Identification of the Possible Security Issues of the 464XLAT IPv6 Transition Technology

Ameen Al-Azzawi, and Gábor Lencse

Abstract—**This paper focuses on one of the most prominent IPv6 transition technologies named 464XLAT. The aim is to analyze the security threats that this technology might face. After carrying out the threat analysis using STRIDE method that stands for Spoofing, Tampering, Repudiation, Information Disclosure and Elevation of Privilege, and using DFD (Data-Flow Diagram) as a core for the analysis, we summarized the security vulnerabilities and attack points possibilities within this infrastructure. We have also built a testbed for 464XLAT topology using several virtual machines, which were created using Debian image. We used our testbed to perform DoS (Denial of Service) attack against the PLAT (provider-side translator) and monitor PLAT's performance and the number of packets being translated under attack by different number of clients using the hping3 command.**

Index Terms—**464XLAT, DNS, IPv4aaS, IPv6, STRIDE, Translation**

I. INTRODUCTION

After the depletion of the IPv4 address pool, several technologies were invented to provide a practical solution for this matter. The high number of *IPv6 transition technologies* are surveyed and they are classified into different categories regarding the importance of their security analysis in [1]. The methodology for the identification of potential security issues of different IPv6 transition technologies has been defined in [2]. That paper also includes a detailed security analysis of DNS64 (DNS extensions for network address translation from IPv6 clients to IPv4 servers) [3] and a shorter security overview of stateful NAT64 (Network address and protocol translation from IPv6 clients to IPv4 servers) [4]. The combination of DNS64 plus NAT64, however, has its own drawbacks, especially that the connection can only be established from the IPv6 only client and not supporting IPv4 literals [5]. Then came 464XLAT [6] with its double translation mechanism, where it did sort out the issues presented by the DNS64 +NAT64 solution.

However, the application of 464XLAT may involve various security vulnerabilities. Therefore, it is essential to analyze the security threats that might affect this promising technology. According to [6], 464XLAT in general is very quick to deploy and has minimal IPv4 resource requirements and maximum IPv4 address sharing efficiency. Moreover, 464XLAT employs traffic engineering and capacity planning without the indirection or obfuscation of a tunnel [6].

Previously, we have published a conference paper [7], in which we have analyzed the potential vulnerabilities of 464XLAT. In this paper we are taking it one step further by building 464XLAT topology with Linux based virtual machines and actually monitoring the operation of 464XLAT and the performance of its translators under DoS attack. (This paper is an extension of our former conference paper[7].)

The main focus of this paper is to highlight security threats facing the network infrastructure as a result of deploying 464XLAT within the network topology and building the testbed where an actual topology is tested with several attacking clients.

In Section II, we discuss the operation of 464XLAT and its structure, section III is about operation of the STRIDE method, its elements and how it works, section IV is about 464XLAT security revealed by applying STRIDE on it, while in section V, we mention some previous publications regarding 464XLAT / NAT64 security threats and in section VI, we build the testbed and explain its infrastructure and its topology elements. In section VII, we demonstrate an attack scenario using hping3 command and adding several clients gradually. In section VIII,

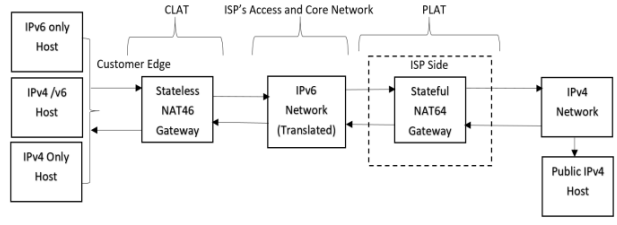

Fig. 1. Overview of 464XLAT Architecture

we analyze the results of our attack. In section IX, we present our plans for future research and the significance of our results. In section X, we summarize and conclude the value of the results the paper came up with where we prove that 464XLAT is effective technology and also susceptible to attack possibilities such as DoS.

A. Azzawi is with the Department of Networked systems & Services, Budapest University of Technology & Economics, Budapest, Hungary.
(e-mail: alazzawi@hit.bme.hu)
G. Lencse is with the Department of Networked systems & Services, Budapest University of Technology & Economics, Budapest, Hungary.
(e-mail: lencse@hit.bme.hu)
Submitted: May 11, 2021.

DOI: 10.36244/ICJ.2021.4.2

II. THE OPERATION OF 464XLAT

The main structure of 464XLAT, as shown in Fig. 1, is divided in two sides; CLAT & PLAT.

A. CLAT (customer-side translator)

CLAT algorithmically translates 1:1 private IPv4 addresses to global IPv6 addresses and vice versa [6]. It acts as IPv6 router, DNS proxy and DHCP server for local client as well. Normally, CLAT must know its own prefix and PLAT side prefix in order to use it as destination for its outgoing packets [6].

As for IPv6 client, its own packets will pass through the CLAT without the need to any translation process and will be forwarded to the PLAT directly.

B. PLAT (provider-side translator)

It translates N:1 global IPv6 addresses with the previously set CLAT prefix to public IPv4 addresses and vice versa [6], it actually implements a stateful NAT64 gateway as described in RFC 6146 [4]. We give an easy introduction to understand to the operation of 464XLAT by Fig. 2. The client in the bottom left hand side corner of the figure (using private IPv4 address 192.168.1.2) wants to connect to the server in the top left hand side corner (using public IPv4 address 198.51.100.1). The prefix at CLAT side is 2001:8db:aaaa::/96, whereas the prefix at PLAT side is 2001:8db:1234::/96. CLAT translates the IPv4 packet into an IPv6 packet, in which the source address will be 2001:db8:aaaa::192.168.1.2, and the destination address will be 2001:db8:1234::198.51.100.1.

