Teredo Security Modernization

Dmitry Viktorov, Narendrakumar Barvaliya, Syed Rizvi
Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT
{dviktoro, nbarvali, srizvi}@bridgeport.edu

***Abstract* – Teredo is a protocol which was developed by Microsoft in order to achieve compatibility between IPv4 and IPv6 protocols. IPv6 protocol is a highly-developed issue nowadays and it is supposed to be used all over the internet soon. Despite Teredo protocol fulfills its task, it has some security problems. The major problem is that while IPv6 packages are being transmitted, they contain information that may be used by hackers to attack computers behind NAT. It may cause many difficulties for system administrators and protection systems. This paper offers one modification of the protocol which hides the information about NAT system in packets.**

***Keywords* – Teredo security, NAT security, Information about NAT in Teredo, IPv6 over IPv4.**

I. Introduction

The analysis which is carried out in this paper is based on the Teredo specification [3] and [4]. The main purpose of the Teredo specification is to make the possibility for the use of NAT while sending IPv6 packets over IPv4 segments in the Internet.

The purpose described above is reached out but the specification makes some security problems for NAT gateways. The major one is that when connection between client behind NAT and server is set, client always must include the type of NAT in its packets. This information opens private issues of NAT and may be used for attacking. This document is intended to resolve this issue.

The described approach proposes additional concerns in order to make possible hiding the information about NAT. First, minimal changes of the protocol have to be made. Second, Teredo servers must support additional features for the transferring of packets.

II. Problem identification

First, the main problems of Teredo security were investigated in [6]. This investigation states that Teredo restores global addressability and routing to hosts using private IPv4 addresses. This is very beneficial to functionality but gives more opportunities for attacking. Reading the information about private IPv4 addresses, hackers may optimize ports scanning and try to get access to the computers behind NAT.

Second, the analysis which was carried out in [7] shows how some information in the packet may be used by hackers to detect the way of attacking. Partially, the decision recommended in [7], has already been implemented in [4]. However, the part which requires many changes has been ignored. In addition, the issue of the changes is incomplete because it doesn't precise how actually the proposed algorithms must work.

This paper offers the decision based on [7] but with the least changes for implementation and solves the topic which is raised in [7].

III. Related work

The analytical work [6] which is the background of this paper, investigates Teredo security implications. The paper [7] continues its investigation and offers a solution for one of the problems described there.

Before the description of [7], let's consider the Teredo address format described in [3]. As we see in the Fig. 1, the Teredo data packet contains the field of IPv6 header. According to [3], The IPv6 header contains the source and destination IPv6 addresses, at least one of which is a Teredo address. Understanding this, let's consider the Teredo address.

As it is presented in the Fig. 2, the address contains the field of flags. This field contains the type of NAT and is the object for analysis in [7].

As the paper [4] is fresher than [3] and extends it in many directions, let's examine the content of the flags field according to [4]. The content is presented in Fig. 3 and has definitions for all 16 bits which it contains. Let's look at the definitions:

- C: This bit should be set to 1 if the client is behind a cone NAT, to 0 otherwise;
- z: This bit is reserved. It MUST be set to zero when the address is constructed;
- Random1: MUST be set to a random value;

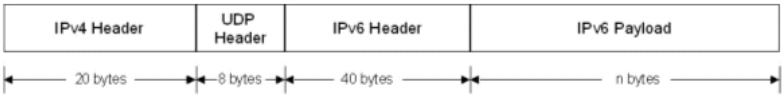

Fig. 1. The format of Teredo data packets.

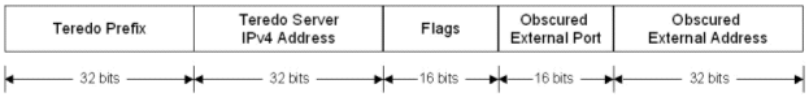

Fig. 2. Teredo address format.

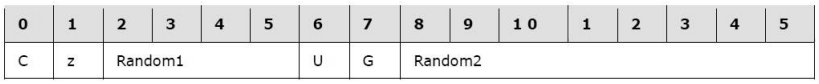

Fig. 3. The format of the Teredo flags field.

- U and G: These bits should be set to “00”, identifying a non global unicast identifier of an IPv6 address. The description of IPv6 address types maybe found in [1];
- Random2: MUST be set to a random value;

After the meaning of every bit of the flag is understood, let’s consider the proposal of [7]. Based on [6], the proposal says that the cone bit C is private information and may give hackers additional knowledge about the type of client’s NAT. Using this knowledge, hackers may apply special kinds of attacks without even spending time for finding out the type of client’s NAT. The information about the NAT systems is available in [8]. All four major implementations of NAT types are grouped in only two categories in Teredo realization: *cone NAT* and *all others*. Consequently, the cone NAT bit C represents only whether the NAT is cone or of any other type.

