2011

NAT denial of service: An Analysis of translation table behavior on multiple platforms

Nathan Winemiller

NAT Denial of Service

An Analysis of Translation Table Behavior on Multiple Platforms

Nathan Winemiller

nsw7619@rit.edu

6/20/2011

Bruce Hartpence

Sumita Mishra

Daryl Johnson

Presented in accordance with the requirements for a Master's of Science in Networking and Systems Administration from the B. Thomas Galisano College of Computing and Information Sciences

Rochester Institute of Technology

Rochester, NY

Table of Contents

Abstract

Network Address Translation or NAT, is a technology that is used to translate internal addresses to globally routable addresses on the internet. It is used extensively in almost every network requiring global connectivity due to the current lack of IPv4 addresses. The primary mechanism used to facilitate the translation of internal addresses to external addresses and vice versa is the translation table. This study takes an in-depth look at how five different vendors: Cisco, Extreme, Linksys, VMWare, and Vyatta, implement the translation table during active NAT sessions. Additionally, this study analyzes the methodology required to fill a translation table and the Denial of Service that is a result of the attack. We consider the relative difficulty of accomplishing this task between the different platforms and protocols (TCP vs UDP vs ICMP). We conclude this study with steps that can be taken to prevent or mitigate the NAT DOS attack.

1. Introduction

1.1: Background

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.

With the large address space provided by IPv6 and its rapid adoption by ISPs and companies around the world, many would argue that NAT is quickly becoming obsolete. Unfortunately, this is not quite the case. The majority of businesses and households currently on the internet (especially in the United States) currently run IPv4 networks using NAT, and getting all of these entities to migrate to IPv6 is not a quick or easy process. Even when the migration has fully occurred, there will still be a need for NAT for legacy devices and applications that do not support IPv6. Additionally, there are some issues present in IPv6 with address renumbering and multi-homing issues that may lead to NAT being adopted in the IPv6 world [8]. Consequently, NAT (especially NAT with Port Translation) will remain a widely adopted and relevant technology for years to come.

Additionally, many users rely on NAT as a type of barrier between their internal network and the internet. The inherent workings of the NAT process can mask the internal addresses of the internal nodes, however those nodes can still be accessed from the outside (NAT was never designed for security purposes). Another danger is that if an internal host establishes a connection to a malicious outside host, the attacker can use the knowledge of the outside IP address and port to push potentially dangerous packets back through the NAT device to the internal host.

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. Lastly, this paper will examine whether or not an attacker can gather enough information from the outside of the network to push malicious packets through the translation device into the internal network.

1.2 Importance

Network Address Translation with Port Translation is a widely deployed technology. 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 were this process 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.

While there are many specifications on how NAT with Port Translation should operate, many of the implementation choices 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. Therefore, examination and standardization of behavior could lead to better compatibility between different vendors as well as clarification on the best practices involved when using NAT with PT.

In addition, making direct connections between internal, controlled equipment to the internet is inherently risky [1]. Despite the fact that users separated from the outside by NAT, when they establish connections to the outside, the details in their packets give outside users all of the

information they need to push packets back through the NAT. An attacker could leverage this to send back unexpected packets that could adversely affect the network.

2. Literature Review

2.1 NAT with Port Translation Operation

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]. Despite the fact that NAT with Port Translation (NAPT) sees the most widespread deployment in modern networks by far, there are actually a few other common types of NAT that could also be deployed: Static NAT and Dynamic NAT. Static NAT involves mapping one internal host address to one external host address [3]. In this instance, all ports are shared and only one entry has to be kept in the table since all connections are directly mapped between the external address and the internal host. Because of the scarcity of external IP addresses Static NAT sees very little use in current networks. Dynamic NAT is similar in operation to Static NAT except that a number of internal hosts can be mapped to an equal number of external host IPs on a rotating basis. Once an internal host is mapped to an external entry, it acts like Static NAT in that the ports in use are mapped one to one between the internal and external IPs. In this case the NAT does have to keep track of session data in order to determine when an IP address is no longer being used and can be freed for use by other internal hosts [3]. Much like Static NAT, Dynamic NAT is not widely deployed anymore because there are generally many more internal hosts that need access to the internet than there are available global addresses for use.

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. By doing so, this not only allows administrative domains to conserve resources by only buying one external IP address, but also allows administrators to segment the administrative domain from the rest of the internet. This allows for an isolation of privately controlled networks in both a security and administrative sense [1].

The basic operation of NAPT can be found in RFC 3022. When an internal host sends a packet destined for an external host, the NAT device takes the internal device and source port and inserts it into a translation table [5]. Then, the device translates the internal address to the globally unique outside address and picks an arbitrary port that will stand in for that specific connection [5]. Additionally, the destination address and port are mapped to the same entry in the table thereby completing the mapping process.

