Methodology for the identification of potential security issues of different IPv6 transition technologies: Threat analysis of DNS64 and stateful NAT64

Gábor Lencse[a,*], Youki Kadobayashi[b]

[a] *Department of Networked Systems and Services, Budapest University of Technology and Economics, Magyar tudósok körútja 2, Budapest, H-1117, Hungary*

[b] *Laboratory for Cyber Resilience, Nara Institute of Science and Technology, Takayama-cho, 8916-5, Nara, 630-0192 Japan*

Abstract

We are faced with the transition from IPv4 to IPv6, which will last for several years or possibly decades. There are different IPv6 transition technologies, which enable the communication between hosts using the two incompatible versions of the Internet Protocol in various scenarios, but they also involve additional security issues. In this paper, we develop a methodology for the identification of potential security issues of different IPv6 transition technologies and their implementations based on the STRIDE approach and its application to IPv6 transition technologies. Our methodology includes the application of the STRIDE approach at two levels: the level of the individual IPv6 transition technologies and the level of their selected implementations. We demonstrate the operation and viability of our methodology by the detailed threat analysis of the DNS64 technology, and we prove the necessity of the implementation level analysis with several examples. We also include the most interesting highlights of the security analysis of the stateful NAT64 IPv6 transition technology. We also make a survey of the published vulnerabilities of DNS64 and NAT64 in different research papers and show the effectiveness of our systematic method by uncovering the threats of DNS64 and NAT64. Finally, we point out the need for the in-depth security analysis of the DNS64 technology and its most important implementations.

Keywords: DNS64, Internet, IPv6 transition technologies, Network security, Stateful NAT64, STRIDE

1. Introduction

The deployment of IPv6, the new standard version of the Internet Protocol [1], has been very slow since IPv6 was defined in a draft standard RFC in 1998 [2], and its deployment accelerated only in the latest years [3]. Several *IPv6 transition technologies* [4] were developed, which address various communication scenarios. The application of a dual stack, that is all networking elements using both IPv4 and IPv6, would have been a good solution for a smooth transition, if IPv6 had been deployed before the depletion of the public IPv4 address pool, which actually happened in 2011. As the transition did not take place yet, and it is expected to last for decades [3], the coexistence of IPv4 and IPv6 necessitates the application of several IPv6 transition technologies for decades.

Unfortunately, the TCP/IP protocol stack, and the most important basic services, such as the Domain Name System [5], were not designed security aware. With the proliferation of the Internet connected devices, *network security* became an important issue. High numbers of potential security issues were identified at the different layers of the TCP/IP protocol stack. In this paper, we focus on the potential security issues of the IPv6 transition technologies. Our aim is to develop a systematic methodology for the identification of potential security issues of different IPv6 transition technologies. Our threat analysis model is based on the STRIDE approach [6], and its application to IPv6 transition technologies [7], which we extend in this paper. We demonstrate the viability of our method in two case studies: the detailed threat analysis of DNS64 [8] and a brief threat analysis of stateful NAT64 [9]. We also perform a survey of research papers, RFCs and other publications concerning the published threats of DNS64 and NAT64. We demonstrate that whereas the coverage of the different threats is rather sporadic in the different publications, our methodology can efficiently uncover them. Therefore, although STRIDE is based on industry best practices (that is, "art") rather than formal methods (e.g., formal protocol verification), we show that our methodology based on STRIDE can be a useful analytical framework to uncover the threats systematically, although there is no guarantee for its completeness.

The remainder of this paper is organized as follows. In section 2, our threat analysis methodology is presented. In section 3, a short introduction is given to the DNS64 and NAT64 IPv6 transition technologies. In section 4, the operation and viability of our threat analysis methodology is demonstrated on the detailed threat analysis of the DNS64 technology. In section 5, the most important

*Corresponding author

Email addresses: lencse@hit.bme.hu (Gábor Lencse), youki-k@is.naist.jp (Youki Kadobayashi)

Table 1: Threats and the security properties that guard against them (from Fig. 2 and Fig. 3 of [6])

Threat	Security Property	Description of the security property
Spoofing	Authentication	The identity of users is established (or you are willing to accept anonymous users).
Tampering	Integrity	Data and system resources are only changed in appropriate ways by appropriate people.
Repudiation	Non-repudiation	Users can not perform an action and later deny performing it.
Information disclosure	Confidentiality	Data is only available to the people intended to access it.
Denial of service	Availability	Systems are ready when needed and perform acceptably.
Elevation of privilege	Authorization	Users are explicitly allowed or denied access to resources.

Table 2: Susceptibility of the DFD elements to the different threats (from Fig. 5 of [6])

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

differences and the most interesting security issues of the stateful NAT64 technology are addressed. Section 6 contains a survey of security issues of DNS64 and NAT64 documented in the literature, and they are compared to our results. Section 7 provides further arguments for the individual security analysis of DNS64 implementations by showing practical examples. In section 8, our plans for future research are disclosed. Section 9 concludes our paper.

2. Our Threat Analysis Model

According to the book by Adam Shostack [10], the well-known and widely used systematic approach to threat modelling, called STRIDE, was invented by Loren Kohnfelder and Praerit Garg. As we could not access their paper from 1999 [11], we used a later paper from 2006 [6] and the aforementioned book [10]. The STRIDE acronym stands for *Spoofing*, *Tampering*, *Repudiation*, *Information disclosure*, *Denial of service*, and *Elevation of privilege*. Table 1 maps these threats to the security properties that guard against them and also gives short descriptions for the mentioned security properties. The STRIDE methodology facilitates the design of secure software (or systems) by means of *DFD* (Data Flow Diagram) *decomposition* of the system and a thorough analysis of the DFD components, which of the six aforementioned threats they are susceptible to. The possible elements of the DFDs are *data flows*, *data stores*, *processes*, and *interactors* (representing an external entity). The *trust boundaries* must also be marked. Table 2 shows the properties of the four elements whether they are susceptible to the given threat (marked by a "✓" sign) or not. The authors of [6] clearly state that the method does not give a guarantee for uncovering all the threats, but this systematic analysis may be of a great help in uncovering attacks that could have been overlooked or have not been thought of otherwise.

The book by Adam Shostack [10] emphasizes that the purpose of STRIDE is to help find the threats, and STRIDE is not for categorizing the threats: a given threat may be found in various ways, therefore, the threats do not have to be (and sometimes cannot be) categorized. The book calls the above described variant of STRIDE using Table 2 as "STRIDE-per-Element", and also mentions another variant called "STRIDE-per-Interaction" and mentions that the latter one is more complicated to use.

The STRIDE approach was applied to the security vulnerability analysis of IPv6 transition technologies in [7] in 2016. In that paper, the high number of IPv6 transition technologies were classified into a small number of general categories: *dual stack*, *single translation technologies*, *double translation technologies*, and *encapsulation technologies*. This approach made it possible to conduct a comprehensive threat analysis of nearly all the IPv6 transition technologies by examining only the elements of the DFDs constructed to represent the above four general categories, concerning the related threats of the STRIDE model.

In our current effort, we would like to complement the above method in two ways. Firstly, we need to add one new category for the DNS64 [8] IPv6 transition technology, which is not covered by any of the four aforementioned general categories. Secondly, the aforementioned comprehensive analysis, which was made much easier by the introduction of the general categories, is complemented by a more in-depth investigation at two levels: the level the individual IPv6 transition technologies and the level of their most important implementations. Fig. 1 illustrates the cost and benefit of the application of these methods to threat analysis. We prove the viability of this approach in section 4 and we also show examples why both these additional levels are necessary in section 4.4.3, section 4.4.4, section 4.6.1, section 4.8, and in section 7.

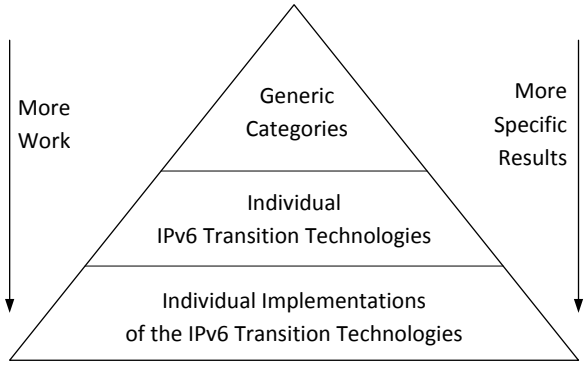


Figure 1: Method hierarchy: Costs and benefits of the different threat analysis methods

As for the implementations, we only deal with those that are free software [12] (also called open source [13]) for multiple reasons:

- Free software comes with source code and free software licenses explicitly allow the study of the source code, which can be essential for security analysis.
- Proprietary software usually does not include source code, and the licenses of certain vendors (e.g. [14] and [15]) do not allow reverse engineering and sometimes even the publication of benchmarking results is prohibited.
- Free software can be used by anyone for any purposes thus our results can be helpful for everyone.
- Free software is available free of charge for us, too.

