2012

NAT Denial of Service: An Analysis of Translation Table Behavior on Multiple Platforms

Nathan Winemiller

Bruce Hartpence

Daryl Johnson

Sumita Mishra

NAT Denial of Service: An Analysis of Translation Table Behavior on Multiple Platforms

Nathan Winemiller, Bruce Hartpence, Daryl Johnson, and Sumita Mishra
Networking and Systems Administration Department
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, NY, USA

Abstract - *Network Address Translation or NAT, is a technology that is used to translate internal addresses to globally routable addresses on the internet. NAT continues to be used extensively in almost every network due to the current lack of IPv4 addresses. Despite being exceptionally commonplace, this networking technique is not without its weaknesses, and can be disabled with a fairly straightforward attack. By overpopulating the translation table, the primary mechanism used to translate the internal to external addresses, an attacker can effectively deny all internal users access to the external network. This paper takes an in-depth look at how five different vendors: Cisco, Extreme, Linksys, VMWare, and Vyatta, implement the translation table during active NAT sessions and how they are affected by TCP, UDP, and ICMP variations of the DOS attack.*

Keywords: Computer Network Security, Denial of Service, Network Address Translation, NAT

Track: Network Security Engineering

1 Introduction

Network Address Translation is a technology that is so widely deployed, it can be found in almost every home and in every company that has an enterprise network. With the scarcity of public IPv4 addresses, NAT with port translation has become a necessity when organizations need to provide Internet access to multiple users on the inside of their network. Devices that do the translations keep track of these connections in the NAT translation table. This table maps inside and outside ports and IP addresses so that internal hosts can have multiple concurrent conversations with the outside world. However, this creates a single point of failure in the network. If the translation table becomes too full or non-functional, the internal network could suffer connectivity issues when trying to reach the global network. This makes translation tables a prime candidate for denial of service attacks by a malicious user.

Therefore, it is important to understand how different vendors handle translation table behavior, especially in an enterprise environment which contains devices from multiple vendors. This paper will analyze how several different vendors handle NAT translation table size and how the devices react once the tables have been filled to capacity. Additionally, this paper will determine whether or not a small number of devices on the inside network can easily and effectively deny service to other users on the network by targeting the translation table on these NAT devices.

2 Background and Problem Statement

NAT is a networking function that is widely deployed in networks today. The IETF defines NAT as a "method by which IP address are mapped from one realm to another, in an attempt to provide transparent routing to hosts" [3]. NAT with Port Translation or NAPT is the most common and widely deployed version of NAT. The main function of NAPT is the conservation of global IP addresses by mapping a large number of private internal host addresses to a single external host address [1]. Essentially, NAPT provides a way for an intermediate device to map and translate internal IP addresses and ports to an external IP address on a different port.

One of the most important and complicated portions of NAPT has to deal with keeping track of the sessions that are in use. RFC 2663 identifies TCP and UDP sessions by keeping track of the:

- Source IP address
- Source TCP/UDP port
- Target IP address
- Target TCP/UDP port

ICMP tracking is similar except that the NAPT device keeps track of the ICMP query ID instead of TCP/UDP ports [3]. These properties create unique entries in the translation table and can be used to create a virtually limitless number of permutations for use in translation.

2.1 The Problem

In order to segment networks from the internet and preserve IP addresses, businesses and households around the world use NAT with Port Translation to map multiple internal hosts to a small number of globally routable public IPs [3]. As far as internal users are concerned, this process is supposed to be transparent, however if this process were to be disrupted, multiple if not all users on the internal network would be affected. The translation table which maps internal ports to external ports for active connections serves as a single point of failure and could be targeted in order to deny service to a large number of users.

Because every conversation that is going from the internal network to the external network has to be mapped and tracked, NAPT is an incredibly processor intensive task when compared to many other services a router could provide. In addition, every packet that goes through the NAT translation has to be rewritten; checksums need to be recalculated, along with many other changes [1]. Consequently, the more translations that occur, the more processor time is consumed to perform the NAT operations.

