To achieve this, the tapping devices needs to be small and portable such that it is undetectable for employees from the company under investigation. The Pwn Plug is such a device that had its third release end 2014; Pwn Plug R3. This version currently only supports gaining unauthorised access to 802.1X networks in an IPv4 configured environment. Any code residing on such a device is proprietary and therefore not easily accessible for auditing and research [6].

Although the research described in this paper focuses on wired networks, some researchers have published about weaknesses of `802.1X` in wireless environments. Mishra and Arbaugh [7] showed that both a *man-in-the-middle* and session hijack attack could be performed. A man-in-the-middle attack can be established since `802.1X` only provides one-way authentication. It is therefore possible as an adversary to act as an *Access Point* to the user, and as a user to the actual Access Point. The second feasible attack, session hijacking, can occur because of the race conditions between `802.1X` and `802.11` state machines. An adversary could kick a legitimate user from the network by means of a disassociate message and could then take over the session.

## 1.3 Report structure

In this chapter, we have provided an introduction to our research. In the next chapter, *Background*, we will provide a theoretical foundation needed throughout the rest of this paper. In Chapter three, *Methodology*, our test environment and the approach followed during this research is described. Results gathered by using this approach are listed in chapter four, followed by a discussion about these results in chapter five. Finally a conclusion based on this research is drawn and future work is suggested in respectively chapter six and seven.

CHAPTER 2

# Background

In this chapter we provide a theoretical basis about 802.1X, a method to gain unauthorised access to an 802.1X and IPv4 configured network and the relevant differences between IPv4 and IPv6 for this research.

## 2.1 802.1X

In 2003, the IEEE released the 802.1X standard. This standard describes a protocol for *port based network access control* and provides an authentication method for devices connecting to a WLAN or LAN; EAPoL. EAPoL is the encapsulation of the Extensible Authentication Protocol (EAP) over a LAN [8].

A typical 802.1X authentication scheme consists of three involved parties: a supplicant, an authenticator and an authentication server. The connecting client is referred to as the supplicant, whereas the device the supplicant is connecting to is called the authenticator (e.g. a network switch). The authentication server is an entity that is able to verify the identity of the supplicant.

When a supplicant initially connects to the authenticator it can only send EAPoL frames due to the *unauthorised* port state of the connected network port. The supplicant will send out an EAPoL-Start frame to the specified 802.1X MAC address[1], which is the authenticator who will subsequently ask for the identity of the supplicant. This identity will then be forwarded to an authentication server (e.g. a Remote Authentication Dial-In User Service (RADIUS) server), which will negotiate an EAP method and authenticate the supplicant. When the supplicant is successfully authenticated the authenticator changes the port status to *authorised*; the supplicant can access the network. Figure 2.1 on page 6 provides a schematic overview of the process described above.

## 2.2 Gaining unauthorised access to an 802.1X and IPv4 configured network

As already briefly mentioned in Section 1.2 (*Related Work*) of this paper, Alva Duckwall introduced a method of attacking an 802.1X enabled network in 2011 by piggybacking on the authentication procedure. This research elaborates on his work by extending the attack from an IPv4 to an IPv6 environment.

[1] This is an IANA reserved special address; see Section 4.2, Table 4.1

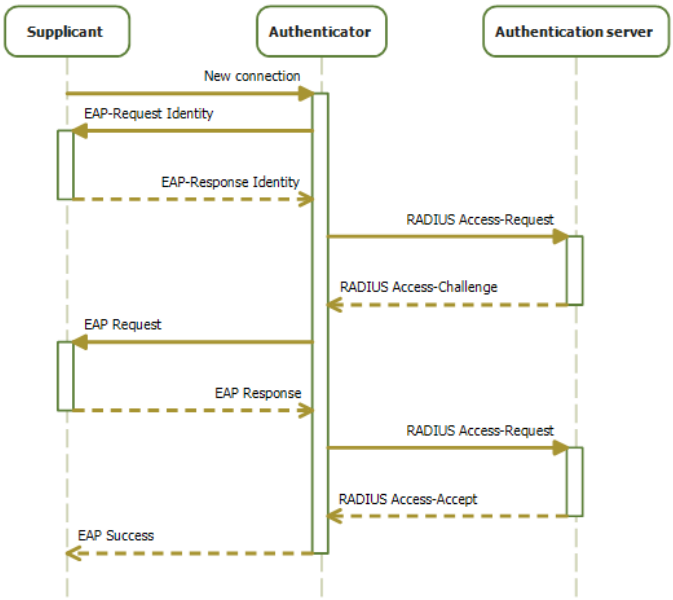


Figure 2.1: The `802.1X` authentication process [8].