NAPT supports TCP and UDP sessions along with most types of ICMP messages (with the exception of redirect) [5]. Since ICMP uses identifiers instead of ports, the NAT device must map the identifiers to another set of identifiers along with the global address [5]. Despite the large number of modifications to the packets during NAPT, it is designed to be transparent to the end users.

2.2 Session Tracking with NAPT

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.

One of the problems involved with session tracking is determining when the sessions begin and when they end. If sessions aren't identified quickly enough, they won't fully establish and if they aren't determined to be finished within a timely manner, superfluous entries can cause unnecessary delay and processing overhead during the translation process. TCP session initiation is fairly simple to detect by looking for a packet containing a SYN flag and no ACK flags [3]. Unfortunately, UDP is a connectionless protocol and cannot be tracked as easily. RFC 2663 states that any non-TCP session can be identified by assuming that the first packet that does not meet any currently existing session parameters can be considered a new session [3].

Session completion is a completely different matter altogether and is more complicated to deal with than session initiation. A TCP session ends when both sides acknowledge the FIN packets that they receive or when a RST packet is sent. However, this is based on the assumption that the FIN or RST packet is delivered to the destination [3]. If the packet is lost or corrupted in transit, and the NAT device automatically deletes the session information out of the translation table, an ungraceful termination of the session would invariably occur. Therefore, a TCP session should be assumed to be terminated after a minimum of 4 minutes after the detection of a FIN or RST packet [3]. This number is based on the Time-Wait duration of twice the Maximum Segment Lifetime [3]. Unfortunately, much like UDP session initiation, there isn't a way to detect if a UDP session has completed by looking at the UDP datagram. Therefore the only way to determine if a UDP session has completed is to base it on the time since the last packet for that

session was sent or received. RFC 2663 states that a 24 hours is a viable assumption to terminate TCP sessions and that non-TCP sessions can be terminated after a few minutes of idle time [3]. However, these assumptions may not be acceptable since every TCP and UDP application functions in a unique matter. Therefore, the authors of the RFC state that the session timeout parameters should be configurable by the NAPT device in use [3]. Having mismatching session timeouts, or timeouts that are too short could adversely impact applications that are running on the network. If two hosts are talking to each other through a NAT device and the translation times out due to a spike in latency during the conversation, this would interrupt the overall session the two hosts were having (leading to failures in the application).

2.3 NAPT Performance Issues

Despite the large amount of detail given to creating the entries, the RFCs provide little detail as to how the translation table should be handled. RFC 3022 states that entries should be added as they are established [5]. However, the maximum number of entries a translation table should contain is not discussed. Furthermore, little discussion is given to what would occur should the translation table not be able to support further entries. RFC 2663 states that should the need arrive, the oldest sessions (based on timestamp) present in the table could be aged out in order to make room for new session data [3]. This is not the default behavior however. Therefore, the decisions regarding the maximum number of entries in the translation table along with how new entries are added to the table after the table has been filled are left up to the vendor and can vary. Generally, many companies rely on a few devices to translate many hosts to very few globally routable IP addresses. Because of this, the translation table becomes a single point of failure and an attacker could deny service to a large number of hosts if they were able to disrupt the normal operation of the translation table. When compared to a traditional distributed denial of service attack that requires thousands of compromised hosts, this type of attack could be extremely efficient since a large number of entries can be created by each compromised host on the internal network.

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.).

2.4 NAPT Security Through Obscurity

One of the advantages of NAPT is that because multiple connections from internal hosts need to be mapped to a smaller number of external global IPs, NAT devices will block inbound connections to both TCP and UDP ports unless they are statically mapped [1]. This "one way" behavior is a byproduct of how NAT works. Since the internal devices are being mapped to an outside address, routes to the internal network(s) will not exist (especially in cases where the internal addresses are private) which prevents direct communication between the internal and external networks [4]. Therefore NAT acts as a firewall of sorts, since in most cases external hosts cannot reach internal hosts unless the internal hosts initiates the communication. This basic function of NAT can prevent many unwanted packets from getting through to the inside network. However, NAT alone cannot act as a firewall. As Hartpence and Johnson showed in their research, outside hosts can bypass the NAT and directly communicate with internal hosts if they have a route to the inside [2]. This doesn't exploit the translation table, but it does bypass it. However, bypassing the NAT translation table requires the attacker to have a route to the internal network. This is a fairly unlikely option for an attacker trying to connect over the internet. However, threats on the internal network may be able to get through a NAT device into a sensitive portion of the network without getting translated. Therefore, NAT cannot be the solely relied on to provide security. It should be used along with access lists or some type of firewalling. Because of this insecurity, many small business and consumer NAT devices also integrate a simple firewall into the device along with their other functions.