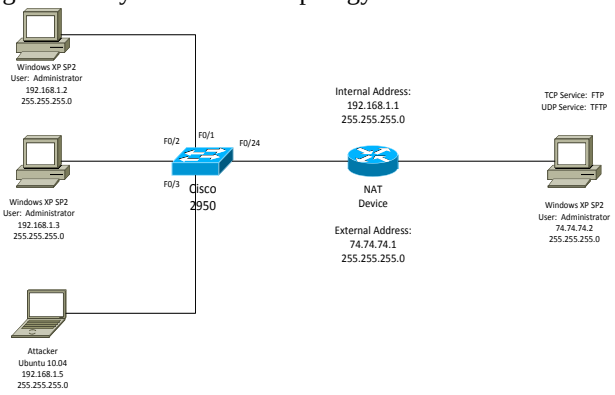
This can lead to performance degradation for all functions on the router if the NAT process consumes the available processing resources. As a result, if an attacker could create a situation where the NAT has so much work to do that it consumes all of the resources, they could cause failures not only for the NAT process, but also other functions that the router is supposed to handle (access lists, routing protocols, DHCP leases, etc.).

While there are many specifications on how NAT with Port Translation should operate, many of the implementation choices are not explicitly defined. Important details such as table entry timeouts for the different protocols, maximum translation table size, and entry tracking are left up to the vendors [3]. If the implementation discrepancies between vendors are significant enough, they could result in significant impact to network performance and application functionality. The next sections will provide scenarios that will demonstrate these issues and will provide data we can use to determine the differences in how each vendor implements NAPT.

3 Experimental Design

For all tests involving physical devices, the topology in figure 1 was constructed.

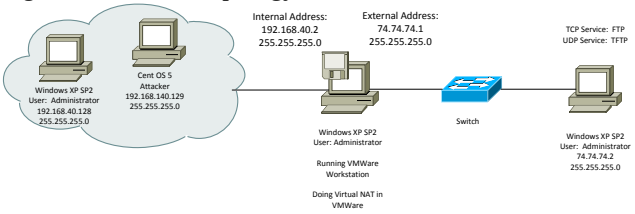
Figure 1: Physical Device Topology



In this diagram there are two networks, one internal and one external. There are two normal hosts on the internal network and an attacker that will launch an attack against the NAT device in the middle. The outside host is there to receive traffic and to provide an end point to determine if a loss of connectivity occurs. The main questions here are: can a single attacker device overwhelm the NAT device with translations thereby blocking legitimate traffic from exiting? What happens when the translation table on the NAT device becomes full?

For all tests involving the VMWare Virtual NAT Process, the topology in figure 2 was constructed.

Figure 2: Virtual Topology



Like the physical topology there is an internal and an external network that the attacker will attempt to disrupt.
Our hypothesis is that the attacker will be able to deny access to the external network by targeting the translation device. By creating a large number of translations, the attacker would be able to deny access to the outside by either filling the translation table or by consuming all of the available resources on the translating device.

3.1 Default Device Timers

Since the RFC documents regarding NAT only give recommendations on what the expiration timers should be for TCP, UDP, and ICMP entries, the individual vendors have their own default settings for each of these entries. These values are important because they determine the amount of time an attacker has to fill the table before the entries start expiring.

	TCP	Syn	Fin/RST	UDP	ICMP
Cisco	24	60	60 secs	5 mins	60

	hours	secs			secs
Extreme	2 mins	60 secs	60 secs	2 mins	3 secs
Linksys	N/A	N/A	N/A	N/A	N/A
Vyatta	12 hours	N/A	N/A	30 secs	30 secs
VMWare	N/A	N/A	N/A	30 secs	N/A

Notes:
1. TCP Default refers to all translated TCP ports that are not specified in the Cisco IOS (DNS and a few other services have their own special timeouts).

2. Timers on the Linksys device are not configurable by the user.

3. The vmnetnat.conf configuration file for the VMWare NAT process does not have a configurable field for TCP timeout. Additionally, there is no way to view the translation table for diagnostic testing to find out the default timeout.

4. Timers on the Vyatta device are not configurable by the user.

3.2 Attack Types

In order to determine the effects of the different protocols on multiple platforms, we had to create a number of scenarios. We tested attacks using TCP, UDP, and ICMP entries into the table. Each of these three tests were done against the five different vendors: Cisco, Extreme, Vyatta, Linksys, and VMWare.

If the attacker proves unsuccessful in filling the translation table for a device using the default timers, the experiments will be repeated for that device using unlimited timeout periods for each protocol. This will allow us to determine the effects of the attack on the device if it were to be successful.

3.3 Attack Scripts

The scripts used to execute the attacks were simple bash scripts that called several concurrent sessions of NMAP. These NMAP scans would attempt to reach various addresses on the outside on a variety of different ports. This allows the attacker to generate numerous translation table entries in a very short period of time. The scripts were very similar, with the main difference being in the –s options that changed the protocol being used by NMAP.