Within the category of the free software implementations, we give further priority to those that are widespread and/or known to be stable and high performance (if such information is available). We have already published this reasoning in [4], where we have made a nearly exhaustive survey of the IPv6 transition technologies and ranked them regarding the importance of their security vulnerability analysis. DNS64 [8] and stateful NAT64 [9], used here as examples, were classified as *essential*, which is the group of the IPv6 transition technologies of utmost importance.

3. Introduction to DNS64 and Stateful NAT64

The DNS64 [8] and stateful NAT64 [9] IPv6 transition technologies can be used together for enabling IPv6-only clients to communicate with IPv4-only servers. This communication scenario is expressly relevant due to the depletion of the global IPv4 address pool, thus we consider the analysis of DNS64 and NAT64 important and inevitable because no other standard IPv6 transition technology exists for this scenario after NAT-PT was moved to historic status [16].

We demonstrate the operation of DNS64 and NAT64 in the example of an IPv6-only client and an IPv4-only web server taken verbatim from our conference paper [17]. Fig. 2 shows a scenario where an IPv6-only client communicates with an IPv4-only web server. The DNS64 server uses the 64:ff9b::/96 *NAT64 Well-Known Prefix* [18] for generating *IPv4-embedded IPv6 addresses* [18]. There are two prerequisites for the proper operation:

1. A DNS64 server should be set as the DNS server of the IPv6-only client.
2. Packets towards the 64:ff9b::/96 network are routed to the NAT64 gateway (routing must be configured that way).

Let us follow the steps of the communication:

1. The client asks its DNS server (which is actually a DNS64 server) about the IPv6 address of the `www.hit.bme.hu` web server.
2. The DNS64 server asks the DNS system about the IPv6 address of `www.hit.bme.hu`.
3. No IPv6 address is returned.
4. The DNS64 server then asks the DNS system for the IPv4 address of `www.hit.bme.hu`.
5. The 152.66.248.44 IPv4 address is returned.
6. The DNS64 server synthesizes an IPv4-embedded IPv6 address by placing the 32 bits of the received 152.66.248.44 IPv4 address after the 64:ff9b::/96 prefix and sends the result back to the client.
7. The IPv6-only client sends a TCP SYN segment using the received 64:ff9b::9842:f82c IPv6 address and it arrives to the IPv6 interface of the NAT64 gateway (since the route towards the 64ff9b::/96 network is set so in all the routers along the path).
8. The NAT64 gateway constructs an IPv4 packet using the last 32 bits (0x9842f82c) of the destination IPv6 address as the destination IPv4 address (this is exactly 152.66.248.44), its own public IPv4 address (198.51.100.10) as the source IPv4 address and some other fields from the IPv6 packet plus the payload of the IPv6 packet. It also registers the connection into its *connection tracking table* (and replaces the source port number by a unique one if necessary). Finally, it sends out the IPv4 packet to the IPv4-only server.
9. The server receives the TCP SYN segment and sends a SYN ACK reply back to the public IPv4 address of the NAT64 gateway.
10. The NAT64 gateway receives the IPv4 reply packet. It constructs an appropriate IPv6 packet using the necessary information from its state table. It sends the IPv6 packet back to the IPv6-only client.

The communication may continue. It seems the client is communicating to an IPv6 server. Similarly, the server

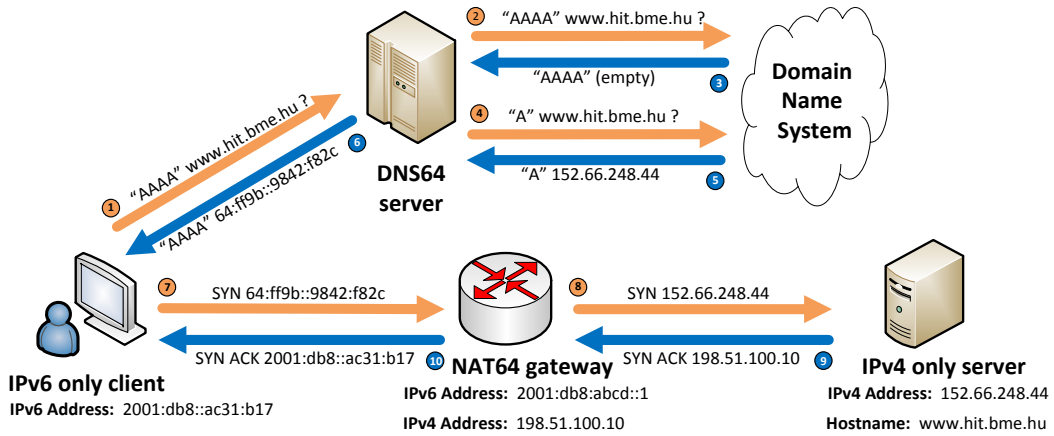


Figure 2: The operation of the DNS64+NAT64 solution: an IPv6-only client communicates with and IPv4-only server [17]

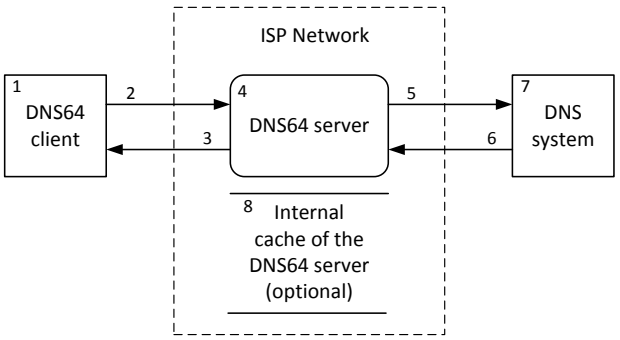


Figure 3: DFD for the threat analysis of DNS64

"can see" an IPv4 client. If it logs the IP addresses of the clients, then it will log the public IPv4 address of the NAT64 gateway.

Most client-server applications can work well with the DNS64 plus NAT64 solution, for more information see [19].

In practice, the usage of the NAT64 *Well-Known Prefix* has several hindrances, see sections 3.1 and 3.2 of [18]. Therefore, the network operators allocate a subnet from their own network for this purpose. It is called *Network Specific Prefix* [18].

4. Threat Analysis for DNS64

Fig. 3 shows the DFD for the threat analysis of DNS64. It contains two external entities (*DNS client* and *DNS system*), a process (*DNS64 server*) two bidirectional data flows (one *between the client and the DNS64 server* and another one *between the DNS64 server and the DNS system*) and a data store, which is the *internal cache of the DNS64 server*. We did not put a data flow symbol between the DNS64 server and the data store: the cache entries are stored inside the DNS64 process giving no surface for attack against data transfer between them. The trust boundaries are between the DNS64 server and the two external entities. Thus, the DNS64 server (including its inside cache) is a trusted element of the system, which has a low likelihood of being exploited. However, the others are untrusted, which have high likelihoods of being exploited [7].

Now we shall examine each element of the DFD for the threats that they may be susceptible to according to Table 2. The sequence number of the subsections within section 4 correspond to that of the DFD elements in Fig. 3. We also present a summary of the DNS64 threats in Table 3.

We note that STRIDE is a generic threat model and in a given situation some of the threats may be relevant and other ones may be not.

4.1. DNS64 Client

4.1.1. Spoofing

Spoofing here means source IP address spoofing, as there are no other ways for the identification of the clients.

There may be different motivations behind spoofing:

- The aim of the attacker may be a *DNS amplification attack* against the victim, whose address is spoofed. It is a kind of DoS (Denial of Service) attack, and its aim is to flood the victim with a high volume of traffic [20].
- Spoofing may also be used go get unauthorized access to the DNS64 service, if the DNS64 server uses an ACL (Access Control List) to enable the service to a limited set of clients only. In this case, the attacker needs another step to get the reply (e.g. evesdropping, routing redirect, etc.).
- In case of a DoS attack against the DNS64 server, the attacker can use spoofing simply to hide its identity or to circumvent rate limiting, see section 4.4.3.

Possible mitigation against spoofing is also described in [20].

4.1.2. Repudiation

There is no real point in repudiating a DNS request, except for the case, when the DNS request is sent as an attack, but then the attacker uses source address spoofing, as mentioned before. (Repudiation would be meaningful e.g. in the case of cash withdrawal or purchasing goods, but DNS and DNS64 are free of charge services.)