At the PLAT side, the 2001:db8:1234::/96 prefix is discovered in the destination address, and an IPv4 packet is built using the embedded 198.51.100.1 IPv4 address as destination address, and the source IPv4 address is chosen from the pool of 192.0.2.1-192.0.2.100 (this time it happened to be 192.0.2.1). Source port is also replaced, when needed, and the connection is registered into the state table of the NAT64 translator to be able to perform the stateful translation in the reverse direction, too. (Please refer to RFC 6146 [4] for further details of the stateful NAT64 translation.)

Besides double translation, there are two other possible scenarios. If both the client and the server have IPv6 addresses, then there is no translation at all, but native IPv6 is used. If the client has an IPv6 address, but the server has only and IPv4 address, then there are two possible modes of operation:

- If DNS64 is configured, then the DNS64 server returns an IPv4-embedded IPv6 address, and only a single translation happens at the PLAT. (This is the DNS64 + NAT64 solution.).
- If no DNS64 is configured, then the client uses IPv4 and double translation happens as described above.

III. THE OPERATION OF STRIDE

STRIDE stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of

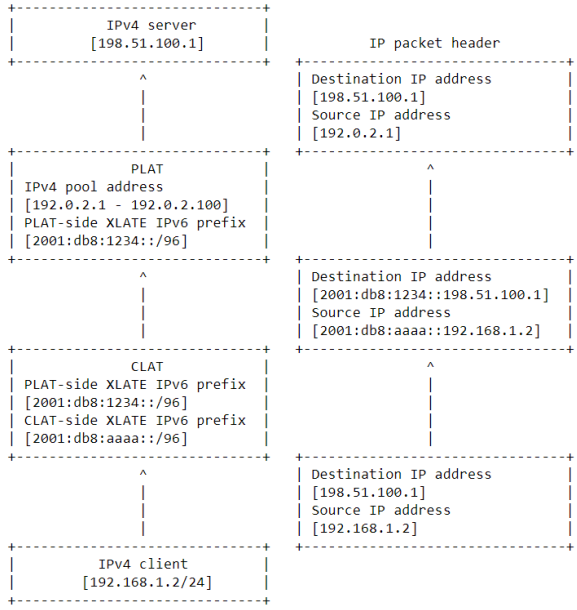

Fig. 2. 464XLAT Packet Processing

Privilege [8]. These are general threats that any network device/node might be susceptible to.

A. *Spoofing:* when an attacker claims to be someone else by changing his real source IP address in order to bypass a filter or an IDS (intrusion detection system) and also to perform DDOS (Dedicated DoS attack) against the potential target [7][8].

B. *Tampering:* the process of changing the content of data flow on its way to the destination, for example, the attacker might alter the packet destination to a malicious server [8]. In wireless communication however, a possibility of MIM (man in the middle) can be used to achieve this attack, such as intercepting HTTP and HTTPS connection between HTTP(S) such as mobile or desktop browser [9].

C. *Repudiation:* it is the claim of a user of not doing an act, while he actually did, like DNS resolution request or in case of ATM money withdrawal where customer might withdraw a specific amount of money then claims that he has not performed any transaction [2]. This threat often appears on the business layer (above network layer in TCP/IP or above application layer such as HTTP/HTML).

D. *Information Disclosure:* an attacker gets sensitive information, which could be used in various ways, e.g. it might help him in hacking, like knowing who's talking to whom by monitoring DNS traffic or TTL value of the packet, which gives the attacker an idea of many hops has the packet gone through then the packet original or source address location will be compromised [2].

E. *Denial of Service:* The attacker can flood a system with useless requests to consume as much processing power as possible in order to prevent the targeted machine from serving legitimate (useful) ones. For example, it can flood a DNS server with huge number of useless queries to prevent legitimate queries from getting a response [2].

TABLE I. VULNERABILITIES OF DIFFERENT DFD ELEMENTS [2]

DFD Element	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Data Flows		✓		✓	✓	
Data Stores		✓		✓	✓	
Processes	✓	✓	✓	✓	✓	✓
Interactors	✓		✓			

F. *Elevation of Privilege:* bypassing the authority matrix of specific organization, like getting root permission on a specific server [9].

The STRIDE method uses the DFD (Data Flow Diagram) of the investigated system in order to examine the critical areas within the system, so it comes up with total security analysis using the four types of elements of the DFD (Data Flows, Data Stores, Processes and Interactors).

Data flow models usually applied on network & architecture systems rather than software products, but they can be applied on both [8]. STRIDE has different approaches regarding threat models:

- Assets-centered threat model: anything the attacker wants to access, control or damage. According to [10], assets-centered threat model is being conducted using 4 approaches: DREAD, Trike, OCTAVE and PASTA. For instance, OCTAVE, which stands for Operationally Threat Asset and Vulnerability Evaluation, is a robust approach but its rather complicated, it takes considerable time to learn and get familiar with its process. Furthermore, its documentation is voluminous [10].
- Attacker-centered threat model: it is based on knowing the attacker, his motivations and skills. It is useful but hard to implement [8].
- Software-centric threat model: focuses on software being built and the deployed systems, it's the best approach for threat modeling [8], because it supposes that software developers are the best people to understand the software they are developing, which makes the software an ideal starting point to trigger the threat modeling process.

In general, the best models are diagrams that help participants understand the software and find threats against it. Each element of the DFD has its own security threats as explained in Table I. It means each element is susceptible to some threats while not susceptible to others [8].