However, Hartpence and Johnson discuss the possibility of peer to peer networking opening connections in the translation table that could be used to exploit internal hosts [2]. Since the NAT device looks for data streams that match the 4 session characteristics listed above, an attacker with sufficient knowledge of the translation entries, could theoretically craft packets to traverse the NAT translation. The packets sent out by peer to peer software like bittorrent are an ideal way for an attacker to gain access to the network. They contain all four pieces of information necessary for an attacker to send packets back in past the NAT device. However the impact of this type of exploit would depend on the end application and operating system of the internal host.

2.5 IPv6 and the Future of NAT

IPv6 is a technology that has been in development for years and is finally seeing wide scale deployments. It offers many improvements over IPv4, but the most significant change involves a significantly larger address space. IPv6 offers 128 bits of address space as opposed to the limited 32 bit space offered in IPv4. Therefore, every device should theoretically be able to receive a globally routable address when IPv6 is fully implemented. This feature eliminates the main reason that an entity would implement NAT on the edge of their network (IPv4 address

scarcity). While this is true, there are still several issues that will require the implementation of NAT for an extended period of time.

The first and most obvious reason NAT will continue to be used is backward compatibility. During the migration to IPv6, NAPT will still be important for networks that are still transitioning. Additionally, even after the majority of a network is IPv6, there will still be legacy devices and applications that will need to remain IPv4 until they can be replaced with newer products. For large corporations that use a lot of custom applications, this could turn out to be a very lengthy process indeed. Legacy hardware is an especially big problem for ISPs who have to find a way to get rid of NAT and implement IPv6 without interrupting to service to everyday people. New hardware will have to be distributed to or purchased by home users (old IPv4 modems / routers, etc.) which will most certainly require a significant amount of time.

In addition to the migration issues that will require NAT to remain implemented, there are a few problems with IPv6 that could lead to the implementation of NAT in the future RFC 5902 discusses several problems, with NAT as a possible solution.

The first main issue in IPv6 is address renumbering for small / medium sized businesses [8]. Today if an organization decides to change providers, relatively few changes need to be made to make the transition because only the globally routable addresses are subject to change. If the organization is very large, the odds are they have provider independent (global) addresses which means that re-addressing does not need to occur at all [8]. However when IPv6 becomes prevalent, small / medium sized organizations which cannot afford to register / buy a provider independent block of addresses will be tied to their provider addresses. Therefore if this kind of an organization wanted to switch providers, they would have to readdress their entire internal network (which is a significant setback) [8]. Therefore, NAT could be used in this case to prevent renumbering in small and medium sized organizations.

Another large issue in IPv6 is the idea of multihoming a network between multiple providers [8]. This allows for an organization to have redundancy in case one provider network becomes unusable. If global addressing becomes the norm, organizations will have to inject both their own routes as well as their provider routes into both provider networks. This effectively nullifies provider-based route aggregation [8]. Essentially this means that the global routing table will become significantly larger with each multihomed site that is added to the internet. NAT has been proposed as a solution to this issue as well.

3. Experiment

3.1 Purpose

The purpose of this thesis was to create a large number of TCP, UDP, and ICMP translations to determine their impact on the translation table. This was done to observe the methods that different vendors use to handle large translation table sizes and how they handle a situation where the translation table can no longer create and store new entries.

3.2 Experimental Objectives

The experiments in this thesis were created to determine:

1) The maximum translation table size for a NAT device.
2) Whether the maximum translation table size is consistent between vendors.
3) Factors that determine the maximum translation table size.
4) Whether or not a single internal host can fill the translation table in a set period of time.
5) The consequences of filling the translation table.
6) Consequences of creating a large number of translations.
7) Ways to effectively prevent malicious users from creating large numbers of translations.
8) Different consequences of filling the translation table using TCP, UDP, or ICMP traffic.

3.3 Topology Design

These experiments consisted of a fairly simple topology involving two normal internal hosts, a malicious host that would be executing the NAT DOS attack, a switch with one VLAN, a router doing NAT or a virtual NAT in the case of VMWare, and an external host. Three main tests would be executed: TCP, UDP, and ICMP attacks. Each of these three attacks would be tested against a NAT device from each of the following vendors: Cisco, Extreme, Linksys, Vyatta, and VMWare.
The following topology design was consistent for all of the experiments except for the VMWare which can be seen in the second diagram.