4.2. Data flow from DNS64 client to DNS64 server

4.2.1. Tampering

Tampering can be used to implement a kind of attack against a legitimate client by changing the requested domain name in the question section to another one and thus preventing the client from receiving an answer for its real question. This attack is similar to DoS attacks in its result (the client fails to receive an answer), however the name "DoS" is used for those attacks that are "absorbing resources needed to provide service" [10]. Thus we call it *Failure of Service* (FoS) attack.

Changing the question to one which results in a large volume of answers would be somewhat closer to the logic of the usual DoS attacks. However, for real DoS attacks, a large number of requests are necessary, thus changing only the client's requests are not appropriate.

Changing the destination IP address of the packet may be used to direct the packet to a fraudulent DNS/DNS64 server, which can implement any of the attacks described in section 4.4.1.

4.2.2. Information Disclosure

The DNS requests sent by a user may be interesting for several purposes, for example:

- Knowing the browsing habits of a user may help in sending targeted advertisements.
- Secret services or even criminals may analyze them for gaining information.
- It can be used for different attacks, see e.g. in section 4.3.1.

4.2.3. Denial of Service

For example, by flooding the network, an attacker may prevent a legitimate client from sending a query to the DNS64 server (and thus form using the service).

4.3. Data flow from DNS64 server to DNS client

4.3.1. Tampering

Tampering can be used to return false IPv6 address and thus deceive the client either implementing a FoS attack similar to the one mentioned in section 4.2.1, or to direct the traffic of the client to a computer controlled by the attacker. If it is used as a part of a larger attack, e.g. the attacker spoofs the website that the victim is intended to use and returns the IPv6 address of the spoof web site to the client, the attacker may be successful in *phishing* [21].

As for mitigation, DNSSEC [22] could help without DNS64, but its interference with DNS64 requires some discussion, please refer to section 4.9.

4.3.2. Information Disclosure

If the DNS64 server is intended to serve only a limited set of clients, an attacker may still use the DNS64 server by spoofing a legitimate client (see section 4.1.1) and eavesdropping the answer.

In general, there is no much point in revealing publicly available information, such as the one stored in the DNS system, except the case if they are intended to be available for internal use only. *DNS enumeration* [23] can be another example for information disclosure.

4.3.3. Denial of Service

The same as in section 4.2.3.

4.4. DNS64 server

Let us recall, that the DNS64 server is within our trust boundaries, thus it is considered as protected. The protection should involve both a physical protection (e.g. being locked into a room to prevent all unauthorized persons from physically approaching it) and firewall protection (to try to prevent unauthorized traffic from reaching it).

4.4.1. Spoofing, Tampering, Information Disclosure

Under the spoofing of a DNS64 server, we mean that an attacker operates an unauthorized DNS64 server, which replies instead of the authorized one. We contend that there is no logical difference, whether another computer is used to execute the counterfeit DNS64 server program, or the original DNS64 server software is replaced by an appropriately patched one, which performs certain extra functions. Therefore, we handle the case of spoofing the DNS64 server together with the case of tampering with the DNS64 server.

The spoofed (or tampered) DNS64 server may do different harmful things such as tampering with the DNS data (e.g. resulting in phishing, see section 4.3.1) or information disclosure (see section 4.2.2), which can be very efficient for the attacker if they can implement the level of the DNS64 server. To do further harmful things is only left to the fantasy and bad will of the attacker (e.g. intermittent and/or selective delayed service for certain computers).

The implementation of these kinds of attacks needs the cooperation, or at least significant negligence of one or more trusted persons responsible for the operation of the network. Thus, we believe the only way to prevent these kinds of attacks is to employ well trained, careful, and trustworthy professionals. DNSSEC might help against several attacks except this one, please see section 4.9 for the details.

Whereas a spoofed or tampered DNS64 server is ideal for information disclosure, it is an important question, is information disclosure possible with a genuine untouched DNS64 server? If yes, we consider it a bug in the DNS64 implementation.

4.4.2. Repudiation

There is no real point in the repudiation of the acts of a DNS64 server (a DNS reply to a client or a DNS request to the DNS system), except for the case, when they belong to an attack, which may happen if the DNS64 server is spoofed, see section 4.4.1.

4.4.3. Denial of Service

A DoS attack may be implemented by malicious but legitimate clients sending a huge number of requests. Some DNS64 servers support *rate liming*, that is, no more than the specified number of requests per second are served for a given client. For example, BIND supports this function [24].

A crafty attacker may spoof the IP addresses of several legitimate clients to by-pass rate limiting and also to hide his/her identity, as mentioned in section 4.1.1.

High performance can be a kind of mitigation of DoS attacks. We have compared the performances of four DNS64 implementations in [25], and we plan to make further performance comparisons. See the details in section 4.10.1.

However, there is a completely different way of making the services of some vulnerable DNS servers unavailable (thus implementing a DoS attack). DNS names follow a special encoding called message compression [26]. It can also use pointers, which we illustrated in section II.A of [17] (see Figures 4-6). An attacker may use a malformed domain name, which contains a pointer that points to itself in order to cause an infinite loop for a vulnerable DNS server [27].

We note that there may be some other specific attacks which try to confuse the DNS server by sending a request or reply with corrupted internal structure. For example, stating higher number or Resource Records than actually included in the reply may cause an improperly written implementation to produce segmentation fault.

An appropriate firewall using deep packet inspection can protect the DNS64 server against a malformed domain name attack, but we recommend the usage of well written DNS64 servers that are not vulnerable to the attack.

We note that the possibility of rate limiting (or the lack of it), the different performances of DNS64 implementations and their vulnerability to malformed domain name attacks are important arguments for their individual analysis.

4.4.4. Elevation of Privilege

An elevation of privilege attack may be performed by malicious (patched) DNS64 software coming from "inside" persons (as mentioned in section 4.4.1), to gain further privileges (root rights) on the server, which the DNS64 software is executed by. However, this is not the only way for an elevation of privilege attack to happen. DNS64 servers receive input in both the queries from the clients and the replies from the DNS system, thus they can be susceptible to *buffer overflow attack*s [28]. If a buffer overflow attack is successful, then the attacker may execute his program code using the rights (privileges) of the DNS64 server. Now, he is able to perform any activities described in section 4.4.1 or 4.4.3. In addition to that, the attacker may also try to find further security holes to gain further rights.

We note that some of the attacks using corrupted internal structure messages (mentioned in section 4.4.3) may also aim to gain control over the system.

However, we also note that these kind of attacks are very sophisticated. Even in the case of the well-known buffer overflow attack [28] its implementation is always CPU type specific, because the code to be executed must be written in the machine language of the given computer [29]. And of course, the writer of the malicious code must know the attacked software very well.

As for mitigation, the book by Adam Shostack [10] has several recommendations:

1. the usage of Address Space Layout Randomization (ASLR) may prevent a successful buffer overflow attack
2. programming in type-safe languages (like Java or C#) instead of C can make many of control flow / memory corruption attacks harder
3. executing the server programs in sandboxes may prevent the attacker from getting control over the whole system.

Again, an appropriately set firewall using deep packet inspection can protect the DNS64 server against a buffer overflow attack, but we recommend the use of well written DNS64 servers that are not vulnerable to the attack, which requires individual analysis. Our approach can be justified by performance considerations. It is much more efficient to execute good software than to execute a vulnerable one plus a second one to cover the vulnerabilities of the first one.

4.5. Data flow from DNS64 server to DNS system

We note that some similarities can be observed between the vulnerabilities of the data flow from the DNS64 server to the DNS system and the vulnerabilities of the data flow from the client to the DNS64 server (see section 4.2), but there are significant differences as well.

4.5.1. Tampering

Very similar things can be said to the contents of section 4.2.1, only the names of the endpoints are different.

4.5.2. Information Disclosure

As the information cannot be bound to a specific user, but it belongs to the whole community that uses the DNS64 server, the statistics are less useful both for targeted advertisements or for gaining information about the users.

However, the knowledge of the requests can be used for an even more effective attack than in section 4.3.1. See its details in section 4.6.1.

4.5.3. Denial of Service

Similarly to section 4.2.3, an attacker may prevent the DNS64 server from sending a query to the DNS system by flooding the network, but here, higher performance may be needed due to the probably one order of magnitude higher line speeds.

4.6. Data flow from DNS system to DNS64 server

Again, similarities can be observed to section 4.3, but there are significant differences, too.

4.6.1. Tampering

Tampering can be much more effective than in section 4.3.1, if the DNS64 server uses caching.