IV. SECURITY ISSUES OF 464XLAT

We presented DFD of 464XLAT in a previous paper [11]. Nevertheless, we made some slight changes on the DFD and after applying the STRIDE method on the DFD diagram of 464XLAT in Fig.3, some security threats are visible at the points (1-11), which represent the threat possibilities within the DFD diagram. In this section, we carefully examine all the elements of the DFD for all possible threats & attacks in details. (Please see the summary of vulnerabilities in Table II.)

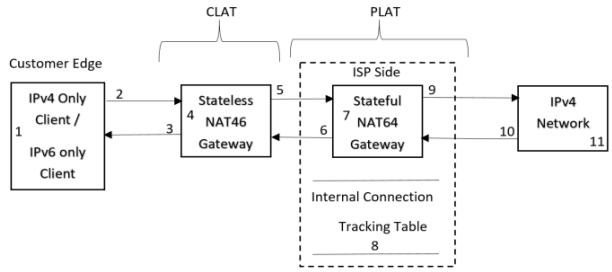

Fig.3 DFD for the threat Analysis of 464XLAT

A. IPV4 / IPV6 Client

1) Spoofing:

- The spoofed IP could be used to send useless packets to the CLAT and this scenario is considered as DOS attack against the CLAT.

2) Repudiation: the client might deny the request he made in the first place.

B. Data flow from IPv4 only client to NAT46

1) Tampering: it can be used as an attack against the domain name or changing the IP address of packet destination, which might be used to direct the packet towards fraudulent server, this kind of attack is also called FoS (Failure of Service) because it prevents the real client from receiving an answer to its real query [2].

2) Information Disclosure: an attacker might be interested in knowing the browsing habits of the requester, and the packet itself might contain some sensitive information sent by the client himself [2].

3) Denial of Service: flooding the gateway with unwanted requests to prevent the real query from getting an answer.

C. Data flow from NAT46 gateway to the client

1) Tampering: an attack against the client, for example sending misleading information at application level or breaking the connecting sending a RST at TCP level.

2) Information Disclosure: an attacker getting access to sensitive data.

3) Denial of Service: sending high number of forged replies to the client to prevent if form processing the genuine ones.

D. NAT46 Gateway (CLAT)

1) Spoofing: in this case, spoofing means unauthorized user controls the gateway and translate the private IPv4 to the wrong IPv6 and send the packet to different destination.

2) Tampering: an attacker tampered with the data within the gateway itself by which might result in e.g. returning the wrong IPv6 address [2].

3) Repudiation: after spoofing the CLAT, an attacker might deny sending a packet that was actually sent by the CLAT himself while hiding his own identity. Logging is the key here, if the database administrator is not fully trusted, then a system in another privilege domain has to be installed.

4) Information Disclosure: an attacker might make use of the browsing data and queries made by the requester in order to hack the main requester later on.

5) Denial of Service: it could be by an attacker spoofing an IP of legitimate user flooding the CLAT with huge number of requests, see section IV.B.3.

6) Elevation of Privilege: it happens when an attacker gain access to a service he shouldn't be getting in the first place. However, it mainly happens due to inside job [2] and the attacker might gain the right of admin or root to whatever he likes later on.

E. Data flow from NAT46 to NAT64

1) Tampering: the packet destination IP might be altered while it's on its way to NAT64 gateway.

2) Information Disclosure: see section IV.B.2.

3) Denial of Service: after spoofing the NAT46, attacker might send numerous useless packets to the NAT64 gateway.

F. Data flow from NAT64 to NAT46 gateway

1) Tampering: see section IV.C.1.

2) Information Disclosure: it is possible that an attacker might access the packet details on its way back to NAT46 gateway and extract sensitive information out of it.

3) Denial of Service: flooding the NAT46 gateway with unwanted packets to prevent it from translating the genuine traffic.

G. NAT64 Gateway (PLAT)

1) Spoofing: an attacker might take control (spoof) the gateway and do many malicious activities with it, see section IV.D.1.

2) Tampering: an attacker might change the content of packet details withing the gateway, see section IV.D.2.

3) Repudiation: see section IV.D.3.

4) Information Disclosure: see section IV.D.4.

5) Denial of Service: DoS attack might come in a way that affect the NAT64 Gateway (PLAT), such as Exhaustion of source port and public IPv4 address pool, which is an issue since the gateway uses 63K[1] number of source ports per public IPv4 address. An enhanced algorithm presented by [12] helps in tackling this issue in details.

6) Elevation of Privilege: one of the elevation problems is called buffer overflow attack [13], which could happen if a device like NAT64 getting inputs from both sides and that might affect its memory storage units.

H. Internal connection tracking table

1) Potential attackers have no direct access to it, they can influence its content in indirect ways only.

1) Denial of Service: The attacker may initiate fake connections (either using his real IPv6 address or fake ones) and thus achieve the insertion of fake entries into the connection tracking table. The high number of fake entries may slow down the operation of the NAT64 gateway or even prevent legitimate users from establishing further connections, when the table is full. If PLAT applies a connection limit per source IPv6 address, then the attacker may exhaust the available number of connections for legitimate users by spoofing their IPv6 addresses and initiating fake connections.

I. Data flow from PLAT to IPv4 Server

1) Tampering: attacker might change the source IP address of the packet so the IPv4 server will not know, who sent the packet in the first place.

2) Information Disclosure: see Section IV.B.2

3) Denial of Service: an attacker might spoof the IP and flood the IPv4 server with plenty of undesired requests.

J. IPV4 server / IPv4 network

1) Spoofing: Source IP address might be spoofed, see section IV.A.1.