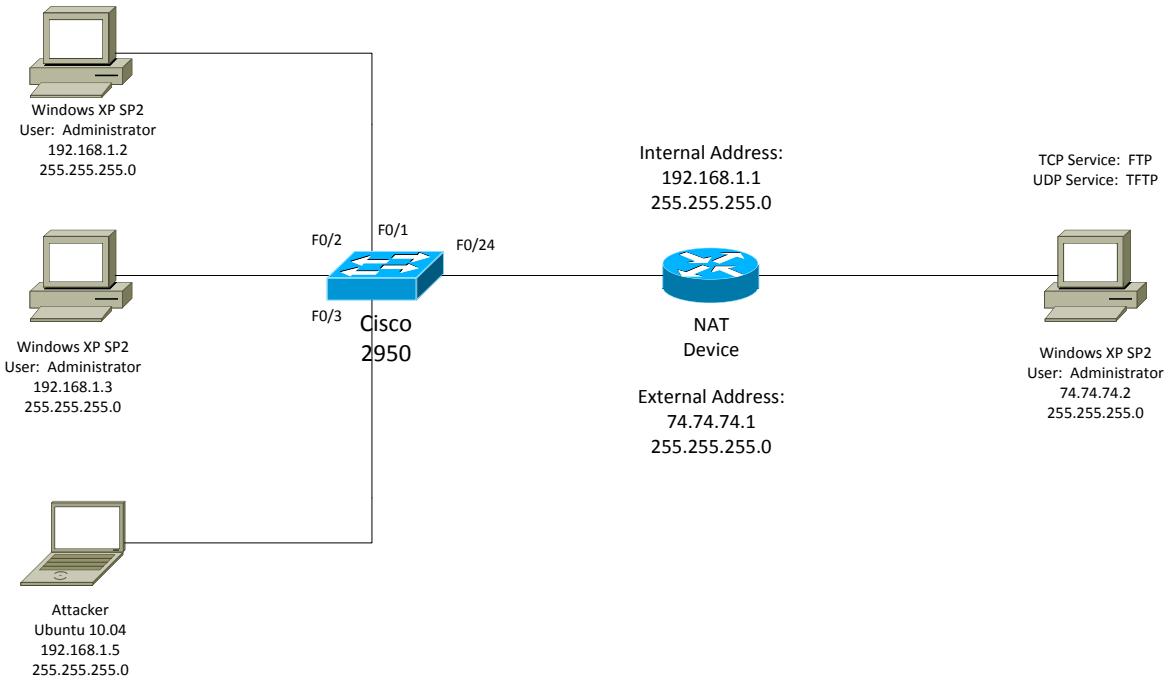


Figure 1: Topology for Cisco, Extreme, Vyatta, and Linksys Tests

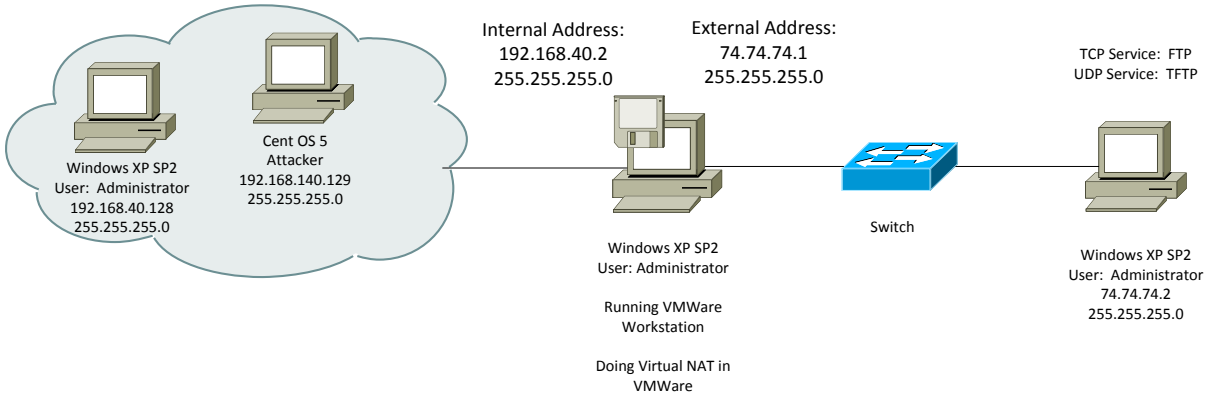


Figure 2: Topology for VMWare Tests

Note: Due to the fact that VMWare virtual NAT service was being tested in the second topology, one of the internal VMs was used to execute the NAT DOS attack instead of another physical host.

3.4: 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.