DNS cache poisoning (also referred to as DNS spoofing) is an attack, which targets to load corrupt data into the cache of a DNS server [30]. As a result, the traffic of the clients that receive the false IP addresses from the server, will be diverted. DNSSEC [22] could help, but its deployment is too low to be an effective protection for all the domain names.

The entries (called Resource Records, abbreviated as RR) are stored in the cache of a DNS server until their Time-To-Live (TTL) expires. Popular domain names (especially those with high TTL) are somewhat protected against cache poisoning as they are usually cached. However, in some cases cached data is overwritten if the trustworthiness of the newly received data is ranked higher. For ranking data, see section 4.5.1 of [31]. Reference [30] gives a detailed analysis of the trust levels used by BIND and shows that an "RRset is overwritten if the trust level of the received RRset is higher or equal to the cached one and its TTL is longer". It also mentions that Unbound behaves similarly, but MaraDNS behaves completely differently.

From this, one of the most important lessons is that different implementations may significantly differ from each other in their vulnerabilities to different threats. This justifies our statement that it is worth examining the individual implementations of a given IPv6 transition technology.

In our threat model, a cache poisoning attack could be implemented in different ways, one of them requires:

1. Either the cooperation of a legitimate client, which would send a request for the targeted domain name, or the attacker may be able to spoof a legitimate client or tamper with the message flow from the client to the DNS64 server.
2. Either spoofing an authoritative DNS server or tampering with the message flow from the DNS system to the DNS64 server.

As for mitigation, DNSSEC [22] could deliberately help without DNS64 (for the domains, which it is used), and we point out in section 4.9 that it still protects against DNS cache poisoning, but with limitations concerning the validation.

4.6.2. Information Disclosure

The same as in section 4.3.2.

4.6.3. Denial of Service

The same as in section 4.5.3.

4.7. DNS System

4.7.1. Spoofing

In practical terms, there is no difference, whether the DNS system is spoofed or the message flow from the DNS system to the DNS64 server is tampered. We described the resulting DNS cache poisoning in section 4.6.1.

4.7.2. Repudiation

There is no real point in the repudiation of a genuine DNS reply. The sender of a forged reply spoofs a legitimate DNS server. DNSSEC could be a good solution if it were deployed.

4.8. Internal Cache of DNS64 Server

Some DNS64 implementations support caching and others do not, thus even the existence of caching is implementation dependent. Those that support caching, may have different cache control algorithms.

Caching is done internally by those DNS64 implementations that support it, thus the general vulnerabilities of the data stores (tampering, information disclosure and denial of service) may not be immediately exploited, only through the DNS64 implementations (as processes), and we have already dealt with them in section 4.4. The reason we still included the internal cache in the DFD is to show that it is worth considering. Although its entries may not be accessed or manipulated immediately, some attacks may target its entries. For example, an attacker may send requests for a AAAA record of a domain name to the DNS64 server to find out the TTL value of the entry, which may be useful in a cache poisoning attack. Moreover, a legitimate client (or a spoofed one) may also manipulate the contents of the cache by simply sending high number of AAAA record requests for different domain names. The attacker may reduce the performance of the server by simply overwriting the valuable cache contents with domain names, which are useless for the users. In this case, the users will still get correct answers, but somewhat later. The cache poisoning attack, which we introduced in section 4.6.1, is an indirect tampering with the contents of the cache. And cache poisoning may be efficiently supported by sending fake name

resolution requests to the DNS64 server at high frequency so that the attacked domain name may be removed from its internal cache.

4.9. DNS64 and DNSSEC

Using DNSSEC [22], a *security aware* and *validating* client may check whether the data from a DNS server has been modified. Of course, it works only if DNSSEC is implemented from the root zone to the zone of the domain name in question (or an appropriate trust anchor is used) and DNSSEC is supported by the recursive DNS server, which performs the name resolution for the client. Now, we explain, how it works from the client side. In its query, the client sets both the DO ("DNSSEC OK") and the CD ("Checking Disabled") bits to signal to the DNS server that the client is security aware (it may process DNSSEC information) and that the client requires to receive validation data (it does not entrust the server to perform the validation). This is the best possible scenario, because in this case, the client is able to discover if tampering happened anywhere. Unfortunately, this scenario is incompatible with DNS64 because the DNS64 server itself "tampers with" the data: it synthesizes the so-called *IPv4-embedded IPv6 address*, as we described it in section 3. This is also confirmed in section 3 of [8], where a detailed analysis is given of the possible scenarios of DNS64 and DNSSEC interference. The following is a workable scenario: the client sets the DO bit to 1 and the CD bit to zero requesting the DNS64 server to perform the DNSSEC validation. In this case, the DNS64 server validates the data. "If it fails, it returns RCODE 2 (Server failure); otherwise, it returns the answer." [8]

For us it means, that we are protected from cache poisoning, due to the DNSSEC validation performed by the DNS64 server, but we are not protected against tampering performed on the data flow from the DNS64 server to the client or against the spoofing (tampering) of the DNS64 server itself.

The solution is simple but not easy: the deployment of both native IPv6 and DNSSEC will result in a more reliable DNS infrastructure. For further reading about the issue, please refer to [32], which also presents some adoption statistics both for IPv6 and DNSSEC as of its writing in 2016.

4.10. Going to Implementation Level

In theory, the implementations should be analyzed in all the aspects their behavior may be different. As this can require a lot of resources (humans, computers) we recommend the ranking of the implementations, and selecting a few of them. The ranking may be based on their performance, popularity, etc.

4.10.1. Performance comparison of DNS64 implementations

There are several free software DNS64 implementations. We have examined the stability and performance of BIND [33], TOTD [34], Unbound [35], and PowerDNS [36] in our aforementioned paper [25]. They all proved to be stable solutions, but they showed different performances. One of the most important lessons learned from their performance comparison was that the DNS64 implementations scaled up very differently as a function of CPU cores on the executing server. On the one hand, we could only go up to four CPU cores using the then available measurement methodology, software, and hardware, but on the other hand, modern servers usually have 16 or more CPU cores. Therefore, we believe that those results are not enough for the performance ranking of the DNS64 server implementations. Since then, we have developed a benchmarking methodology for DNS64 servers [37], which is also a part of our new RFC on benchmarking methodology for IPv6 transition technologies [38]. We have developed a measurement tool, `dns64perf++` [39], which complies with the compulsory requirements of the draft. We have also added the optional feature of testing the caching performance of DNS64 servers [40].

During the first review process of this paper we have benchmarked BIND, PowerDNS and Unbound using 16-core servers at NICT StarBED, Japan. We have found significant differences both in their single core DNS64 performances and in their scale-up from 1 to 16 CPU cores: whereas Unbound showed the highest single core performance, PowerDNS scaled up the best. We have reported a serious issue regarding BIND: its DNS64 performance did not increase from 4 to 16 CPU cores at all. Please refer to [41] for further details.

4.10.2. Popularity of DNS64 server implementations

We do not have direct data about the popularity of DNS64 servers, thus we can only suppose that their popularity is similar to that of the DNS servers. This is surely a rough approximation, because some DNS servers do not support DNS64 and there are expressly DNS64 servers, which do not implement traditional DNS services such as authoritative or recursive DNS service.

ISC states that "BIND is far the most widely used DNS software on the Internet" [33] and as it supports DNS64, thus the security analysis of BIND is a must.

4.10.3. Transaction ID prediction attack

There is another interesting issue, which we have discovered about the TOTD DNS64 implementation [42]. It used an incremental counter for Transaction IDs, which were trivial to predict, therefore it was very much susceptible to Transaction ID prediction attacks. We have mitigated this vulnerability by introducing pseudorandom transaction IDs [42]. Our security patch has been included into the source code of TOTD 1.5.3 [43].