Basically, gaining unauthorised access to an `802.1X` configured network is achieved by an adversary by placing an *attack device* between the authenticator and the supplicant. The attack device has two network interfaces; one connected to a legitimate host and the other to an `802.1X` configured authenticator. By default, when the attacker device generates traffic on the network it will use its own `MAC` and possibly `IP` address, leading to an anomaly on the authenticator port since multiple `MAC` addresses are coming from one port. Whether or not the authenticator port will shutdown depends on the configuration of the authenticator.

The victim host's connection should not be interrupted, this is achieved by *bridging* the two network interfaces while dropping all traffic originating from the attacker device. Any network traffic leaving the victim host will flow from the incoming interface via the bridge to the outgoing interface connected to the authenticator. With such a set up, the attacker can easily sniff network traffic generated by the victim host. Please note that by default a bridge does not forward `802.1X` frames, as specified in the IEEE `802.1D` [9] standard for `MAC` bridges, which is a requirement for this attack to work. After all, the victim host needs to be able to authenticate to the authenticator.

To actually use the network with the attacker device, it should generate traffic as if it came from the victim computer. Two parameters were identified from the victim host that should be mimicked such that the authenticator will (continue to) allow traffic: the `MAC` and `IP` address.

Mimicking the IP address from the victim host is trivial, one can use the existing source network address translation (NAT) technique to masquerade the source IP address and thus evade address detection. Such a technique also exists on OSI layer 2 for MAC addresses.

## 2.3 Differences between IPv4 and IPv6

Apart from having a much larger address space, IPv6 introduces a new set of protocols for discovering hosts and routers on the local network. Where IPv4 uses Address Resolution Protocol (ARP) to discover hosts on the physical link and typically Dynamic Host Configuration Protocol (DHCP) for default gateway discovery, IPv6 uses the Neighbor Discovery Protocol (NDP) [10] for both functions.

NDP uses ICMPv6 messages to announce a new (or existing) device on the local network. With so-called *Router Sollicitations* and *Advertisements* the gateway for the local network is determined. Similarly, devices which are on the same local network are discovered by sending and receiving *Neighbor Sollicitations* and *Advertisements*.

Establishing which routes to use for a host is achieved by receiving Router Advertisement (RA) messages. A new host can solicit for an RA by sending out a Router Solicitation (RS) message, but the network also sends them out periodically to the all-node multicast address. These RA messages optionally contain information about the prefix of the network, enabling client hosts to use *Stateless Address Autoconfiguration* (SLAAC) [11] to configure their IPv6 address. At the very least RA messages contain information about the router host such as the link-local address.

When a device would like to send traffic to a host it first sends out a *Neighbor Solicitation* (NS) message to a multicast address. This NS message contains the (link-local) address the sending device would like to know the MAC address of. If this host exists on the local network, it sends out a *Neighbor Advertisement* (NA) in response to the NS message. The sending device then knows that the destination host is on the same local network and traffic does not need to be sent via the router.

CHAPTER 3

# Methodology

This chapter focuses on our approach for conducting our analysis. First, the used test environment is described, followed by the method of addressing the research question stated in Section 1.1.

## 3.1 Environment setup

The test environment consisted of one Cisco Catalyst 3550 Switch, a generic router, a desktop computer used as the victim host, a RADIUS server and an attacker device (Raspberry Pi model 3B). The RADIUS server was connected to the switch over a separate VLAN. The remaining switch ports were all configured as 802.1X ports. In Figure 3.1 an overview of the test network topology is shown after placing the attacker device. The `802.1X` ports are configured with the strictest security settings, that is, only one host (`MAC` address) is allowed per port. If more than one `MAC` address is detected the port will shut down. This is also known as 'single host' configuration.