2) Repudiation: denying of sending a request is viable in this case, *see section IV.A.2.*

K. Data flow from IPV4 server / IPv4 network to PLAT

1) Tampering: the attacker might send TCP-RST packets to erase the mapped entries within the NAT64 gateway.

2) Information Disclosure: it is possible that an attacker might access the packet details on its way back to NAT64 gateway and extract sensitive information out of it, like TTL value or browsing habits [2].

3) Denial of Service: flooding the *NAT64* gateway with unwanted requests to prevent it from translating the real traffic.

L. Summary of the results

To summarize the attacks or the vulnerabilities within 464XLAT structure, we concluded the following threats:

A. Spoofing of NAT46 or NAT64 gateways results in altering packets destination or returning the wrong IP address to the requester.

B. DoS: denying access of a legitimate user to his authorized traffic and obstructing the function of NAT46 & NAT64 gateways.

C. FoS: preventing the real client from receiving an answer to its real query.

D. Leaking of confidential information like IP address, TTL value and browsing habits.

E. Tampering with NAT64 tracking table: loosing of mapped entries.

F. Privileges level altering: getting root privilege will increase the inside job attack very often.

G. Buffer overflow attack in case of NAT64, which affects the storage (connection tracking table) entries and might erase them accidently.

[1] Similarly to NAT devices, NAT64 gateways usually use source port numbers from the range of 1024 – 65536.

TABLE II. SUMMARY OF 464XLAT THREATS

DFD Element	*Threat*	*Possible attacks*
1	Spoofing & Repudiation	DoS attack against the CLAT
2, 3	Tampering, Information Disclosure and Denial of Service	FoS, collecting unauthorized information, DoS
4	All STRIDE Elements	FoS, DoS and unauthorized access,
5, 6	Tampering, Information Disclosure and Denial of Service	FoS, collecting unauthorized information, DoS
7	All STRIDE Elements	FoS, DoS and unauthorized access,
8	Only indirect attacks	Tampering with Connection Tracking Table; DOS attack (exhaustion of connection tracking table, slowing down look up speed)
9, 10	Tampering, Information Disclosure and Denial of Service	FoS, collecting unauthorized information, DoS
11	Spoofing & Repudiation	DoS attack against the PLAT

V. RELATED WORK

Very few papers have been published regarding our topic. However, [14] has focused on the IPv6 security issues as far as cellular networks concern and it came up with different categories of possible attacks. They demonstrated three different DoS attacks on NAT64 block targeting features that only exist in IPv6 cellular networks:

A. *NAT overflow attack:* According to [14], most of service providers tend to drop the source address of a spoofed packet and replace it with a public IPv4 address. Therefore, a host can send & receive packets using single private IPv4 address assigned by NAT.

As a result, the maximum of external mapping for single targeted service is 65,535. Meanwhile, in IPv6 cellular networks, a device can utilize 2^{64} IPv6 addresses. So, if a device creates mapping on NAT64 using all the 2^{64} IPv6 addresses, the result will be $65{,}535 * 2^{64}$ mappings, which can lead to overload for NAT64 [14] . It also showed that NAT64 gateway will stop the mapping process for any incoming request after 1500 entries (depending on the preset value) within its tracking table (if the requester is sending from the same IP address targeting the same service) and sends back TCP-RST packet back to the requester as response for the TCP-SYN packet. However, this policy of NAT64 can be exploited as DoS attack [14].

B. *NAT wiping attack*: The targeted victim in this case is the mapping entry itself. NAT64 uses the N:1 mapping criterion. If an adversary targets the external IPv4 of NAT64 gateway, N hosts are sharing the same external IPv4 address will be liable to DoS attack. The adversary will send malicious TCP-RST packets to wipe out the target mappings within the NAT64. As a result, the mapped users to the very same external IPv4 address will be denied access to their service.

To do so, the attacker needs to know the TCP 5-tuple of the targeted service (Protocol, Destination IP address, port number, External IP address of NAT64 and External port number of NAT64).

C. NAT Bricking attack: it's type of DoS attack which also exploits the N:1 mapping algorithm adopted by NAT64. Basically, the adversary can send huge number of requests using the external IPv4 address(es) of the NAT64 gateway [14]. However, big vendors (google, YouTube, etc.) have IP blocking approach if it exceeds specific number of requests per minute. Nevertheless, [14] has done an experiment to target Google scholar [2] website, which is an IPv4 based site. So, the IPv6 cellular host sends 150 requests per minute to trigger CAPTCHA request. Every time CAPTCHA request emerges, adversary source IP address is being changed by turning the airplane mode on and off, this process was repeated 1000 times. Finally, the NAT bricking attack was able to trigger CAPTCHA request for a total of 631 external IPv4 from Google Scholar, including one of the victim's external IP address [14].

Moreover, [15][15] has explained that the majority of the transition technologies use some form of NAT, NAT44, NAT64, NAT46, etc. and how it is a myth that NAT is putting the user inside this secured box of protective shield from the outside attackers, the sequence of communications below explains how vulnerable the NAT client could be:

1- Attacker attracts the victim towards specific website.
2- Victim clicks on the malicious URL and enters the page.
3- The page has a hidden form connecting to http:// attacker.com:6667 (IRC port).
4- The victim submits the form without his consent.
5- An HTTP connection is created to the (fake) IRC server.
6- The form as well has hidden value which sends:" OPEN DCC CHAT PORT".
7- Router sees an "IRC connection" then open a port back through the NAT.
8- The attacker now has an open path to the network.