5. Stateful NAT64 in a Nutshell

We presume that the systematic security analysis of the DNS64 servers was a sufficient demonstration of our se-

Table 3: Summary of DNS64 threats (please refer to the text of the corresponding subsection of section 4 for explanation)

DFD element	Threat	Possible attacks or benefits of the attacker
1	spoofing	DNS amplification attack (DoS); unauthorized access to service; identity hiding; circumventing rate limiting (see section 4.1.1)
	repudiation	- (see section 4.1.2)
2	tampering	failure of service (similar to DoS) (see section 4.2.1)
	information disclosure	collecting information for different purposes (see section 4.2.2)
	denial of service	denial of service attack (see section 4.2.3)
3	tampering	failure of service; part of phishing (see section 4.3.1)
	information disclosure	unauthorized access to service; DNS enumeration (see section 4.3.2)
	denial of service	denial of service attack (see section 4.3.3)
4	spoofing, tamp., inf. disc.	phishing; failure of service; collecting information for different purposes; etc. (see section 4.4.1)
	repudiation	- (see section 4.4.2)
	denial of service	different kinds of denial of service attacks, including malformed domain name (see section 4.4.3)
	elevation of privilege	buffer overflow attack; DNS message with corrupted internal structure
5	tampering	failure of service (similar to DoS) (see section 4.5.1)
	information disclosure	collecting information for different purposes (see section 4.5.2)
	denial of service	denial of service attack (see section 4.5.3)
6	tampering	DNS cache poisoning (see section 4.6.1)
	information disclosure	unauthorized access to service; DNS enumeration (see section 4.6.2)
	denial of service	denial of service attack (see section 4.6.3)
7	spoofing	DNS cache poisoning (see section 4.7.1)
	repudiation	- (see section 4.7.2)
8	(only indirect attacks)	DNS cache poisoning; reducing the efficiency of caching; etc. (see section 4.8)

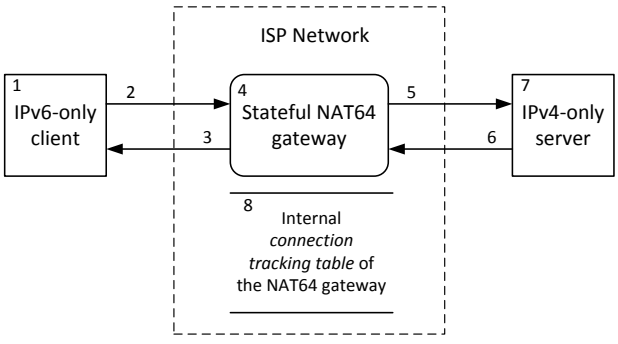


Figure 4: DFD for the threat analysis of stateful NAT64

curity analysis method. As for NAT64, we would like to touch upon some interesting differences only.

Fig. 4 shows the DFD for the threat analysis of stateful NAT64, which looks very similar to that of DNS64. An important difference is that the connection tracking table is not optional, as NAT64 is stateful. There is another important difference. Whereas the cache of the DNS64 server stores information that originated from the replies of the DNS system, the entries of the connection tracking table of the stateful NAT64 gateway are created solely on the basis of the information available in the header of the packets from IPv6-only clients thus this technology gives less surface for attacks.

5.1. Possible DoS Attacks Against Stateful NAT64

We present a summary of the most interesting DoS attacks against stateful NAT64 in Table 4.

5.1.1. Exhaustion of the connection tracking table

The exhaustion of the connection tracking table may be used for a DoS attack. This attack can be easily implemented by a legitimate client using its own IPv6 address as source address, but forging source IPv6 address may somewhat help the attacker in hiding. As for the target

address, the attacker needs to use all different IPv4 embedded IPv6 addresses, which can be easily synthesized in the knowledge of the prefix used by the DNS64 server. Maximum 2^{32} number of target addresses can be synthesized[1], and if it is not enough, the number of different 5-tuple combinations (source IP address, source port number, target IP address, target port number, and protocol number) may be easily increased by using different (forged) source addresses, or different source port numbers (perhaps the destination port numbers are limited due to firewall policies).

We note that – depending on the implementation – a very high number of entries in the connection tracking table may have a negative impact on its lookup speed.

5.1.2. Exhaustion of the source port number and public IPv4 address pool

The depletion of the pool of available source port numbers at the NAT64 gateway may also be a solution for a DoS attack. The gateway may use 64k (or 63k, if the 0-1023 system port range is left out) number of source ports per public IPv4 address.

The traditional type of Network Address and Port Translation (NAPT) always replaces the source port number by a unique one at the NAPT device. The extended algorithm also includes the destination IP address and port number into the tuple, and replaces the source port number only if the current tuple would be identical with an existing one in the connection tracking table [44]. Thus, depending on the type of the NAT64 implementation, the task of the attacker may be relatively easy or very hard.

5.1.3. Brute force DoS attacks

Implementing a simple "brute force" DoS attack against the stateful NAT64 gateway may not be simple because the stateful NAT64 function is a much less demanding task than DNS64. See section 5.2.1 for the details. Of course, the CPU, memory (size or bandwidth), or network capacity (understanding as line capacity measured in bit per second or interrupt processing capacity measured in packets per second, which can be significantly reduced by *interrupt coalescing*) of a NAT64 gateway may be exhausted by an appropriately powerful "brute force" DoS attack.

5.2. Implementations

5.2.1. Performance analysis

As for free software stateful NAT64 implementations, we have experience with PF (Packet Filter) of OpenBSD, and the combination of the stateless TAYGA and the Netfilter of Linux (also called `iptables` after the name of its user interface tool). We examined and compared their stability and performance using ICMP traffic [45]. To test the worst case behavior of the examined NAT64 gateways, we used all different target addresses, which were redirected to the same "responder" computer by an `iptables` rule. Both NAT64 implementations proved to be stable, but PF outperformed TAYGA more than three-fold by means of served requests per second, which was not a surprise, as TAYGA (stateless NAT64) is implemented in user space and the stateful NAT part was done by another software (`iptables`), whereas PF runs in kernel space and performs the full stateful NAT64 by itself. The experiments were also executed by using TCP and UDP traffic [46].

If an attacker tries simple "brute force" attack, the significant performance difference may mean, that PF can survive a DoS attack, which can seriously decrease the performance of the TAYGA+`iptables` system.

However, our experience shows that the same DUT (Device Under Test), which was intentionally a very old hardware to be able to overload by using our then available technology, could serve less than 500 AAAA record queries per second as a DNS64 server [47], whereas it could process more than 22,000 packets per second (and their replies) as a NAT64 gateway [45]. Thus simple "brute force" attacks do not seem to be easy to implement.

Ecdysis and Jool are two other free software stateful NAT64 implementations, which we did not test yet, but we consider them good candidates.

We plan to compare their performances using the benchmarking method described in RFC 8219 [38], however currently no stateful NAT64 benchmarking tool is available, which fully implements its measurements for stateful NAT64 testing. The only benchmarking tool that complies with RFC 8219 implements only the stateless NAT64 tests [48].

5.2.2. Results concerning port number exhaustion

We have examined the source code of `iptables` and found that it implements extended NAPT [49]. Thus, it will efficiently resist port number exhaustion attacks.

Since the extended NAPT algorithm was published more than 10 years ago [44], we suppose that the other NAT64 implementations also use it, but we have not verified this yet.

6. Related Work

We survey the results of other researchers concerning the security issues of DNS64 and NAT64 to examine how efficiently our methodology could discover the threats that are known about these technologies.

Unlike in the case of e.g. 6to4 (the security issues are discussed in a separate RFC [50]), or NAT-PT, the predecessor of NAT64+DNS64, which was deprecated, inter alia, due to its security issues [16], we have found very few published results concerning the security issues of DNS64 and NAT64.

[1] The number of possible IPv4 addresses may be less than 2^{32}, because Class D (224.0.0.0/4, multicast block) and Class E (240.0.0.0/4, experimental block) addresses may be unusable for the attack.

Table 4: Summary of the most interesting DoS attacks against stateful NAT64

Type of DoS attack
exhaustion of the connection tracking table (see section 5.1.1)
exhaustion of the source port number and public IPv4 address pool (see section 5.1.2)
simple "brute force" DoS attack (see section 5.1.3)

6.1. Security issues of DNS64

6.1.1. DNS64 threats in the literature

Unfortunately, neither [51] or [52] nor other research papers deal with the security issues of DNS64, except for our paper [42].

The defining RFC of DNS64 [8] contains the usual section of "Security Considerations" (section 8), which mentions the following things:

1. DNS64 is subject to the security considerations of DNS (we discuss them in section 6.1.3).
2. DNS64 may interfere with DNSSEC, because it modifies the messages. (The RFC provides an in depth description of the case, we have addressed its essence in section 4.9.)
3. If an attacker manages to change the prefix used by the DNS64 server, it may result in:
    - a DoS attack (if the resulting IPv6 addresses are not assigned to any device)
    - a flooding attack (if the resulting IPv6 addresses are assigned to devices that do not wish to receive the traffic),
    - an eavesdropping attack (in case, if the prefix is routed through the attacker).