3.4.1 Cisco 2651

Default TCP Timeout: 24 hours (non-port-specific)

Default Syn Timeout (syn with no ack response): 60 seconds

Default fin/rst Timeout: 60 seconds

Default UDP Timeout: 5 minutes (non-port-specific)

Default ICMP Timeout: 60 seconds

* Non-port-specific applies to all translated ports that are not specifically specified in the Cisco IOS (DNS and a few other services have their own special timeouts) *

3.4.2 Extreme Summit 200

Default TCP Timeout: 2 minutes

Default Syn Timeout (syn with no ack response): 60 seconds

Default fin/rst Timeout: 60 seconds

Default UDP Timeout: 2 minutes

Default ICMP Timeout: 3 seconds

3.4.3 Linksys WRT120N Wired/Wireless Router/AP

Default TCP Timeout: N/A

Default UDP Timeout: N/A

Default ICMP Timeout: N/A

Not available in documentation and the limited firmware doesn't allow for diagnostic testing to find out translation table settings

Note: Timers on the Linksys device are not configurable by the user.

3.4.4 VMWare Workstation

Default TCP Timeout: N/A

Default UDP Timeout: 30 seconds

Default ICMP Timeout: N/A

*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. *

3.4.5 Vyatta 5.0.2

Default TCP Timeout: 12 hours

Default UDP Timeout: 30 seconds

Default ICMP Timeout: 30 seconds

*Default timeouts for Vyatta were determined by looking at the entries after creation and observing their time until expiration. *

Note: Timers on the Vyatta device are not configurable by the user.

3.5 TCP NAT Attack Methodology

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
```

The TCP attack script is a bash script that initiates 300 instances of NMAP to run. The flag -sS means that it is going to send out TCP packets with the syn flag set to the range of IP address from 74.74.74.3 to 74.74.74.254. Since NMAP is used to scan for ports, it will be sending out syn packets to every port for every IP address in the range.

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. All tests in this portion of the experiment utilized the default timers on all devices.

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.

3.6 TCP NAT Attack Observations

3.6.1 Cisco 2651 Results

After five minutes of the attack the maximum achievable entries in the translation table totaled on average 58,000 concurrent entries. This was mainly because with no acknowledgements of the SYN packets being sent out, the default timeout of these entries was 60 seconds. Therefore, this attack was unable to fill the translation table under default timer settings. Despite being unable to fill the translation table, the Cisco device was negatively impacted during the attack. CPU usage spiked to nearly maximum during the attack as can be seen in Figure 3.

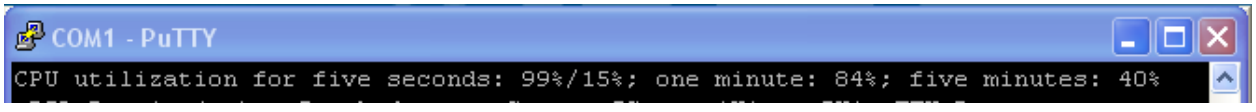


Figure 3: Cisco CPU utilization 2 minutes into the attack

During this attack, the connectivity between the inside and outside hosts became intermittent with a high rate of ICMP failure between the internal and external hosts. However, connectivity between the internal hosts (excluding the gateway) remained unaffected. Additionally, FTP and TFTP services became unavailable during the attack. Figure 4 shows that because of the short timeout, most of the entries during the experiment were expiring, leading to the low number of entries in the table after five minutes.

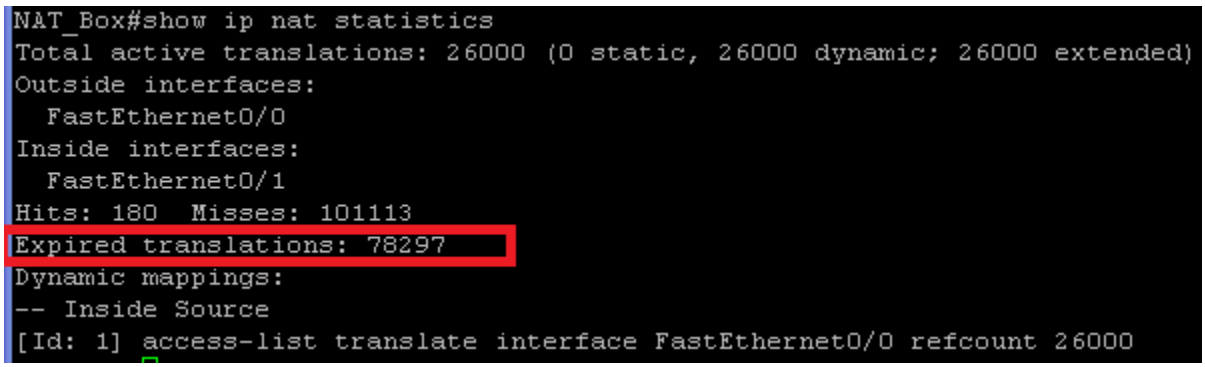


Figure 4: NAT statistics 3 minutes into the attack

After the attack stopped, the entries in the translation table continued to expire until none were left. Once the attack was no longer occurring, internal hosts regained full connectivity to the external host and FTP and TFTP sessions were able to occur.

3.6.2 Extreme Summit 200 Results

After 5 minutes of running the TCP attack, the average maximum number of entries achievable totaled around 46,000. Because there were no acknowledgements of the TCP SYN packets that were being sent out, the corresponding entries would expire in 60 seconds. Due to the fast rate of expiration, the attack was unable to fill the translation table using TCP. Figure 5 shows an example of the translation table size during the attack and that despite adding over 70,000 entries, only 44,000 or so could stay active due to the default timers. However, much like the Cisco attack, the CPU utilization became extremely high leading to a large number of dropped packets making connectivity to the outside intermittent at best. Hosts were able to make a few packets through but 50% or more were usually dropped as can be seen in Figure 6.

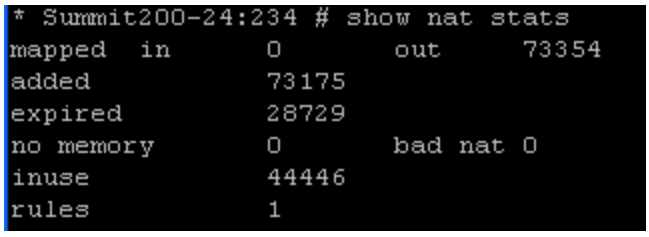


Figure 5: Extreme NAT Statistics after 4 minutes