The very same process could have been applied using FTP NAT helper if not IRC.

According to [15], todays preferred transition technologies are 6rd, DS-Lite and 464XLAT, while risk Mitigation Strategies can be summarized as follow:

1- Minimizing the need for SP-NAT (Service-Provider NAT).

[2] Google, Google Scholar. https://scholar.google.com.

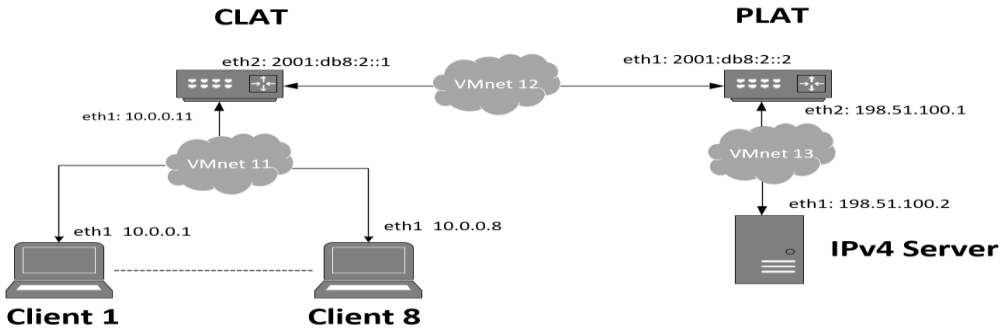

Fig. 4 464XLAT Testbed

2- The more IPv6 established sessions, the less you rely on SP-NAT and all the security issues associated with that.
3- Search for a transition plan that uses native IPv6 such as 464XLAT & DS-Lite.

As for the testbed, several attempts were conducted by researchers to build an efficient testbed in order to test the transition technologies, their weak spots and vulnerabilities. A successful testbed was built by Marius Georgescu [16], in which he measured the latency, throughput and packet loss by adopting 464XLAT transition technology and some other methods as well.

VI. 464XLAT TESTBED

The testbed was built through remote access to a Windows-based virtual machine with the following specifications:

- Intel(R) Xeon(R) Silver 4215 CPU @ 2.50GHz, (16 VCPUs)
- 16.0 GB RAM
- 64-bit Windows 10 operating system.

Further virtualization was created by installing VMware workstation 12 Player. Several virtual machines were created using Debian image, which was created by using the debian-vm tool of Daniel Bakai [3]. Every machine had Debian 8.9 operating system.

A. Test Topology

The topology of the 464XLAT testbed is shown in Fig. 4, it can be divided into two sides:

- On the left side, there are four clients (10.0.0.1 -- 10.0.0.8) and the CLAT.
- On the right side, there are the PLAT and the IPv4 server.

B. Testbed Implementation

In our testbed, each virtual machine has the following specifications:

- Clients 1-8: 1GB RAM, 1 CPU core, and 20 GB hard disk.
  CLAT: 1GB RAM, 3 CPU cores, and 20 GB hard disk.
- PLAT: 1GB RAM, 3 CPU cores, and 20 GB hard disk.
- IPv4 Server: 1GB of RAM, 1 CPU core, and 20 GB hard disk.

Furthermore, the topology was build using three separated virtual networks: VMnet11, VMnet12, and VMnet13.

- VMnet11: the network between the eight clients and CLAT eth1. The network is IPv4 only.
- VMnet12: the network between CLAT eth2 and PLAT eth1. The network is IPv6 only.
- VMnet13: the network between PLAT eth2 and IPv4 server eth1. The network is IPv4 only.

Table III shows the Linux and VMware settings used for each virtual machine.

C. Test configuration

The main configuration blocks are within CLAT (stateless translator) and PLAT (stateful translator).

In both cases, the configurations included three steps:

- configuring TAYGA to run
- configuring the operating parameters of TAYGA
- further settings

The details of their settings are presented below.

1) Configuring CLAT

It was set in the **`/etc/default/tayga.conf`** file that TAYGA should be started at operating system boot time:

```
RUN="yes"
CONFIGURE_NAT44="no"
```

Here, we did not intend to use TAYGA as stateful NAT64 because CLAT is a stateless NAT46 translator.

The operating parameters of TAYGA were set in the **`/etc/tayga.conf`** file as follows:

```
tun-device nat64
ipv4-addr 10.0.0.9
ipv6-addr 2001:db8:2::9
prefix 2001:db8:a::/96
map 10.0.0.1 2001:db8:c::10.0.0.1
map 10.0.0.2 2001:db8:c::10.0.0.2
map 10.0.0.3 2001:db8:c::10.0.0.3
map 10.0.0.4 2001:db8:c::10.0.0.4
map 10.0.0.5 2001:db8:c::10.0.0.5
map 10.0.0.6 2001:db8:c::10.0.0.6
map 10.0.0.7 2001:db8:c::10.0.0.7
map 10.0.0.8 2001:db8:c::10.0.0.8
```

[3] D. Bakai, "Debian VM", [Online]. Available: https://git.sch.bme.hu/bakaid/debian-vm

TABLE III. LINUX AND VMWARE NETWORK SETTING FOR VIRTUAL MACHINES

VM name	Clients 1-8	CLAT	PLAT	IPv4 Server
eth0 Linux setting	DHCP	DHCP	DHCP	DHCP
eth1 Linux setting	Static IPv4: 10.0.0.1-8	Static IPv4: 10.0.0.11	Static IPv6: 2001:db8:2::2/64	Static IPv4: 198.51.100.2
eth2 Linux setting	N/A	Static IPv6: 2001:db8:2::1/64	Static IPv4: 198.51.100.1	N/A
eth0 VMware setting	NAT	NAT	NAT	NAT
eth1 VMware setting	VMnet11	VMnet11	VMnet12	VMnet13
eth2 VMware setting	N/A	VMnet12	VMnet13	N/A

As for further settings, the following bash shell script was responsible for setting up routes and enabling Linux kernel routing:

```
#!/bin/bash
ip route add 198.51.100.0/24 dev nat64
ip route add 2001:db8:c::/96 dev nat64
ip route add 2001:db8:a::/96 via 2001:db8:2::2
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
ip route del 2001:db8:a::/96 dev nat64
```

We note that the last command was to delete a routing rule that was automatically set by TAYGA.