We have also found similar issues in other RFCs. The RFC about the discovery of the IPv6 prefix used for IPv6 address synthesis [53] mentions in its section 3.1 that if the client node uses an insecure channel between itself and the DNS64 server, then the attacker may influence the prefix discovery procedure, which "may result in denial-of-service, redirection, man-in-the-middle, or other attacks." As a solution, it recommends the use of a secure channel between the host and the DNS64 server, "for example, an IPsec-based virtual private network (VPN) tunnel or a link layer utilizing data encryption technologies" [53]. As an alternative, RFC 6889 [54], which also points out the problem that a client using DNS64 server may not perform its own DNSSEC validation, recommends that the host should perform its own DNS64 synthesis.

Section 2.7.3.2 of the Internet Draft on the operational security considerations for IPv6 networks [55] also mentions that "DNS64 has an incidence on DNSSEC".

Section 7 of another RFC about the analysis of solution proposals for hosts to learn NAT64 prefix [56] mentions the problem, that IP address of the DNS64 server configured in the clients may not be reliable, if they received it by DHCPv6. "Therefore, if, for example, the host cannot trust DHCPv6, it cannot trust the DNS server learned via DHCPv6 either, unless the host has a way to authenticate all DNS responses (e.g., via DNSSEC)." [56]

6.1.2. Expressly DNS64 threats covered by our results

We can state that we have found all the possible types of threats including tampering, although we did not think of the kind of its exploitation that the attacker only changes the prefix, but it is only one subtype of tampering, which was covered in general.

6.1.3. General DNS threats in the literature

There are a high number of papers addressing the threats of the Domain Name System.

Reference [57] enumerates the following threats:

- DNS cache poisoning
- tampering with the zone file at the authoritative DNS servers
- DDoS (Distributed Denial of Service) by using DNS amplification or DNS reflection.

Reference [58] lists the following ones:

- DNS cache poisoning
- DNS protocol attacks, that is, malformed DNS queries or responses. ("Examples of such attacks include malformed packets, code insertion, buffer overflows, memory corruption, NULL pointer de-reference or the exploitation of specific vulnerabilities." [58])
- DNS redirection (MITM) attacks, mentioning DNS changer and DNS replay, or illegitimate redirection as examples.
- DNS fast fluxing (changing the DNS records at extremely high frequency, e.g. using 60s TTL, to assist the attacker in hiding).
- DNS tunneling (using the DNS communication as a covert channel for malicious communications to bypass traditional defense mechanism, e.g. firewalls).
- Domain phishing.

Reference [59] is an older and more general paper about the internet infrastructure security, and it addresses the attack against the DNS system in two ways. It deals with the possible "impacts" of the attacks and the "types" of the attacks. Under "impacts" it enumerates:

- Denial-of-Service, with the meaning that the client may not connect to the desired server, achieved e.g., by sending back negative responses (that the domain does not exist) or an incorrect IP address. (We called it as FoS.)
- Masquerading, with the meaning that presenting a compromised DNS server for the users.
- Information Leakage (the same as information disclosure in our terminology).
- Domain Hijacking, with the meaning that the attacker re-registers a legitimate domain with forged data (exploiting the insecure nature of the communication channel between the domain owner and the registrar).

As for types, it lists:

- Cache Poisoning
- Server Compromising (the attacker takes control over the DNS server and thus the attacker may do whatever wishes).
- Spoofing (the same as "masquerading" above).

A more current survey on the DNS threats [60] mentions virtually the same types of threats that we have already enlisted above.

And there is an RFC about the threat analysis of DNS [61], which addresses the following threats:

- Packet Interception (including both information disclosure and tampering)
- ID Guessing and Query Prediction (means the guessing of the 16-bit Transaction ID combined with the knowledge of prediction of the QNAME and QTYPE values to be able to insert a bogus response).
- Name Chaining, which is a special case of cache poisoning, based on using a kind of RR, e.g. CNAME or NS, which can redirect a victim's query to a location of the attacker's choice.
- Cache Poisoning (mentioned at the previous threat, but it can be a more general type, too).
- Betrayal by Trusted Server (a DNS server misbehaves due to a bug or by being tampered).
- Denial of Service
- Authenticated Denial of Domain Names (It is debated whether there is a requirement for authenticating the non-existence of a name or not.)
- Wildcards (special handling of names starting with the label "*", for more information see section 4.3.3 of [5]).

6.1.4. DNS threats covered by our results

Not all DNS threats are relevant to the DNS64 service, e.g. tampering with the zone files at the authoritative DNS servers, the re-registration of a domain by an attacker, or changing the IP-address of the DNS/DNS64 server in the client, etc. are general attacks against the Domain Name System, and we consider that these types of attacks are beyond the scope of our paper. However, those types of attacks that are relevant to DNS64 were covered by our detailed threat analysis of DNS64 in section 4.

6.2. Security issues of NAT64

6.2.1. Threats in the literature

Reference [51] states: "The main security issue of NAT64 is DoS (Deny of Service) attack on the binding table, with ingress filtering on the IPv6 side as the solution." Reference [52] states: "In terms of security, the main threats against NAT64 are potential Denial-of-Service attacks, targeted to consume the scarce NAT64 resources including memory, processing power, and the IPv4 transport-address pool. NAT64 can mitigate these attack vectors by limiting the amount of resources assigned to different purposes (e.g. the amount of memory used for temporarily storing fragments waiting for reassembly)." Because of the common authors, it is not surprising that the RFC defining stateful NAT64 [9] also mentions more or less the same vulnerabilities in its section 5.3, namely different DoS attacks aimed:

- to consume transport addresses by initiating high number of bindings initiated from different IPv6 transport addresses
- to consume the memory of the NAT64 device by sending high number of fragments to be stored
- to consume the memory of the NAT64 device in the form of session and/or binding table entries by sending high number of SYN segments.

The Internet Draft on the operational security considerations for IPv6 networks [55] mentions that: "A specific issue with the use of NAT64 is that it will interfere with most IPsec deployments unless UDP encapsulation is used."

6.2.2. Threats covered by our results

Our results exactly cover the exhaustion of the binding table (called connection tracking table in section 5.1.1) as well as the exhaustion of the transport addresses (called public IPv4 addresses and port numbers in section 5.1.2). The "brute force" DoS, which we mentioned in section 5.1.3, generally covers the CPU, memory and communication capacity exhaustion, although we did not

explore the possible implementations of exhausting the memory capacity by either SYN or fragmentation attacks.

Only the interference with IPsec was not addressed, which is not a specific "threat" as we used above, but a "general feature" of NAT64.

Thus, our methodology can be considered to be successful in uncovering the known threats of NAT64, although we did not do a full scope analysis, but only examined those issues seemed to be interesting enough.

6.3. Discussion

Our methodology for the identification of potential security issues of different IPv6 transition technologies proved to be viable for finding the documented vulnerabilities of DNS64 and NAT64 in a systematic way. This is very important, because the examined research papers usually reported overlapping but significantly different attack vectors. Thus, a systematic method is definitely of a great help. However, we would like to emphasize that we do not mean to suggest any guarantee of covering all possible threats by using our methodology. The paper on STRIDE [6] explicitly states that there is no such guarantee.

Another important observation is that while we did not have any difficulty in finding research papers addressing the security issues of stateful NAT64, we did not find any research papers on the potential security issues of DNS64 (except for our earlier paper, which dealt with a specific DNS64 implementation, TOTD). Our experience was rather similar, when we surveyed the performance analysis of DNS64 and NAT64 implementations:

"The performance of the TAYGA NAT64 implementation (and implicitly of the TOTD DNS64 implementation) is compared to the performance of NAT44 in [62]. The performance of the Ecdysis NAT64 implementation (that has its own DNS64 implementation) is compared to the performance of the authors' own HTTP ALG in [63]. The performance of the Ecdysis NAT64 implementation (and implicitly the performance of its DNS64 implementation) is compared to the performance of both the NAT-PT and an HTTP ALG in [64]." [25]

DNS64 was omitted from the four categories of [7]. It seems from these examples that DNS64 is not treated as co-ordinate with NAT64, but rather a subordinate, less important protocol than NAT64. While we agree that users' datagrams are carried by the NAT64 gateway which is the lion's share of the work, NAT64 is unusable[2] without DNS64. The poor performance of the DNS64 server directly influences the users' Quality of Experience (QoE), and a successful DoS attack against the DNS64 server makes the IPv4-only servers unavailable for the IPv6-only clients. Tampering with the DNS64 server (or only with its cache contents as cache poisoning) may cause significant harm to the users (e.g. by phishing). Therefore, we conclude that the threat analysis of DNS64 is essential. Our threat analysis of the DNS64 technology is the first step to fill in this important gap, and it definitely should be completed by the threat analysis of the most important DNS64 implementations.