```
C:\Documents and Settings\Administrator>ping 74.74.74.2

Pinging 74.74.74.2 with 32 bytes of data:

Reply from 74.74.74.2: bytes=32 time=229ms TTL=127
Reply from 74.74.74.2: bytes=32 time=226ms TTL=127
Request timed out.
Request timed out.

Ping statistics for 74.74.74.2:
    Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
Approximate round trip times in milli-seconds:
    Minimum = 226ms, Maximum = 229ms, Average = 227ms
```

Figure 6: Higher than normal latency for ICMP requests and large amounts of dropped packets.

Additionally, all attempts to make an FTP connection or to transfer files using TFTP between the inside and outside networks failed during the attack. Internal connectivity (excluding the gateway) was unaffected during the attack. Despite the large disruption during the attack, after the attack subsided full connectivity was reestablished. FTP and TFTP services could be utilized within a few seconds of the attack being over and since the default timers were so short, the entries in the table expired to nothing after another few minutes.

3.6.3 Linksys WRT120N Wired/Wireless Router/AP Results

Due to the limited features of the Linksys device, it was next to impossible to discover anything about the default timers or how many entries could be held in the table at a time. The only available data that could be gathered from the device was the outgoing translation log, part of which can be seen in Figure 7. Unfortunately, even though the entries that are added into the table can be seen, expirations are not listed in the log and the log is only a certain size so only a limited amount of connections can be seen at a time.