To summarise, here is a list of the used equipment:

- **Attacker device;** a Raspberry Pi 3B running Linux Kali kernel 4.1.17v7
- **Victim;** multiple systems were used, ranging from Dell desktops running Linux to laptops running Mac OSX
- **RADIUS server;** FreeRADIUS Version 2.2.8, an open source RADIUS server, running on a (virtualised) Ubuntu 16.04 server
- **Switch;** the Cisco Catalyst 3550 series, compatible and configurable with `802.1X` authentication

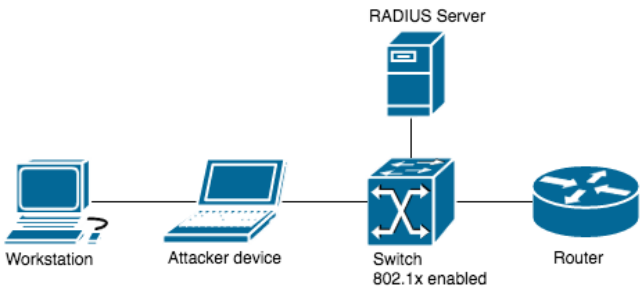


Figure 3.1: Topology as used during experimentation.

## 3.2 Approach

The research question that needs to be answered is: is it possible to gain unauthorised access to an 802.1X and IPv6 configured network? To answer this question three guidelines were defined, which can roughly be divided in to two concepts:

1. How is the attack performed in an IPv4 configured network and what are the key components?
2. Are these key components applicable to an IPv6 configured network?

The structure of each guideline is discussed in the following sections.

### 3.2.1 Identifying key components

Duckwall presented his bypass attack during Defcon 2011 without any published paper work regarding his findings. Therefore, our only material covering the attack are his talk combined with the presented slides. The test environment described in the previous section was set up to recreate the attack in an IPv4 environment given the findings and explanations from Duckwalls presentation. Any code published by Duckwall regarding this attack was no longer available providing both a disadvantage as well as an advantage; identifying key components took more time than expected, still, by writing our own code a better understanding of the attack in an IPv4 environment was gained.

### 3.2.2 Applicability to IPv6

Two phases were addressed during the transition process between both IP versions; a theoretical and practical one. A literature study was performed on basic IPv6 concepts like routing and address layout, followed by a comparison between the key components found in the previous guideline to sketch possible theoretical differences between the researched attack in both IP environments.

Finally, to prove the theoretical research, the IPv4 network was converted to an IPv6 network by disabling IPv4 address allocation on the victim host. There was no need to change switch settings since it automatically forwarded any IPv6 traffic. To extend on the theoretical research of IPv6 performed in the previous stage, IPv6 traffic originating from the victim host was analysed. By combining literature study and real-time IPv6 behaviour analysis together with the code derived from Duckwall's presentation, a fundamental idea was established about any possible differences between an attack in both IP versions.

To prove these possible differences, the code used in the IPv4 configured network was updated such that it could be used in an IPv6 configured network. By then analysing the behaviour of the network the code could be fine tuned such that bypassing 802.1X in an IPv6 configured network was achieved.

CHAPTER 4

# Results

This chapter describes the findings from our research. First an inventory is made of the needed variables for the target network. Then, assuming these variables are collected, the attack is described in stages for both `IPv4` and `IPv6`. Part of this attack is based on the work of Alva Duckwall [2] and Bernard Thaler [12].

## 4.1 Preliminary variables

Multiple preliminary variables were mandatory for performing the researched attack. There are various methods of collecting these, such as social engineering, printer configurations or by analysing network traffic flowing from the victim host over the bridge to the authenticator. The following variables are mandatory for the attack to be successful:

- Local subnet (`IPv4`/`IPv6`)
- Router (`IPv4`/`IPv6`)
- Victim host IP address (`IPv4`/`IPv6`)
- Victim host MAC address
- Router MAC address

The local subnet is needed for the attacker device to determine whether to route traffic for a certain `IP` address to the router or on the local link. The router address is needed to be able to communicate with hosts that are not reachable via the local link.