Generally, [7] suggests to deprecate the cone NAT bit and always set it to 0. In addition, it affects the qualification procedure described in section 5.2.1 of [3]. This suggestion is difficult in realization and still has not been realized by Microsoft.

IV. Proposed Solution

This paper proposes the solution based on [7] but with less alterations. In addition, this solution almost covers the problem of the cone bit.

It is easy to see that in order to deprecate the cone bit, we need to rewrite software for Teredo protocol almost completely. Therefore, this work saves the cone bit for the Teredo qualification procedure. This procedure works only once when the user establishes the initial connection with Teredo Server. Even though, if a hacker intercepts qualification packets at any time, it will not mean that he/she has gotten correct information.

After the qualification procedure, Teredo and/or Teredo Server should save the cone bit on their side and attach it to the established connection with a client. At this point, it must be noted that Teredo Server poses a stateless mechanism and the property of the cone bit doesn’t violate this issue.

When necessary, connections are established and the cone bit is saved, all procedures of [3] and [4] may function as they are supposed to do it.

Let’s consider the initial configuration for Teredo clients from [5] which is presented in Fig. 4. Exactly this configuration includes the mentioned qualification procedure. There is only one type of this configuration and, as a result, it sets the cone bit for the client.

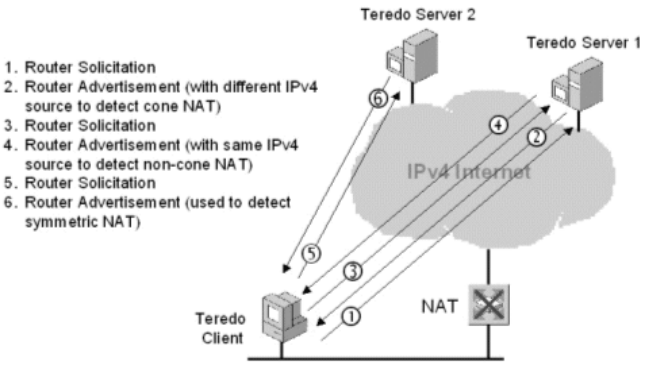

Fig. 4. Initial configuration for Teredo clients.

From [5], the initial configuration for Teredo clients consists of the following process:

1. A Router Solicitation (RS) message is sent from the Teredo Client to a preferred Teredo server (Teredo server 1). The Teredo Client sends the RS from a link-local address for which the Cone flag is set.
2. Teredo Server 1 responds with a Router Advertisement (RA) message. Because the RS had the Cone flag set, Teredo Server 1 sends the RA from an alternate IPv4 address. If the Teredo Client receives the RA, then it determines that it is behind a cone NAT.
3. If an RA is not received, the Teredo Client sends another RS from a link-local address for which the Cone flag is not set.
4. Teredo Server 1 responds with an RA. Because the RS had the Cone flag not set, Teredo Server 1 sends the RA from the source IPv4 address corresponding to the destination IPv4 address of the RS. If the Teredo Client receives the RA, then it determines that it is behind a restricted NAT.
5. To ensure that the Teredo Client is not behind a symmetric NAT, it sends another RA to a secondary Teredo server (Teredo Server 2).
6. Teredo Server 2 responds with an RA. The Teredo Client compares the mapped addresses and UDP ports in the Origin indicators of the RAs received by both Teredo servers. If they are different, then the NAT is mapping the same internal address and port number to different external addresses and port numbers. The Teredo Client determines that the NAT is a symmetric NAT and cannot use Teredo to communicate.

After the initial configuration, the Teredo Client knows its type of NAT. Therefore, it will use one of the two available algorithms for any Teredo communications. If NAT is not cone, then the cone bit is automatically set to zero. Accordingly, we will not consider this case in Teredo communications, as the problem is already solved.

From [5], on a periodic basis (by default every 30 seconds), Teredo clients send a single bubble packet to the Teredo server to maintain the NAT mapping. If the client doesn't do it, the mapping becomes stale and is removed. As the maintaining is obligatory, Teredo Server can save the cone bit as one of its variables for the connection and avoid requiring the client to send it. The scheme of the maintaining the NAT mapping is presented in Fig. 5.

The improvement which is offered in this paper, is activated between the procedures of Initial Configuration and Maintaining the NAT mapping. In other words, when the Teredo Server responds by the Router Advertisement message, it saves the cone bit on its side.

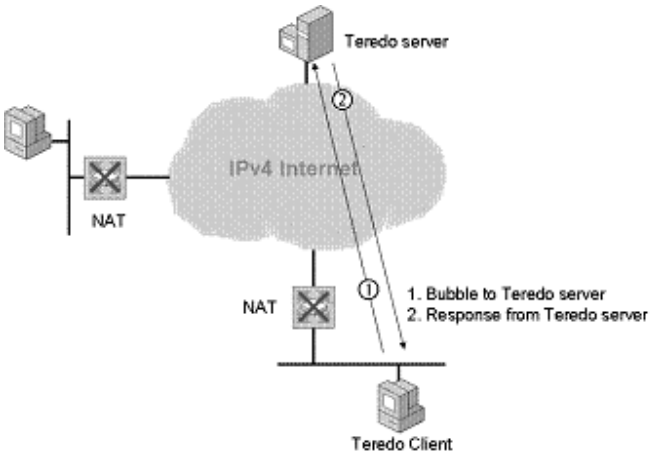

Fig. 5. Maintaining the NAT mapping.