Table 1: TCP Attack Script

```
#!/bin/bash
for ((i=0; i<300; i++))
do
      sudo nmap -sS 74.74.74.3-254 1>/dev/null 2/dev/null &
done
```

Table 2: UDP Attack Script

```
#!/bin/bash
for ((i=0; i<300; i++))
do
      sudo nmap -sU 74.74.74.3-254 1>/dev/null 2/dev/null &
done
```

There was one alteration that had to be made during the UDP experiment. The Cisco device would not register the UDP entries in the translation table unless it received some sort of reply from the outside on those ports. Therefore, the UDP Attack Script in the Cisco scenario pointed to the outside host instead of a range of non-existent hosts.

Table 3: Cisco UDP Attack Script

```
#!/bin/bash
for ((i=0; i<300; i++))
do
      sudo nmap -sU 74.74.74.2 1>/dev/null 2/dev/null &
done
```

Table 4: ICMP Attack Script

```
#!/bin/bash
for ((i=0; i<300; i++))
do
      sudo nmap -sP 74.74.74.3-254 1>/dev/null 2/dev/null &
done
```

4 Methodology

The topology for each test was set up as shown in the figures above with three PCs connected to a switch behind a NAT device which then connected to an external host on the external network. The only exception was the VMWare test where the topology consisted of two VMs on the internal virtual network and one physical host on the external network. The external host was hosting an FTP server and a TFTP server which had access one large movie file to transfer. Only one NAT device from one vendor would be tested every time. Each vendor device would run through the test 5 times to ensure the validity of the data.

Before the attack began, all of the hosts issued pings to each other to verify connectivity. This also allowed for verification that the NAT translations were occurring correctly between the inside network and the external network. After initial connectivity and translation functionality were established, an internal host tested TCP and UDP functionality by using FTP and TFTP between the inside and outside. After full functionality was determined, a unidirectional JPerf test was run between an inside host and the outside host. This ensured that the NAT device could not

be brought down by a full load of traffic from a single device. After all of the testing had finished, commands would be issued to the NAT device to clear the translation table and reset the NAT statistics when applicable.

After the verification process, the attacker would kick off the script and a timer (on a stopwatch) would be started. Each attack period would last for 5 minutes and then the NMAP processes would be killed. Every minute during the attack, internal hosts would attempt to ping the internal gateway, the external gateway, and the outside host. Additionally, every minute, the internal hosts would attempt to establish an FTP and a TFTP session and attempt to transfer a file. Also, every minute the relevant commands would be issued to the NAT device (when applicable) to determine the number of translations and to verify that the correct translations were being put into the translation table.

After the attack had completed, internal hosts would then attempt to ping each other, the internal gateway, the external gateway, and the external host. Additionally, internal hosts would then attempt to establish FTP and TFTP sessions with the external hosts in order to transfer the movie file. This process would be repeated until all attempts were successful or until 15 minutes elapsed.

These tests would be run initially using the default timers on the devices. However if the attacker could not fill the translation table for a particular device with any of the attacks, then tests would be run on that specific device using unlimited timers. This would allow us to observe the consequences of filling the translation table on that device.

4.1 Problems Encountered

During the course of our experiments, we did run into a few issues during the testing. Unfortunately some of the devices had very little or no visibility into what was happening during the translation process. The Linksys device was the worst for this since it did not have any relevant means of logging NAT Translations. This problem was exacerbated by the fact that all management for the Linksys device was in-band, which became unreachable during the attack. The VMWare NAT process also had no means to debug the translations while they were occurring. This meant that all we had to go on were the results of the experiments to determine what was actually going on during the attack.

Another issue was the processing limitations of the devices used to execute the attacks. In most cases it was a netbook that had very little processing power, which limited the number of translations we could do within a certain period of time. In the other cases it was a virtual machine with very limited resources (in the VMWare tests). We originally wrote the scripts to loop 1000 times to maximize the number of IMAP instances running, but ultimately had to scale it down to 300 concurrent sessions to keep the attacking machines from locking up.

4.2 Metrics

In order to determine whether connectivity was affected or not, every minute of the attack we would run: ICMP echo requests to the outside, a TFTP transfer to the outside, and an FTP transfer to the outside. We would also look at the current size of the translation table along with the CPU utilization on the device in question. The final thing we would watch for was whether or not the packets the outside device was receiving were being translated (by looking at the source address of the traffic).