The combination victim host `IP` and `MAC` address is needed to correctly implement the NAT rules will be shown in the following sections.

## 4.2 Bridge settings

For this attack to work, the attacker device must reside on the physical link between a legitimate network user (victim host) and the authenticator. Therefore, the attacker device must have at least two Ethernet interfaces. Remaining undetected is partly achieved by bridging both interfaces in such a way that packets sent by the victim host will flow from one interface to the other and thus reach the authenticator without the packets being altered. Bridging interfaces is trivial on a Linux system with the `brctl` tool:

| MAC-Address | Description |
|---|---|
| 01-80-C2-00-00-00 | Bridge Group Address |
| 01-80-C2-00-00-01 | (MAC Control) 802.3 |
| 01-80-C2-00-00-02 | (Link Aggregation) 802.3 |
| 01-80-C2-00-00-03 | 802.1X PAE address |
| 01-80-C2-00-00-0*X* | Reserved for future standardization |

Table 4.1: 802.1D defined MAC addresses excluded from forwarding by bridges. $X$ ranges from 4 to F. Source: Table 7-10 of the IEEE 802.1D standard.

```
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
```

In the example provided above first a bridge (br0) is created, to which two interfaces are added; eth0 and eth1. The victim host will not notice any tampering with its physical link to the authenticator. Recall that due to the recommended *802.1D* standard[1] [9], any EAPoL (authentication) frames are dropped by the bridge. Therefore, a minor modification to the bridge needs to be applied (on Linux machines):

```
echo 49144 > /sys/class/net/br0/bridge/group_fwd_mask
```

This modifies the bridge in such a way that it is no longer 802.1D compliant, and will allow Ethernet frames sent to the excluded address range as listed in Table 4.1 to pass through. Do note that a value of $8d = 1000b$ would also work for this specific case, since the 802.1X PAE address ends with 03. However, to be as stealthy as possible it also has to match addresses reserved for future standardisation, which 8 does not. When applying a value it also should not match BR_GROUPFWD_RESTRICTED = $4007h = 0111110100111b$[2]. Any value higher than 49144 will match with BR_GROUPFWD_RESTRICTED and therefore not be accepted as a mask in the bridge.

A bridge operates on layer 2 of the OSI model. Therefore, the version of the IP protocol that is rolled out in the target network does not influence the creation of the bridge.

## 4.3 Preventing detection

With a bridge as mentioned in the previous section, detection happens nearly instantly since all packets are forwarded, even the locally generated ones. In order to make the device undetectable, the bridge must be configured to only

[1] The IEEE standard for bridges

[2] See http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=515853ccecc6987dfb8ed809dd8bf8900286f29e as to why.

forward packets from the *victim host* and to drop all packets from the attacker device.

### 4.3.1 IPv4

By default, a host connected to an IPv4 network will send out and respond to ARP packets. Such a packet contains both MAC and IP addresses and will thus form a collision at the authenticator side of the network; two different MAC and IP addresses are received on one network port, leading to the violation of the single host configuration of the authenticator. In other words; the victim host is disconnected from the network.

To prevent ARP request packets from leaving the attacker device, one could use the following command:

```
arptables -A OUTPUT -o eth0 -j DROP
```

Any ARP packets originating from the attacker device are dropped, hence not received by the authenticator. Note that eth0 is the interface that is connected to the switch.

### 4.3.2 IPv6

For the attacker device to not be noticed on an IPv6 network, it needs to be configured to ignore Router Advertisement messages for itself. When a Router Advertisement message is accepted the bridge can use SLAAC to configure an IPv6 address. This is undesirable since another host would be seen on the authenticator side. Configuring the bridge in such a way is done by issuing the following command:

```
echo 0 > /proc/sys/net/ipv6/conf/br0/accept_ra
```

When a host does not have an IPv6 address, it does not respond to Neighbor Solicitation messages.