```
192.168.1.5 to 74.74.74.199:443 is accepted
192.168.1.5 to 74.74.74.235:443 is accepted
192.168.1.5 to 74.74.74.165:WWW is accepted
192.168.1.5 to 74.74.74.164:WWW is accepted
192.168.1.5 to 74.74.74.178:WWW is accepted
192.168.1.5 to 74.74.74.156:WWW is accepted
192.168.1.5 to 74.74.74.119:WWW is accepted
192.168.1.5 to 74.74.74.98:WWW is accepted
192.168.1.5 to 74.74.74.181:WWW is accepted
192.168.1.5 to 74.74.74.28:WWW is accepted
192.168.1.5 to 74.74.74.230:WWW is accepted
```

Figure 7: Portion of the Linksys outgoing translation log

Despite the fact that minimal data could be gathered from the device from the attack, the effect of the attack on the Linksys device is immediate and devastating. Within seconds of the attack beginning, the external network and the Linksys device became unreachable. All attempts to query the external network using ICMP fail along with any attempts to establish an FTP

connection or TFTP transfer. Since the configuration for the Linksys device is done in-band, no configuration changes could be done, nor could any activity statistics be pulled during the time of the attack. Generally the gateway proved to be unreachable but occasionally partial connectivity could be established to it as can be seen in Figure 8.

Figure 8: Partial but ineffectual connectivity to the gateway during the attack

Even though the gateway is almost completely unreachable during the attack, the outside device continues to see the attacker traffic but not traffic from any other host. This raises the possibility of a queuing or full buffer issue as the cause of the outage. However, the Jperf test results were consuming 95Mbps of bandwidth during testing before the attack and network connectivity remained unaffected. This makes the possibility of a queuing issue less likely. It is more likely that the processing unit of the device was negatively affected during by the attack, thereby causing the network outage.

Although the attack is devastating to network service during the attack, full network connectivity is regained just as quickly after the attack ceases. Full ICMP connectivity along with functional FTP and TFTP transfers are available within seconds of the attack subsiding. This also makes the possibility that the network outage was caused by a congestion of the buffers less likely.

3.6.4 VMWare Workstation Results

VMWare is a platform that is widely utilized for the purpose of virtualizing machines. When using VMWare, one physical host can have multiple virtual hosts running independent operating systems and doing different tasks concurrently. One of the ways VMWare uses to facilitate network connectivity to the virtual hosts is through its own VMWare NAT process. When using this, the physical host has a globally routable address that is on the normal network and the virtual hosts have internal IPs and use the physical host address in the traditional NAPT fashion.