2) Configuring PLAT

It was set in the **/etc/default/tayga.conf** file that TAYGA should be started at operating system boot time:

```
RUN="yes"
CONFIGURE_NAT44="no"
```

Here, we used TAYGA as a stateful NAT64 translator, but we set the **iptables** rule by hand (see below).

The operating parameters of TAYGA were set in the **/etc/tayga.conf** file as follows:

```
tun-device nat64
ipv4-addr 198.51.100.9
ipv6-addr 2001:db8:2::9
prefix 2001:db8:a::/96
map 10.0.0.1 2001:db8:c::10.0.0.1
map 10.0.0.2 2001:db8:c::10.0.0.2
map 10.0.0.3 2001:db8:c::10.0.0.3
map 10.0.0.4 2001:db8:c::10.0.0.4
map 10.0.0.5 2001:db8:c::10.0.0.5
map 10.0.0.6 2001:db8:c::10.0.0.6
map 10.0.0.7 2001:db8:c::10.0.0.7
map 10.0.0.8 2001:db8:c::10.0.0.8
```

As for further settings, the following **bash** shell script was responsible for setting up routes and enabling Linux kernel routing:

```
#!/bin/bash
ip route add 10.0.0.0/24 dev nat64
ip route add 2001:db8:a::/96 via 2001:db8:2::1
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Furthermore, the below **iptables** command was applied:

```
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

The aim of this command is to perform a stateful NAT44 translation using the well-known Netfilter framework. It was necessary, because TAYGA is only a stateless NAT64 translator by itself, and thus it requires an additional stateless NAT44 translator to implement stateful NAT64.

VII. SAMPLE DOS ATTACK

The aim of the DoS attack is to exhaust the resources of the PLAT device. It is carried out by sending high frequency ping requests from the eight clients to the IPv4 server using the **hping3** command.

To carry out the attack, SSH authentication were established between client one (10.0.0.1) and the rest of the network elements in order to be able to carry out the experiments by a script. The script was responsible for starting traffic capture by using **tshark** and for starting the attacking program on all clients at (mostly) the same time by a script.

The attack was through a **hping3** command with specific arguments:

```
hping3 -S -p80 -s5000 -k 198.51.100.2 -iu1500
```

-S: TCP Syn attack.
-p: destination port number.
-s: source port number.
-k: to maintain the same port number and avoid its increment.
-iu: to control the number of sent packets per second.

We note that the last parameter does not directly set the packet rate, but it specifies a kind of targeted delay in microseconds. Different packet rates can be applied by manipulating the "-iu" argument of the **hping3** command (e.g. 100 packets/s, 1000 packets/s, etc.). The experiments were carried out using various packet rates. For the demonstration of the DoS attack, we selected a rate (about 460 packets/s) at which almost all packets were correctly translated by the PLAT, when only a single client was used, but a significant amount of the frames were not correctly translated, when 8 attacking clients were used. Please see a fragment of the **tshark** output in Fig 5. The incorrectly translated frame is highlighted by a red oval.

The attack process was as below:

a. Start the measurements, then wait for 10 seconds to monitor the performance before the attack.
b. Start attack with client 1, then wait for 100 seconds to monitor the effect of one client.

```
158   0.081455 198.51.100.1 -> 198.51.100.2 TCP 54 [TCP Port numbers reused] 5000 ^ ^ 80 [SYN] Seq=0 Win=512 Len=0
159   0.081510    10.0.0.2 -> 198.51.100.2 TCP 54 [TCP Port numbers reused] 5000 ^ ^ 80 [SYN] Seq=0 Win=512 Len=0
160   0.081953 198.51.100.1 -> 198.51.100.2 TCP 54 [TCP Port numbers reused] 5000 ^ ^ 80 [SYN] Seq=0 Win=512 Len=0
161   0.082078 198.51.100.2 -> 198.51.100.1 TCP 60 [TCP ACKed unseen segment] 80 ^ ^ 5000 [RST, ACK] Seq=1 Ack=1329834715 Win=0 Len=0
162   0.082130 198.51.100.2 -> 198.51.100.1 TCP 60 [TCP ACKed unseen segment] 80 ^ ^ 5000 [RST, ACK] Seq=1 Ack=579732553 Win=0 Len=0
163   0.082666 198.51.100.2 -> 198.51.100.1 TCP 60 80 ^ ^ 5000 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

Fig. 5. PLAT eth2 **`tshark`** capture

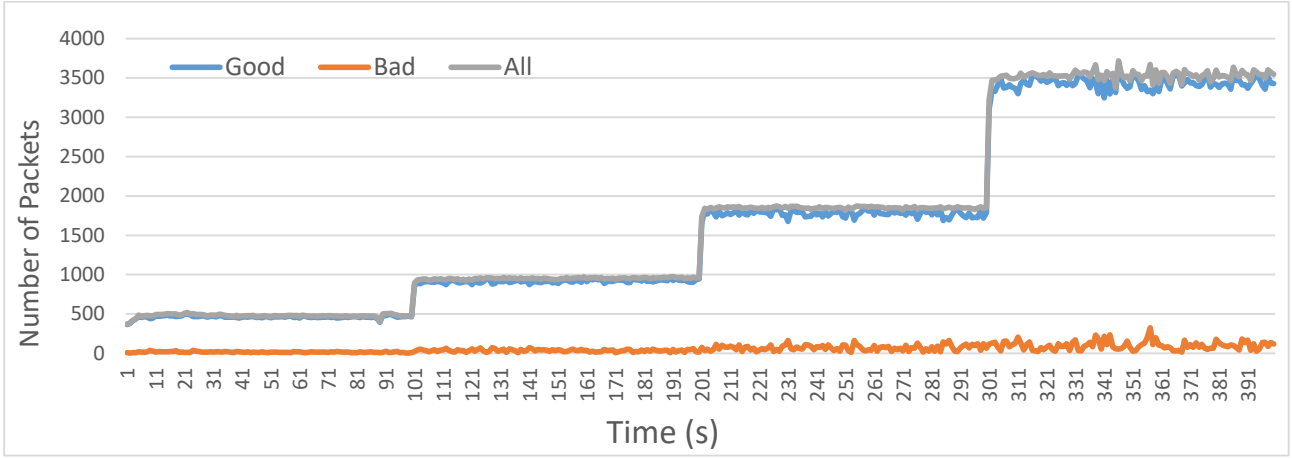

Fig. 6 Measurements with 460 packets/s per client load: the number of good/bad/all packets as a function of time (number of attacking clients: 1, 2, 4, and 8)

c. Start attack with client 2, then waiting for 100 seconds.
d. Start attack with client 3 & client 4, then wait for 100 seconds.
e. Start attack with clients 5, 6, 7, 8, then wait for 100 seconds.
f. End the measurements, then stop the eight attacks.

VIII. Results and Analysis

Fig. 6 shows the number of correctly translated, not translated and all frames per second as a function of time in case of 460 packets/s rate. It is fairly obvious that the number of sent packets are increasing by doubling the number of clients (1, 2, 4, 8). Every spike in the graph represents new added client(s).

- From 0-100 second (only client1).
- From 100-200 second (client 1& 2).
- From 200-300 second (client 1,2,3 and 4).
- From 300-400 second (client 1,2,3,4,5,6,7 and 8).

The number of packets arriving at the PLAT is slightly fewer than the ones sent from the CLAT. The average value of sent packets when 8 clients were applied was around 3500 packets /s. Furthermore, the orange line represents the number of untranslated packets by **`iptables`**.

We divided the packets into three types (good, bad, all). "Good" label represents the properly translated packets, "Bad" label represents packets that failed in the masquerading process and kept their original source IP address. As for "All" label, it is fairly obvious that it represents the total number of good and bad packets together.

The MASQUERADING feature is supposed to change the source IP address of packets leaving the PLAT and heading towards IPv4 server. That means every packet heading towards IPv4 server should have the source IP address of PLAT eth2 interface (198.51.100.1). However, some packets are passing through the filter without getting translated by keeping their original IP address (10.0.0.1-8) instead of PLAT eth2 interface IP address (198.51.100.1) as shown in Fig. 5. We don't know the exact root cause for this behavior. What we know is the frequency of these untranslated packets increases by increasing the number of applied clients as illustrated by Fig. 6.

IX. Plans for Future Research and the Significance of Our Results

A. Plans for Future Research

As for our future research focus, we plan to implement other types of DoS attacks against the PLAT in order to exhaust:

- its TCP source port number pool,
- its internal connection tracking table.

We are also planning to find mitigation for this and for other types of attacks, too.

B. Significance of our Results

464XLAT is very important for network operators, especially ISPs (Internet Service Providers). It is among the five most important IPv4aaS technologies that has to be supported by all customer edge routers [17]. It has several advantages like its port number efficiency and its wide spread support in cellular networks [18]. Therefore, its security properties may be extremely important decision factors for network operators.

Other areas of applications include any kinds of IoT [19] and/or LoRaWAN systems [20], where IPv6 may be needed in the access network due to the high number of devices, but some legacy devices still need IPv4 support. Further important application segment is the intelligent transportation systems in smart cities [21] for the same reason.

X. CONCLUSION

Threat analysis of the 464XLAT IPv6 transition technology was performed by applying the STRIDE method in order to point out the potential vulnerabilities of the technology. This method of using the data flow diagram of the 464XLAT system to analyze its potential security vulnerabilities has proven its efficiency.

The double translation mechanism of 464XLAT proved its effectiveness in terms of IPv4 literals communications over IPv6 infrastructure. However, it has some security issues and vulnerabilities such as DoS attack possibility. Some faulty translated packets were monitored and their percentages increased by adding more load to the topology and therefore affects the PLAT performance. Some readings have visible fluctuation and this was mainly caused by the instability of hping3 command and its packet rate controlling ratio. In general, the experiment proved that 464XLAT is an effective transition technology to establish a stable connection over different IP versions infrastructure.