In order to minimise the amount of NDP messages the attacker device needs to send out, all Neighbor Advertisement (NA) messages that pass the bridge can be interpreted by the attacker device to be active hosts on the local link.

## 4.4 Address translation

Thus far, a device is created that can undetectably spoof any traffic leaving the victim host. Ultimately, an adversary would like to not only spoof traffic but also gain access to the secured network. To achieve this, the attacker device needs to mimic the MAC and IP addresses such that when packets received by the authenticator seem to originate from a legitimate host and hence not violate the single host configuration. Source NAT is a technique that captures an Ethernet frame and substitutes the source MAC and/or IP address to an address of choice.

A layer 2 (MAC address) source NAT can be instantiated with ebtables in the following way:

```
ebtables -t nat -A POSTROUTING -s $SWMAC \
   -o $SWINT -j snat --to-src $COMPMAC
```

In this rule, any frame leaving interface $SWINT with source $MAC address $SWMAC has its source address translated to $COMPMAC. In Table 4.2 the definition of the variables is shown.

| Variable name | Meaning |
|---|---|
| $SWINT | Name of the interface on the switch side |
| $SWMAC | MAC address of the interface on the switch side |
| $COMPMAC | MAC address of the *victim host* |

Table 4.2: Meaning of variables in the ebtables rule.

### 4.4.1 IPv4

Within Linux distributions the most prevalent way of instantiating NAT rules is by means of the iptables tool. The following rule is an example of a source NAT as used during experimentation:

```
iptables -t nat -A POSTROUTING -s $BRIP \
    -o $BRINT -p tcp -j SNAT --to $COMIP:$RANGE
```

This example rule instantiates a source NAT with IP addresses; any TCP packet with source IP address $BRIP leaving interface $BRINT is changed to $COMIP. An ephemeral port range is added to the $COMIP[3]. The meaning of the variables is shown in table 4.3. There are two similar rules added; one for UDP traffic containing a different ephemeral port range compared to the TCP rule and one for ICMP traffic.

| Variable name | Meaning |
|---|---|
| $BRIP | Internal IP address of the bridge |
| $BRINT | Interface name of the bridge |
| $COMIP | IPv4 address of the *victim host* |
| $RANGE | Ephemeral port range of the *victim host* |

Table 4.3: Meaning of variables in the iptables rule.

[3]See section 5.1.4 for more on ephemeral ports

### 4.4.2 IPv6

For IPv6 there is a similar tool to instantiate source NAT rules; ip6tables. The following rule is an example of a source NAT for IPv6 used during experimentation:

```
ip6tables -t nat -A POSTROUTING -s $BRIP6 \
    -o $BRINT -j SNAT --to-source [$COMIP6]:$RANGE6
```

This rule instantiates an NAT66[4] for packets originating from the bridge to appear as if they came from the *victim host*. The definition of the variables is shown in table 4.4.

| Variable name | Meaning |
|---|---|
| $BRIP6 | Internal IPv6 address of the bridge |
| $BRINT | Interface name of the bridge |
| $COMIP6 | IPv6 address of the *victim host* |
| $RANGE6 | Ephemeral port range of the *victim host* |

Table 4.4: Meaning of variables in the ip6tables rule.

## 4.5 Summary

To summarise this chapter, gaining unauthorised access to an 802.1X configured network is analysed using the test setup described in section 3.1. The key components of the attack are dropping any ARP/NDP traffic originating from the bridge, setting up a source NAT for the MAC and IP addresses and interpreting all ARP/NDP messages that flow through the bridge. This combination of different steps makes for a stealthy attacker device.