5 Results and Discussion

The results of this experiment show that a small number of compromised nodes with the right software can severely impact enterprise grade equipment in a short period of time. While almost all platforms tested were negatively impacted by the TCP and UDP attacks, the effectiveness of certain protocols for filling the translation tables varied from vendor to vendor. Additionally, filling the translation tables also had varied effects on the devices performing the NAT translations depending on what vendor created the device.

On the Cisco device, the translations filled the available memory on the device which not only prevented new entries from being entered into the table (and therefore being translated) it also caused memory allocation errors which negatively impacted other processes that the router was performing. The Cisco device fared better against the TCP attacks due to its session tracking capability and the aggressive timeouts on the SYN packets. The UDP attack on the other hand was able to fill the table on this device and deny translation capabilities to all other devices on the network.

The Extreme device on the other hand, limited the number of NAT translations allowed on one address instead of letting it consume all of the other memory. This allowed for smaller number of total entries, but preserved the integrity of the device when the table was filled. The Extreme device fared very well against both TCP and UDP attacks due to its very aggressive default timers. However, these timers are a double edged sword and by defying best practices and standards laid down in the RFCs, could cause problems on a network with higher latency times. For example, in the case of the Extreme device, the two minute default TCP timer could cause certain TCP applications to fail because it expires so quickly. TCP logical connections can last for a very long time even if connectivity is temporarily disrupted or there isn't any traffic to send for a period of time. By expiring these TCP entries, the internal host will have to establish a new session to the outside and most likely start the conversation over.

The Linksys device by far fared the worst of all the devices that were tested. It was unable to handle the processing load of translating such a large amount of requests and therefore became completely unreachable during the attack. TCP and

UDP attacks were equally effective against this device. Unfortunately, the lack of accessibility during the attack (in-band management only) and limited logging made any more insight into the operation of this device all but impossible.

While the VMWare device fared well against the UDP attack (unlike the Cisco device) it was severely impacted by the TCP attack. Service was denied during the attack and unlike the rest of the devices tested, TCP and UDP connections could no longer be established even after the attack had ended. The only way to restore full connectivity was to restart the VMWare NAT service. This would pose a significant problem to network administrators since outside connectivity would appear to be restored (ICMP echo requests would work), but any service related applications would cease to function. Unfortunately, the VMWare NAT process has limited configuration and debug capabilities so determining the exact cause of this disruption is also next to impossible.

The Vyatta device fared extremely well during these tests. These attacks didn't appear to have any impact despite having several thousand translations entered into the table. Due to the resources at my disposal, the only device to test this platform was markedly more powerful than the other devices tested in this series of experiments. Interestingly enough, Vyatta also defies RFC recommendations by not allowing for the administrator to adjust any of the timers that are used in the translation process. Additionally, while I was unable to do so in my experiments, the default TCP timer is extremely long and the Vyatta device doesn't appear to do any session tracking (unlike the Extreme or Cisco devices). This could possibly be exploited on a device that is less powerful or being actively utilized on an enterprise network.

It is interesting to note however, that unlike when the CAM (Content Addressable Memory) table, which holds the MAC address / port entries, becomes full on a switch, the NAT device does not "fail open" and pass all traffic. When the translation table became full (in all cases where this was possible), no new translations could occur and the NAT devices would drop any new traffic not already in the table. Another point to note is that the NAT devices did not FIFO any translation entries. While expired entries were FIFO'd out, if the table became full, the entries remained in the table until they expired. Therefore, new entries could not be added until the old entries expired (which could be a significant amount of time).

Another interesting conclusion that can be drawn from these experiments is that the differing implementations of NAT between the vendors (especially in the case of timers) could cause interoperability issues if a connection has to traverse a NAT device on the source end and the destination end. While TCP connections are generally given a duly long amount of time before expiration, UDP connection timeouts varied greatly between the vendors. With VMWare and Vyatta allowing only 30 seconds before the UDP entry times out, significant application issues could occur over high latency networks that span long distances.

The relatively small number of compromised nodes required to execute the attack paired with the fact that NAT generally causes a single point of failure for a large number of users makes this a relatively easy and effective type of denial of service attack to execute. Furthermore, while it may be difficult to fill up a translation table on a device with a significant amount of memory, the effects of the attack on the processing capabilities of the NAT devices makes this attack scalable to almost any network size. Therefore, we can conclude that this attack is viable in an enterprise setting and against several mainstream vendor devices that are currently deployed.