[2]We are aware of the fact that stateful NAT64 is also used as the PLAT implementation of 464XLAT [65], but now we talk about the solution, which enables an IPv6-only client to communicate with an IPv4-only server, and DNS64 is inevitable in this communication scenario.

7. Cache Poisoning Vulnerability Analysis of DNS64 Implementations

During the first review process of this paper we performed a detailed cache poisoning vulnerability analysis of several DNS64 implementations [66]. On the basis of RFC 5452 [67], we have shown the three most important countermeasures against cache poisoning:

- usage of cryptographic random numbers as Transaction IDs
- usage of cryptographic random numbers as source port numbers
- refraining from sending out multiple equivalent queries (having identical QNAME, QTYPE, and QCLASS fields) concurrently.

We have also created a methodology and a testbed to determine, whether a specific DNS64 server implemented these three countermeasures against cache poisoning. The test results of several free software DNS64 implementations are shown in Table 5. Please refer to [66] for all the details of our tests. Although TOTD and mtd64-ng [68] do not support caching, and thus cache poisoning is not applicable to them, mtd64-ng is going to be enabled for caching in the near future, thus it must apply all three countermeasures against cache poisoning before it can be used as a production class DNS64 server.

These results also justify our statement that the individual security analysis of the implementations is inevitable.

8. Plans for Future Research

We are planning to select the most important free software DNS64 server and NAT64 gateway implementations (both on performance and deployment basis) and submit them to detailed security analysis.

We also plan to select further IPv6 transition technologies from our survey [4] for the identification of their potential security issues as well as that of their most important free software implementations.

9. Conclusion

We have presented a threat analysis model for IPv6 transition technologies by extending an earlier model that applied the STRIDE approach to the potential security vulnerability analysis of IPv6 transition technologies and

Table 5: Summary of the vulnerability test results [66]

DNS64 Implementation	Attack Type			
	Transaction ID Prediction	Source Port Number Prediction	Multiple Equivalent Queries	DNS Cache Poisoning
BIND 9.9.5	no problem found	no problem found	protected	no problem found
TOTD 1.5.2	vulnerable	vulnerable	vulnerable	not applicable
TOTD 1.5.3	protected	vulnerable	vulnerable	not applicable
mtd64-ng 1.1.0	vulnerable	vulnerable	vulnerable	not applicable
PowerDNS 3.6.2	no problem found	no problem found	protected	no problem found
Unbound 1.6.0	no problem found	no problem found	protected	no problem found

dealt with categories for the IPv6 transition technologies, at two levels: the level of the given IPv6 transition technologies and the level of their implementations.

We have demonstrated the operation of our method for the security vulnerability analysis of IPv6 transition technologies with the detailed threat analysis of DNS64 servers, and we have also shown the necessity of further analysis. We have also touched upon some interesting security issues of NAT64 gateways.

We have shown the efficiency of our method by comparing our results with the threat vectors can be found in the literature, and we have pointed out that the in-depth threat analysis of the DNS64 technology and its implementations is essential.

We also presented our plans for future research concerning the threat analysis of the most important free software DNS64 and NAT64 implementations, as well as further IPv6 transition technologies and their implementations.

Acknowledgement

This work was supported by the International Exchange Program of the National Institute of Information and Communications Technology (NICT), Japan.

References

[1] S. Deering, R. Hinden, Internet protocol, version 6 (IPv6) specification, IETF RFC 8200 (STD: 86) (2017). doi:10.17487/RFC8200.

[2] S. Deering, R. Hinden, Internet protocol, version 6 (IPv6) specification, IETF RFC 2460 (1998). doi:10.17487/RFC2460.

[3] M. Nikkhah, R. Guérin, Migrating the Internet to IPv6: An exploration of the when and why, IEEE/ACM Transactions on Networking 24 (4) (2016) 2291–2304. doi:10.1109/TNET.2015.2453338.

[4] G. Lencse, Y. Kadobayashi, Survey of IPv6 transition technologies for security analysis, IEICE Tech. Rep. 117 (187) (2017) 19–24.

[5] P. Mockapetris, Domain names – concepts and facilities, IETF RFC 1034 (1987). doi:10.17487/RFC1034.

[6] S. Hernan, S. Lambert, T. Ostwald, A. Shostack, Threat modeling: Uncover security design flaws using the STRIDE approach, MSDN Magazine 6 (11) (2006) 68–75.

[7] M. Georgescu, H. Hazeyama, T. Okuda, Y. Kadobayashi, S. Yamaguchi, The STRIDE towards IPv6: A comprehensive threat model for IPv6 transition technologies, in: Proc. 2nd International Conference on Information Systems Security and Privacy, Rome, Italy, 2016. doi:10.13140/RG.2.1.2781.6085.

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

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

[10] A. Shostack, Threat Modeling: Designing for Security, Wiley & Sons, Indianapolis, Indiana, USA, 2014.

[11] L. Kohnfelder, P. Garg, The threats to our products, Microsoft Interface (1999).

[12] Free Software Foundation, The free software definition. URL http://www.gnu.org/philosophy/free-sw.en.html

[13] Open Source Initiative, The open source definition. URL http://opensource.org/docs/osd

[14] Cisco, End user license agreement. URL http://www.cisco.com/c/en/us/products/end-user-license-agreement.html

[15] Juniper Networks, End user license agreement. URL http://www.juniper.net/support/eula/

[16] C. Aoun, E. Davies, Reasons to move the network address translator – protocol translator (NAT-PT) to historic status, IETF RFC 4966 (2007). doi:10.17487/RFC4966.

[17] G. Lencse, A. G. Soós, Design of a tiny multi-threaded DNS64 server, in: Proc. 38th International Conference on Telecommunications and Signal Processing (TSP 2015), Prague, Czech Republic, 2015, pp. 27–32. doi:10.1109/TSP.2015.7296218.

[18] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, X. Li, IPv6 addressing of IPv4/IPv6 translators, IETF RFC 6052 (2010). doi:10.17487/RFC6052.

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

[20] US-Cert, DNS amplification attacks, Alert TA13-088A (2016). URL https://www.us-cert.gov/ncas/alerts/TA13-088A

[21] Z. Ramzan, Phishing attacks and countermeasures, in: P. Stavroulakis, M. Stamp (Eds.), Handbook of Information and Communication Security, Springer, 2010, pp. 433–448. doi:10.1007/978-3-642-04117-4_23.

[22] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, DNS security introduction and requirements, IETF RFC 4033 (2005). doi:10.17487/RFC4033.

[23] T. Wilhelm, Professional Penetration Testing: Creating and Learning in a Hacking Lab, Elsevier, Waltham, MA, USA, 2013.

[24] C. Almond, Recursive client rate limiting in BIND 9.9.8, 9.10.3 and 9.11.0, ISC Knowledge Base, Reference Number: AA-01304 (2015).
URL https://kb.isc.org/article/AA-01304

[25] G. Lencse, S. Répás, Performance analysis and comparison of four DNS64 implementations under different free operating systems, Telecommunication Systems 63 (4) (2016) 557–577. doi:10.1007/s11235-016-0142-x.

[26] P. Mockapetris, Domain names – implementation and specification, IETF RFC 1035 (1987). doi:10.17487/RFC1035.

[27] Cert, Vulnerability note VU#23495: DNS implementations vulnerable to denial-of-service attacks via malformed DNS queries, VU #23495 (2001).
URL https://www.kb.cert.org/vuls/id/23495

[28] A. D. Keromytis, Buffer overflow attacks, in: H. C. A. van Tilborg, S. Jajodia (Eds.), Encyclopedia of Cryptography and Security, Springer, 2011, pp. 174–177. doi:10.1007/978-1-4419-5906-5_502.

[29] E. Levy, Smashing the stack for fun and profit, Phrack Magazine 7 (49) (1996) article 14.
URL http://phrack.org/issues/49/14.html#article

[30] S. Son, V. Shmatikov, The hitchhiker's guide to DNS cache poisoning, in: Proc. Security and Privacy in Communication Networks – 6th Iternational ICST Conference (SecureComm 2010), Singapore, 2010, pp. 466–483. doi:10.1007/978-3-642-16161-2_27.

[31] R. Elz, R. Bush, Clarifications to the DNS specification, IETF RFC 2181 (1997). doi:10.17487/RFC2181.