[4] NAT 6-to-6

CHAPTER 5

# Discussion

## 5.1 Layer 2 protocol

While 802.1X operates on layer 2 of the OSI model and IPv6 is a layer 3 protocol, there may be differences in the attack. Especially communication to hosts in the same local network and the victim host. This is because the attacker device is imitating the victim host, even spoofing its MAC address to be the same as the victim. Other devices on the network will send data to the attacker device. Therefore device needs to properly filter out data destined for itself and forward data destined to the victim host. Since 802.1X operates on a layer lower than IP, one could argue that comparing these versions in the researched attack is not relevant. Still, some differences between both IP versions were found when the attacker device would like to remain undetected on the network. It needs to handle certain protocol operations differently. For instance, when two hosts on the local link would like to communicate via IP, there needs to be a mapping between IP and MAC address. Resolving IP addresses to MAC addresses is achieved differently between the two IP versions and the attacker device needs to handle the messages accordingly.

What's more is that by default in Linux, bridges automatically use SLAAC to configure an IPv6 address. This generates traffic originating from the bridge and therefore needs to be disabled.

So while 802.1X is actually a layer 2 protocol whereas IP operates on layer 3, there is a difference in remaining undetected, which is what the adversary would like to achieve.

## 5.2 Mitigation techniques

As demonstrated in our analysis, gaining unauthorised access to an 802.1X and IPv6 configured network is essentially based on bridging traffic from the victim host to the authenticator and by mimicking the victims MAC and IP address. Recall that both addresses needed for mimicking can easily be gained when sniffing the network. Therefore, to reduce the feasibility of this attack happening in your network, you could deploy several security protocols which are mentioned below. All of the following protocols have one thing in common, that is they authenticate and, possibly, encrypt every frame or packet that comes from a legitimate host whereas 802.1X only authenticates a *port*.

### 5.2.1 IPsec

Internet Protocol Security (IPsec) [13] is a protocol suite providing authentication and possibly end-to-end encryption for every IP communication session. One could imagine that, when IPsec is used in tunnel mode, encrypting the datagram and therefore the `IP` source address effectively mitigates the researched attack. Initially, IPsec was a mandatory and default feature for `IPv6` but eventually became optional due to Cisco releasing `IPv6` ready network devices *without* IPsec compatibility. Besides becoming an optional feature in `IPv6`, IPsec also is difficult to properly configure, making it less convenient for network engineers to deploy it in their networks.

### 5.2.2 MACsec

Whereas IPsec provides `IP` datagram authentication and encryption, `MACsec` (or IEEE 802.1AE) provides Ethernet frame authentication and possibly encryption. Encryption ensures confidentiality and integrity between two hosts on a point to point link, making it impossible for an attacker device to interfere on the physical link between the authenticator and supplicant [14]. Again, 802.1AE ensures that every frame that leaves the interface is authenticated.

### 5.2.3 SEND

When communicating in a `SEND` enabled network, all NDP messages are cryptographically signed. Therefore, it is not possible for a host on the local link to send out Neighbor Solicitation without knowing the appropriate keys. The attacker device is therefore likely unable to communicate with hosts on the local network that the victim host has not contacted before.

### 5.2.4 Host-based Intrusion Detection System

The victim host is sending traffic to the bridge when under attack. For the victim host, this cannot be the same `MAC` address as the victim host's interface since two hosts with equal `MAC` addresses would appear on the same link. So the attacker device advertises itself to the victim host with another `MAC` address. If there is a host-based intrusion detection system is running on the victim host, this change might be noticed and subsequently someone can be warned. Please note that this is easily counter-mitigated by spoofing the `MAC` address of the bridge to the victim host to be equal to the `MAC` address of the gateway.

What's more is that if the victim host is rather active, some connections from the victim host may drop due to the ephemeral ports[1] being taken by the attacker device. In fact, for a Windows 8 machine the range of ephemeral ports is `49152-65535`. This means that there is a one in $65535 - 49152 = 16383$ chance of terminating an existing connection on the victim host due to a collision in

[1]Ephemeral ports are the source ports for connections chosen by the client, see `https://www.cymru.com/jtk/misc/ephemeralports.html`

source port. This is a relatively low probability when the attacker performs an aggressive (naive) threaded port-scan.

### 5.2.5 Network Intrusion Detection Systems

Network Intrusion Detection System (NIDS) interprets and parses all TCP packets that come from a certain host. A smart NIDS will be capable of fingerprinting a host using TCP/IP parameters. The TCP/IP parameters that are set in every packet are the window size and the Time To Live. Several other parameters can be set in a TCP packet, such as window scaling and maximum segment size. A fingerprint consists of (ranges of) values that a normal host usually sends. When the attacker device does not match the fingerprint for a certain device on the network, there is an anomaly.