Consequently, when the client is 'qualified', it starts to send bubble packets. Meanwhile, the server remembers the cone bit from its last RA message and applies appropriate algorithms for Teredo communications.

V. SYSTEM MODEL

The best way to present results of this work is to define the following model: there is a company which has K computers behind the cone NAT. Every computer is being turned off and on every M minutes where the time for turning off and on is N minutes. While a computer is working, it constantly sends and receives data. Within this time, a hacker tries to intercept packets every G minutes. The duration of hacker's attempts in average is H minutes. The work of model is simulated for T hours in respect to all other variables which are stated in minutes.

The best way to realize the model is a program which constructs the common model and the model explained in this paper. Every computer behind NAT in the program is a separate thread and a hacker is simulated by only one thread. When the program starts, it reads all execution parameters from the special configuration file. Manipulating the parameters in the model, it's easy to make conclusions about the efficiency of the proposal.

While the program was being developed in Java language, a high dispersion among the intercepted packets was revealed. The dispersion occurs because of the specific nature of threads in operating systems. To decrease the dispersion, the program runs itself 10 times and finds the average value.

VI. Simulation results

The simulation results are grouped in Table 1 and Table 2 for the original and proposed solutions respectively.

The results of the tables show that some parameters of the model are less significant than others, i.e. the change of some parameters doesn't affect results considerably. However, for the analysis in this paper all parameters are considered equal in their importance. The reason for this decision is that the significance of the parameters is different for the original and proposed models.

All variables in rows of both tables coincide for the convenient comparison. In order to make final conclusions, it is needed to compare all successful attempts for both cases. As the results depend on the speed of a computer, it is better to use them as the ratio of successful interceptions in the original model to the successful interceptions in the proposed model. Table 3 presents the results as ratios.

The minimal ratio in Table 3 is $0.3*10^{-3}$ and the maximum ratio is $4.947*10^{-3}$. To be closer to the real comparison, the ratio in average is $20.302/12 = 1.692*10^{-3} = 0.001692$.

TABLE 1
Simulation for the Original Model

K	M	N	G	H	T	Succeeded
1	30	7	60	10	24	34384
10	30	7	60	10	24	14494
10	10	7	60	10	24	18460
10	30	20	60	10	24	29988
10	30	7	30	10	24	13346
10	30	7	60	30	24	32172
10	30	7	60	10	48	30266
50	30	7	60	10	24	14650
100	30	7	30	10	24	21773
150	30	7	20	5	24	19815
200	10	7	20	5	24	11117
200	10	20	20	5	24	8159

TABLE 2
Simulation for the Proposed Model

K	M	N	G	H	T	Succeeded
1	30	7	60	10	24	12
10	30	7	60	10	24	5
10	10	7	60	10	24	8
10	30	20	60	10	24	9
10	30	7	30	10	24	19
10	30	7	60	30	24	25
10	30	7	60	10	48	35
50	30	7	60	10	24	55
100	30	7	30	10	24	76
150	30	7	20	5	24	49
200	10	7	20	5	24	55
200	10	20	20	5	24	7

TABLE 3
Ratios for the Results

No	Successful Attempts for the Original Model	Successful Attempts for the Proposed Model	Ratio, $*10^{-3}$
1	34384	12	0.349
2	14494	5	0.344
3	18460	8	0.433
4	29988	9	0.300
5	13346	19	1.423
6	32172	25	0.777
7	30266	35	1.156
8	14650	55	3.754
9	21773	76	3.490
10	19815	49	2.472
11	11117	55	4.947
12	8159	7	0.857

VII. Conclusion

From the simulation results it is evident that the proposed solution significantly decreases the probability to intercept the type of the cone NAT. It is fair to say that in average the proposed solution decreases the probability to intercept a packet with a cone bit in 591 times, in minimal case in 202 times and in maximum case in 3333 times. These results prove the idea of the proposal and make it well-grounded for the realization.

VIII. References

[1] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[2] B. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[3] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[4] Microsoft, "Teredo Extensions", MS-TERE, January 2009.

[5] Microsoft, "Teredo Overview", published: January, 2003, updated: January, 2007.

[6] J. Hoagland, "The Teredo Protocol: Tunneling Past Network Security and Other Security Implications", Symantec, November 2006.

[7] S. Krishnan and J. Hoagland, "Teredo Security Updates", Draft Update 2, February 2008.

[8] J. Rosenberg, J. Weinberger, C. Huitema and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC3489, March 2003.