REFERENCES

[1] G. Lencse and Y. Kadobayashi, "Comprehensive Survey of IPv6 Transition Technologies: A Subjective Classification for Security Analysis", *IEICE Transactions on Communications*, vol. E102-B, no.10, pp. 2021-2035. DOI: 10.1587/transcom.2018EBR0002

[2] G. Lencse and Y. Kadobayashi, "Methodology for the identification of potential security issues of different IPv6 transition technologies: Threat analysis of DNS64 and stateful NAT64", Computers & Security, vol. 77, no.1, pp. 397-411, 2018, DOI: 10.1016/j.cose.2018.04.012.

[3] M. Bagnulo, A Sullivan, P. Matthews and I. Beijnum, "DNS64: DNS extensions for network address translation from IPv6 clients to IPv4 servers", IETF RFC 6147, 2011, DOI: 10.17487/RFC6147.

[4] M. Bagnulo, P. Matthews and I. Beijnum, "Stateful NAT64: Network address and protocol translation from IPv6 clients to IPv4 servers", IETF RFC 6146, 2011, DOI: 10.17487/RFC6146.

[5] S. Répás, T. Hajas, G. Lencse, "Application compatibility of the NAT64 IPv6 transition technology," in Proc. 37th International Conference on Telecommunications and Signal Processing, Berlin, 2014, pp. 49–55, DOI: 10.1109/TSP.2015.7296383.

[6] M. Mawatari, M. Kawashima, C. Byrne, "464XLAT: Combination of stateful and stateless translation, IETF RFC 6877 (2013).

[7] A. Al-Azzawi and G. Lencse, "Towards the Identification of the Possible Security Issues of the 464XLAT IPv6 Transition Technology," 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), 2020, pp. 439-444, DOI: 10.1109/TSP49548.2020.9163487.

[8] A. Shostack, "Threat modeling: Designing for security", 1st Edition, Wiley, Indiana, 2014.

[9] G. Fisher; L. Valenta, Monsters in the Middleboxes, accessed 12 September 2021, https://blog.cloudflare.com/monsters-in-the-middleboxes/.

[10] L. O. Nweke and S. D. Wolthusen, "A Review of asset-centric threat modelling approaches" International Journal of Advanced Computer Science and Applications (IJACSA), 11 (2), 2020. DOI: 10.14569/IJACSA.2020.0110201

[11] A. Al-Azzawi, "Plans for the security analysis of IPv4aaS technologies", in Proc.14th International Symposium on Applied Informatics and Related Areas, University of Obuda, Székesfehérvár, Hungary, 2019, pp. 101–105.

[12] M. S. Ferdous, F. Chowdhury, J. C. Acharjee, "An extended algorithm to enhance the performance of the current NAPT", in Proc. International Conference on Information and Communication Technology, Dhaka, Bangladesh, 2007, pp. 315–318, DOI: 10.1109/ICICT.2007.375401.

[13] A. D. Keromytis, "Buffer overflow attacks", in H. C. A. van Tilborg, S. Jajodia (Eds.), Encyclopedia of Cryptography and Security, Springer, Boston, 2011, pp. 174–177.

[14] Hong, H., Choi, H., Kim, D., Kim, H., Hong, B., Noh, J., & Kim, Y. (2017), "When cellular networks met IPv6: Security problems of middleboxes in IPv6 cellular networks", IEEE European Symposium on Security and Privacy (EuroS&P), 2017, DOI: 10.1109/eurosp.2017.34.

[15] Itu.int. (2016). "IPv6 Security Mechanisms", [online] Available: https://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/Documents/s11-ipv6-securingtransitionmechanisms.pdf [Accessed 5 Feb. 2020].

[16] Georgescu, M., Hazeyama, H., Kadobayashi, Y. and Yamaguchi, S., 2014, May. Empirical analysis of IPv6 transition technologies using the IPv6 Network Evaluation Testbed. In International Conference on Testbeds and Research Infrastructures (pp. 216-228). Springer, Cham.

[17] J. Palet Martinez, H. M.-H. Liu, M. Kawashima, "Requirements for IPv6 Customer Edge Routers", IETF RFC 8585, 2019, DOI: 10.17487/RFC8585

[18] G. Lencse, J. Palet Martinez,, L. Howard, R. Patterson, I. Farrer, "Pros and Cons of IPv6 Transition Technologies for IPv4aaS", IETF v6ops Internet Draft, work in progress, 2021, [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-v6ops-transition-comparison

[19] A. Pekár, J. Mocnej, W. K. G. Seah, I. Zolotova, "Application Domain-Based Overview of IoT Network Traffic Characteristics", ACM Computing Surveys, vol 53. no. 4. art. no. 87, DOI: 10.1145/3399669

[20] I. Drotár, B. Lukács, M. Kuczmann, "LoRaWAN Network Performance Test", Acta Technica Jaurinensis, vol. 13, no 4, pp. 268–280. DOI: 10.14513/actatechjaur.v13.n4.547

[21] A. M. Nagy, V . Simon, "Traffic congestion propagation identification method in smart cities" Infocommunications Journal, vol. 13, no 1, March 2021, pp. 45-57. DOI: 10.36244/ICJ.2021.1.6

Ameen Al-Azzawi received his MSc in Communication Engineering from Northumbria University, Newcastle, England in 2014.
He is now a PhD student at the Budapest university of Technology and Economics, Budapest, Hungary. He has been working full time on his research at MediaNets Laboratory in the Department of Networked Systems and Services since September 2019. His research focus is on IPv6 transition technologies and their security analysis.

Gábor Lencse received his MSc and PhD in computer science from the Budapest University of Technology and Economics, Budapest, Hungary in 1994 and 2001, respectively.
He has been working full time for the Department of Telecommunications, Széchenyi István University, Győr, Hungary since 1997. Now, he is a Networked Systems and Services, Budapest University of Technology and Economics, Budapest, Hungary as a Senior Research Fellow since 2005. The main area of his research is the performance and security analysis of IPv6 transition technologies.