Therefore, when constructing an attacker device one should take into account the TCP/IP options originally set by the victim host.

## 5.3 Device portability

Ideally, an adversary (or *pentester* for that matter) would prefer the attacker device to be a small portable device such that it can easily be hid in the attacked environment. Therefore, in our physical test setup, initially a Raspberry Pi was chosen to function as the attacker device. However, during the research with this portable device only unauthorised access was gained in an `IPv4` environment. Bernard Thaler, a German researcher, pointed out that in Linux Kernel versions lower than 4.2 a bug was present regarding *NAT66* on a network bridge [15], which could be resolved by patching the kernel. From kernel version 4.3 the patch was included by default. Still, Raspberry Pi 3B only supports Kali Linux kernel versions as high as 4.1.17. For the sake of research we decided to improve our test setup by replacing the Raspberry Pi with a laptop running Kali Linux with kernel version 4.3, leading to a successful attack in an `IPv6` network. In the future work section of this paper we sketch and motivate briefly how to still use the Raspberry Pi as the portable attacker device.

CHAPTER 6

# Conclusion

This work addresses whether or not it is possible to gain unauthorised access to an `802.1X` and `IPv6` configured network. This research was performed following and answering the outline as described in the Research Question section of this document.

First, gaining unauthorised access to an `802.1X` and `IPv4` configured network was analysed both theoretically and practical on a low level, leading to a proper understanding of the attack. After inserting the attacker device between the victim host and the authenticator a bridge must be set up such that the victim host is still able access to the network it was authenticated to. In essence, after forming this bridge, the attacker device functions as a sniffing device. With this sniffing property any preliminary variables mandatory for the researched attack can be extracted from traffic flowing over the bridge. From an adversaries point of view, one would ideally not only sniff the network, but also gain access it. To achieve this, the attacker device must not advertise itself with a unique `MAC` and `IP` address, since the authenticator will then unauthorise the network port to which the victim PC (and thus attacker device) is connected. Therefore, any traffic leaving the attacker device when attached to the network needs to be dropped and it must mimic the victim host by instantiating a *Source NAT* for `MAC` and `IP` address by using `ebtables` and `iptables`.

Second, the *IPv6* protocol was investigated given the main components of the attack. Instantiating a *Source NAT* with *IPv6* is theoretically comparable to `IPv4`. On the other hand, dropping `ARP` traffic does *not* translate between both protocols; `IPv6` deploys `NDP` as discovery protocol. Therefore, the attacker device needs to be modified to also parse and interpret `NDP` messages.

Finally, during the practical implementation phase, both setting up a *Source NAT* and dropping *NDP* traffic were achieved in a similar way as with *IPv4*, only with different tools. There was however one unpredicted factor; Linux kernels lower than version 4.3 have difficulties regarding `NAT66` over a bridge [15].

CHAPTER 7

# Future work

## 7.1 Device portability

As already mentioned before, gaining unauthorised access to an `802.1X` and `IPv6` configured network was only achieved with a laptop as the attacker device, which is not ideal for adversaries/pentesters since it is complicated to hide. A follow up research could be to focus on small portable devices such as the Raspberry PI. We did start this research with a small device as the attacker device but was ultimately not feasible given the `NAT66` bug. There is a patch available for this bug however, which was not addressed in this research. Note that this is applicable when Kali is the preferred Linux distribution; it could very well be that other distributions compatible with the Raspberry Pi do use the right kernel version.

## 7.2 Remote access

Placing the attacker device is essentially phase one of the attack. Ultimately, an adversary would like to have remote access to this device to gain access to the network from outside the company it is attacking. In the test setup used in this research there was no focus on remote access; the attacker device was accessed directly. A future research question could thus be how to access an *attacker* device remotely via a third network interface. One could imagine this to be a 4G sim card such that the connection to it is wireless. Additional question arise when connecting to it remotely given the *invisible* property of the device.