[32] J. Linkova, Let's talk about IPv6 DNS64 & DNSSEC, APNIC Blog (2016).
URL https://blog.apnic.net/2016/06/09/lets-talk-ipv6-dns64-dnssec/

[33] Internet Systems Consortium, BIND: Versatile, classic, complete name server software.
URL https://www.isc.org/downloads/bind

[34] M. Dunmore (Ed.), An IPv6 Deployment Guide, The 6NET Consortium, 2005.
URL http://www.6net.org/book/deployment-guide.pdf

[35] NLnet Labs, Unbound.
URL http://unbound.net

[36] Powerdns.com BV, Powerdns.
URL http://www.powerdns.com

[37] G. Lencse, M. Georgescu, Y. Kadobayashi, Benchmarking methodology for DNS64 servers, Computer Communications 109 (1) (2017) 162–175. doi:10.1016/j.comcom.2017.06.004.

[38] M. Georgescu, L. Pislaru, G. Lencse, Benchmarking methodology for IPv6 transition technologies, IETF RFC 8219 (2017). doi:10.17487/RFC8219.

[39] G. Lencse, D. Bakai, Design and implementation of a test program for benchmarking DNS64 servers, IEICE Transactions on Communication E100-B (6) (2017) 948–954. doi:10.1587/transcom.2016EBN0007.

[40] G. Lencse, Enabling dns64perf++ for benchmarking the caching performance of DNS64 servers, review version is available.
URL http://www.hit.bme.hu/~lencse/publications/

[41] G. Lencse, Y. Kadobayashi, Benchmarking DNS64 implementations: Theory and practice, review version will be available.
URL http://www.hit.bme.hu/~lencse/publications/

[42] G. Lencse, S. Répás, Improving the performance and security of the TOTD DNS64 implementation, Journal of Computer Science & Technology 14 (1) (2014) 9–15.
URL http://sedici.unlp.edu.ar/handle/10915/34537

[43] F. W. Dillema, DNS proxy and translator for IPv6 and IPv4, TOTD source code at GitHub.
URL https://github.com/fwdillema/totd/tree/1.5.3

[44] 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 2007 (ICICT'07), Dhaka, Bangladesh, 2007, pp. 315–318. doi:10.1109/ICICT.2007.375401.

[45] G. Lencse, S. Répás, Performance analysis and comparison of the TAYGA and of the PF NAT64 implementations, in: Proc. 36th International Conference on Telecommunications and Signal Processing (TSP 2013), Rome, Italy, 2013, pp. 71–76. doi:10.1109/TSP.2013.6613894.

[46] S. Répás, P. Farnadi, G. Lencse, Performance and stability analysis of free NAT64 implementations with different protocols, Acta Technica Jaurinensis 7 (4) (2014) 404–427. doi:10.14513/actatechjaur.v7.n4.340.

[47] G. Lencse, S. Répás, Performance analysis and comparison of different DNS64 implementations for Linux, OpenBSD and FreeBSD, in: Proc. IEEE 27th International Conference on Advanced Information Networking and Applications (AINA 2013), Barcelona, Catalonia, Spain, 2013, pp. 877–884. doi:10.1109/AINA.2013.80.

[48] P. Bálint, Test software design and implementation for benchmarking of stateless IPv4/IPv6 translation implementations, in: Proc. 40th International Conference on Telecommunications and Signal Processing (TSP 2017), Barcelona, Catalonia, Spain, 2017, pp. 74–78. doi:10.1109/TSP.2017.8075940.

[49] G. Lencse, Estimation of the port number consumption of web browsing, IEICE Transactions on Communications E98-B (8) (2015) 1580–1588. doi:10.1587/transcom.E98.B.1580.

[50] P. Savola, C. Patel, Security considerations for 6to4, IETF RFC 3964 (2004). doi:10.17487/RFC3964.

[51] P. Wu, Y. Cui, J. Wu, J. Liu, C. Metz, Transition from IPv4 to IPv6: A state-of-the-art survey, IEEE Communications Surveys and Tutorials 15 (3) (2013) 1407–1424. doi:10.1109/SURV.2012.110112.00200.

[52] M. Bagnulo, A. Garcia-Martinez, I. V. Beijnum, The NAT64/DNS64 tool suite for IPv6 transition, IEEE Communications Magazine 50 (7) (2012) 177–183. doi:10.1109/MCOM.2012.6231295.

[53] T. Savolainen, J. Korhonen, D. Wing, Discovery of the IPv6 prefix used for IPv6 address synthesis, IETF RFC 7050 (2013). doi:10.17487/RFC7050.

[54] R. Penno, T. Saxena, M. Boucadair, S. Sivakumar, Analysis of stateful 64 translation, IETF RFC 6889 (2013). doi:10.17487/RFC6889.

[55] K. Chittimaneni, M. Kaeo, E. Vyncky, Operational security considerations for IPv6 networks, IETF OPSEC WG Internet Draft (2017).
URL https://tools.ietf.org/html/draft-ietf-opsec-v6-12

[56] J. Korhonen, T. Savolainen, Analysis of solution proposals for hosts to learn NAT64 prefix, IETF RFC 7051 (2013). doi:10.17487/RFC7051.

[57] J.-Y. Bisiaux, DNS threats and mitigation strategies, Network Security 2014 (7) (2014) 5–9. doi:10.1016/S1353-4858(14)70068-6.

[58] C. Marrison, DNS as an attack vector – and how businesses can keep it secure, Network Security 2014 (6) (2014) 17–20. doi:10.1016/S1353-4858(14)70061-3.

[59] A. Chakrabarti, G. Manimaran, Internet infrastructure security: A taxonomy, IEEE Network 16 (6) (2002) 13–21. doi:10.1109/MNET.2002.1081761.

[60] R. Rasmussen, P. Vixie, Surveying the DNS threat landscape, Infoblox white paper (2016).
URL https://www.infoblox.com/wp-content/uploads/infoblox-white-paper-surveying-the-dns-threat-landscape.pdf

[61] D. Atkins, R. Austein, Threat analysis of the domain name system (DNS), IETF RFC 3833 (2004). doi:10.17487/RFC3833.

[62] K. J. O. Llanto, W. E. S. Yu, Performance of NAT64 versus NAT44 in the context of IPv6 migration, in: Proc. International Multiconference of Engineers and Computer Scientists 2012 (IMECS 2012), Hong Kong, Hongkong, 2012, pp. 638–645.
URL http://www.iaeng.org/publication/IMECS2012/IMECS2012_pp638-645.pdf

[63] C. P. Monte, M. I. Robles, G. Mercado, C. Taffernaberry, M. Or-

biscay, S. Tobar, R. Moralejo, S. Pérez, Implementation and evaluation of protocols translating methods for IPv4 to IPv6 transition, Journal of Computer Science & Technology 12 (2) (2012) 64–70.
URL http://sedici.unlp.edu.ar/handle/10915/19702

[64] S. Yu, B. E. Carpenter, Measuring IPv4 – IPv6 translation techniques, Tech. Rep. 2012-001, Dept. of Computer Science, Univ. of Auckland, Auckland, New Zeeland.
URL http://hdl.handle.net/2292/13586

[65] M. Mawatari, M. Kawashima, C. Byrne, 464XLAT: Combination of stateful and stateless translation, IETF RFC 6877 (2013). doi:10.17487/RFC6877.

[66] G. Lencse, Y. Kadobayashi, Methodology for DNS cache poisoning vulnerability analysis of DNS64 implementations, review version will be available.
URL http://www.hit.bme.hu/~lencse/publications/

[67] A. Hubert, R. van Mook, Measures for making DNS more resilient against forged answers, IETF RFC 5452 (2009). doi:10.17487/RFC5452.

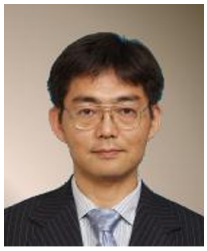
[68] G. Lencse, D. Bakai, Design, implementation and performance estimation of mtd64-ng a new tiny DNS64 proxy, Journal of Computing and Information Technology 25 (2) (2017) 91–102. doi:DOI:10.20532/cit.2017.1003419.

About authors

Gábor Lencse received 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 an Associate Professor. He has been working part time for the Department of Networked Systems and Services, Budapest University of Technology and Economics, Budapest, Hungary since 2005. Currently he is a Guest Researcher at the Laboratory for Cyber Resilience, Nara Institute of Science and Technology, Japan, where his research area is the security analysis of IPv6 transition technologies.

Dr. Lencse is a member of IEICE (Institute of Electronics, Information and Communication Engineers), Japan.



Youki Kadobayashi received his Ph.D. degree in computer science from Osaka University, Japan, in 1997.

He is currently a Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2013, he has also been working as the Rapporteur of ITU-T Q.4/17 for cybersecurity standardization. His research interests include cybersecurity, web security, and distributed systems.

Dr. Kadobayashi is a member of IEEE Communications society.