5.1 Mitigation

The main ways that the NAT denial of service attack takes advantage of the NAT device is to either fill up the translation table to prevent new entries or to create so many entries that the processor on the device cannot keep up. Therefore the most obvious way to mitigate the effects of this attack is to limit the number of translations that hosts are allowed to make. Making a rule that limits the number of overall translations allowed is common, and while this may solve the issue of high CPU utilization, this technique makes it easier for the attacker to fill the table and deny service to legitimate users service. Because of this, limiting the number of translations must be applicable to individual hosts.

Cisco has features in place in their IOS to accomplish this goal relatively well. Most platforms support rate limiting of some kind, but the Cisco platform supports rate limiting by either the total number of translations, the number of translations that can be created by a list (that includes a number of hosts), and the number of translations a specific host can make [6]. Obviously limiting the total number of translations would prove counter-productive, and while limiting the number of translations a specified host can use is fine on a smaller network, this mitigation technique cannot scale to a large network.

The Extreme device on the other hand supports what the vendor calls "Auto-constraining," which limits the amount of ports a single internal host can use at one time [7]. This works by evenly distributing the amount of port space between each internal user evenly [7]. While this is easy to configure and can be effective in smaller networks, this feature could end up preventing legitimate users from making new connections in a scenario where there are a large number of internal users and a very small number of external addresses to map to.

Unfortunately, the Linksys, VMWare, and Vyatta platforms don't appear to support any of these types of features at all which makes stopping this type of attack much more difficult when using these platforms.

Another feature that is built into most NAT platforms is the ability to control the amount of time that passes before an entry in the table expires. This is best evidenced by the

table on the Extreme device that couldn't be filled in this experiment using the normal timers. Unfortunately the Vyatta and Linksys platforms do not allow the administrator to configure these timers, which makes them less able to prevent the NAT DOS attack. However, administrators need to be careful when setting these timers since they could time out legitimate sessions prematurely and cause applications to malfunction, especially in higher latency networks with a high traffic volume.

6 Conclusions

The purpose of this paper is to closely examine infrastructure security risks when using NAT with Port Translation. Additionally, this paper analyzes the default behavior of NAPT devices from different vendors. The results of these experiments show that a single device on the internal network is capable of overwhelming the translation table. These experiments also show that the default behavior when implementing NAPT varies significantly between vendors. Default timers, entry behavior, and configurable settings are all vendor specific. We believe that the almost universal use of NAPT justifies further investigation and standardization of its use. The reality is that most consumers utilize NAPT and do not consider that the very mechanisms that allow it to work can be taken advantage of to the detriment of all internal hosts. As the various experiments outlined in the paper show, many vendors use default behaviors that leave their devices open to exploitation when implementing NAPT. Furthermore, some vendors do not even offer the ability to change crucial settings that could be used to mitigate this type of attack.

7 References

[1] Smith, M.; Hunt, R.; , "Network security using NAT and NAPT," Networks, 2002. ICON 2002. 10th IEEE International Conference on , vol., no., pp. 355- 360, 2002 doi: 10.1109/ICON.2002.1033337. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1033337&isnumber=22194

[2] Hartpence, B.; Johnson, D.;, "A Re-examination of Network Address Translation Security," RIT Network Security and System Administration. SAM 2010.

[3] Srisuresh, P, & Holdrege, M. (1999, August). Rfc: 2663 ip network address translator terminology and considerations. Retrieved from http://tools.ietf.org/html/rfc2663

[4] Hain, T. (2000, November). Rfc 2993: architectural implications of nat. Retrieved from http://www.ietf.org/rfc/rfc2993.txt

[5] Srisuresh, P, & Egevang, K. (2001, January). RFC 3022: traditional nat. Retrieved from http://www.ietf.org/rfc/rfc3022.txt

[6] Cisco Systems Incorporated. (2003). Rate Limiting NAT Translation. Retrieved from http://www.cisco.com/en/US/docs/ios/12_3t/12_3t4/feature/guide/gt_natrl.html

[7] Extreme Networks Incorporated. (2003, June). Summit 200 Series Switch Installation and User Guide. Retrieved from http://www.extremenetworks.com

[8] Lebovitz, G.; Thaler, D.; Zhang, L. (2010, July). RFC 5902: IAB Thoughts on IPv6 Network Address Translation. Retrieved from http://tools.ietf.org/html/rfc5902#section-1