## 7.3 Mitigation techniques

In the discussion section we suggested some techniques to mitigate the researched attack in an `IPv6` environment. These suggestions are based on literature and theoretical study but were not practically implemented and tested in this work. Therefore, a possible future research could be to practically implement either *MACsec*, *IPsec* or *SEND* and analyse the behavior of the attack in such an environment.

## 7.4 Multiple IPv6 addresses

One new feature in IPv6 is that a host can have multiple IPv6 addresses assigned to one interface. During this research however, only one IPv6 address was assigned per interface which was subsequently spoofed by the attacker device. Theoretically, gaining unauthorised access to an 802.1X enabled network can still be achieved with multiple IPv6 addresses. One could imagine that selecting any of the non link-local addresses from the assigned addresses would make no difference compared to host that is configured with one IPv6 address. Still, this matter was not addressed during the research and therefore leaves an opportunity for future research.

# Bibliography


[1] IEEE Computer Society. *Local and metropolitan area networks — Port-Based Network Access Control*. IEEE 802.1D Std. 2010.

[2] Alva Duckwall. *A bridge too far*. July 2011. URL: https://www.defcon.org/images/defcon-19/dc-19-presentations/Duckwall/DEFCON-19-Duckwall-Bridge-Too-Far.pdf (visited on 05/28/2016).

[3] Cisco. *Chapter: IEEE 802.1X Port-Based Authentication*. URL: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/dot1x.html#wp1133555 (visited on 05/31/2016).

[4] Abb. *Tapping 802.1x Links with Marvin*. Jan. 15, 2011. URL: http://www.gremwell.com/marvin-mitm-tapping-dot1x-links (visited on 05/31/2016).

[5] Carmaa. *Nacker*. Sept. 24, 2014. URL: https://github.com/carmaa/nacker (visited on 05/30/2016).

[6] Pwnie Express. *Pwn Plug R3*. URL: https://www.pwnieexpress.com/product/pwn-plug-r3penetration-testing-device/ (visited on 05/30/2016).

[7] Arunesh Mishra and William A Arbaugh. "An initial security analysis of the IEEE 802.1 X standard". In: (2002).

[8] Wikipedia. *IEEE 802.1X — Wikipedia, The Free Encyclopedia*. [Online; accessed 5-July-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=IEEE_802.1X&oldid=724952337.

[9] IEEE Computer Society. *Local and metropolitan area networks — Media Access Control (MAC) Bridges*. IEEE 802.1D Std. 2004.

[10] W. Simpson T. Narten E. Nordmark and H. Soliman. *Neighbor Discovery for IP version 6 (IPv6)*. RFC 4861. RFC Editor, Sept. 2007, pp. 1–97. URL: https://tools.ietf.org/rfc/rfc4861.txt.

[11] T. Narten S. Thomson and T. Jinmei. *IPv6 Stateless Address Autoconfiguration*. RFC 4862. RFC Editor, Sept. 2007, pp. 1–30. URL: https://tools.ietf.org/rfc/rfc4862.txt.

[12] Bernard Thaler. *Fooling wired Network Access Control*. 2014. URL: https://itsecx.fhstp.ac.at/wp-content/uploads/2014/11/Thaler_2014_Fooling_wired_NAC.pdf (visited on 06/07/2014).

[13] S. Kent and K. Seo. *Security Architecture for the Internet Protocol*. RFC 4301. RFC Editor, Dec. 2005, pp. 1–101. URL: https://tools.ietf.org/rfc/rfc4301.txt.

[14] Cisco and/or its affiliates. *Identity-Based Networking Services: MAC Security*. May 1, 2011. URL: http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/identity-based-networking-services/deploy_guide_c17-663760.pdf (visited on 06/27/2016).

[15] Bernard Thaler. *[PATCH 1/1] bridge: Fix NAT66ed IPv6 packets not being bridged correctly*. May 1, 2011. URL: http://www.spinics.net/lists/linux-ethernet-bridging/msg05573.html (visited on 09/15/2014).