Strangely, while the timer settings for UDP appear in the vmnetnat.conf file that controls the virtual NAT settings, TCP timer settings appear to be nonexistent. In addition, due to the fact that the NAT process is virtualized, no output or statistics commands can be issued to determine the state of the process during the attack. When the attack begins, the VMWare NAT process appears to be completely unaffected. For the first three minutes, ICMP requests are answered, FTP transfers are possible and TFTP transfers are possible as can be seen in Figure 9.

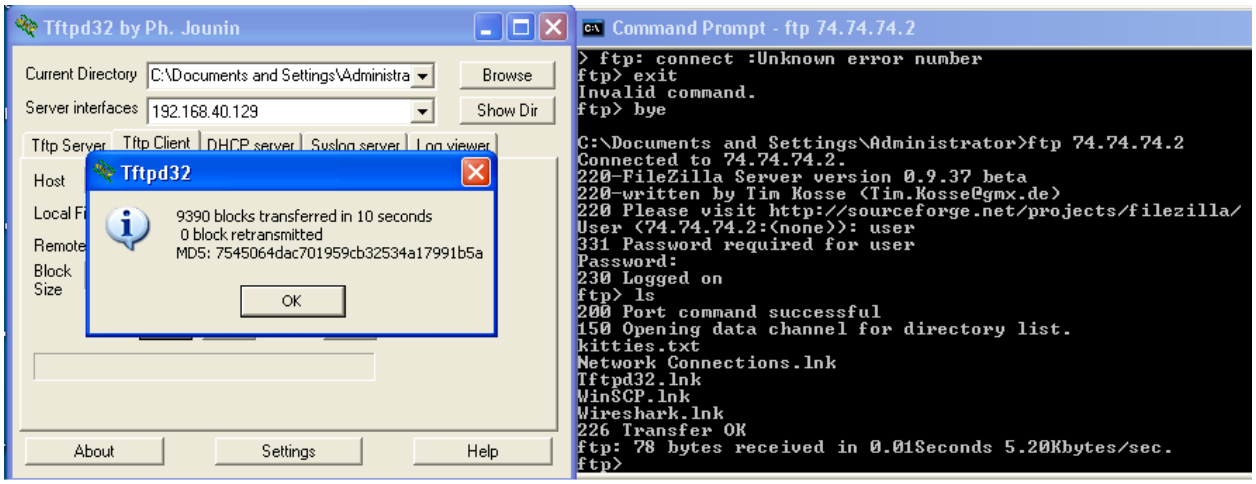


Figure 9: FTP and TFTP Attempts Successful 3 minutes into the attack

However, after about 4 minutes the NAT translation process appears to fail. ICMP, FTP, and TFTP attempts from the inside to the outside fail. In addition, after this point in time, the outside host fails to see any packets from the virtualized machines (even the attacker). On the other hand, inside hosts are still able to communicate with each other, including the virtual gateway. After the attack ends, ICMP connectivity is restored after about 2 minutes.

The TCP attack on the VMWare process results in strange circumstances after the attack abates. Even though ICMP connectivity is regained between the inside and the outside, the inside hosts are unable to use FTP or TFTP (or any other TCP or UDP service) between the inside and outside as can be seen in figures 10 and 11. This outage lasts for a significant amount of time. Connectivity using FTP and TFTP did not reestablish even after 15 minutes of inactivity after the attack.

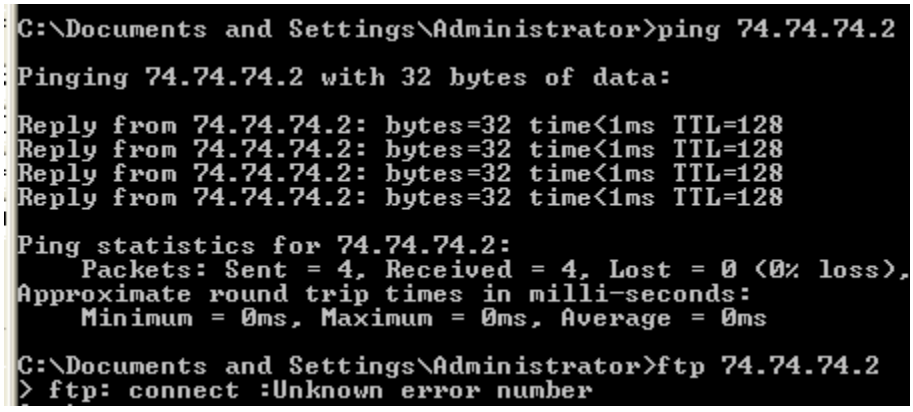


Figure 10: Internal host unable to establish FTP connection after the attack

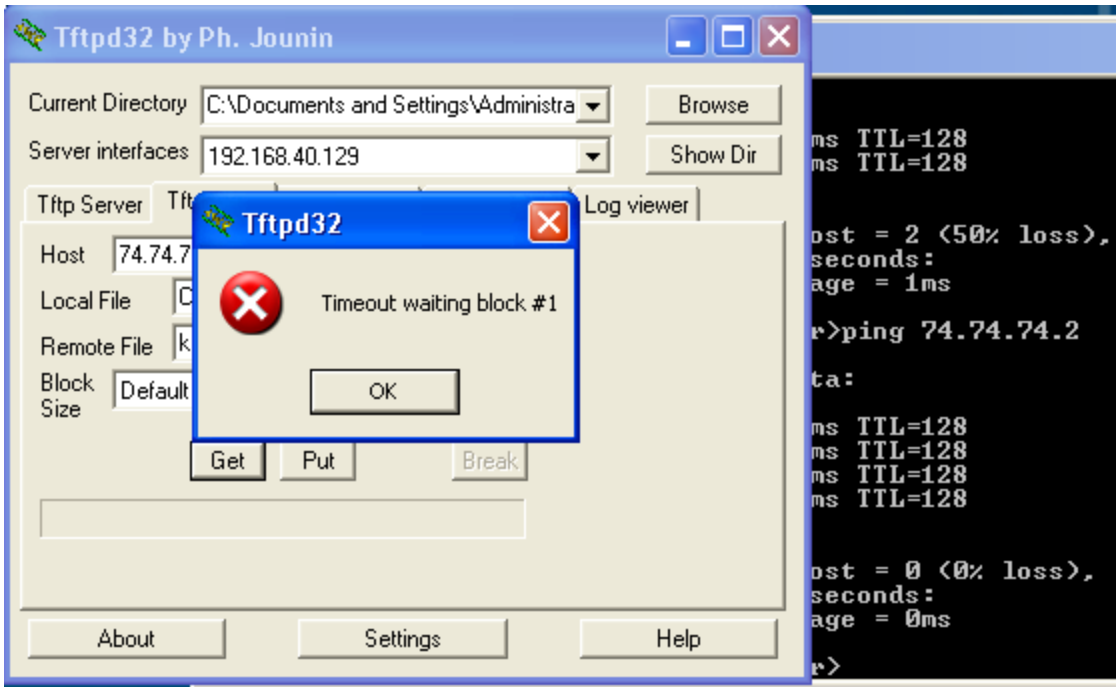


Figure 11: Internal host unable to establish TFTP transfer after the attack

The only way to regain TCP / UDP connectivity between the inside and outside after this attack was executed was to restart the VMWare NAT service which can be seen in Figure 12. Only after the service was reset, could full connectivity (ICMP, FTP, and TFTP) be achieved.

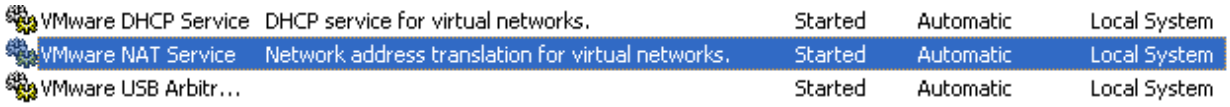


Figure 12: VMware NAT Service

3.6.5 Vyatta 5.0.2 Results

Vyatta is a network OS that can be used on a multi-NIC PC to perform routing and other IP service functions. It has a Linux base and there are free versions and enterprise versions that require a paid license. The command structure is very similar to that of Juniper devices, and despite the fact that the open source version of the software used in these tests is free, it still provides a high level of functionality.

Unfortunately despite the high level of functionality available with Vyatta, there doesn't appear to be a way to change or view the default configuration of the TCP or UDP timers. Therefore, all tests done to the Vyatta device were done using default timers. The Vyatta device seemed completely unaffected by the TCP translations during and after the attack. TCP entries appeared to last a very long time (upwards of 12 hours) and because of this several hundred thousand entries were assumed to be translated. Unfortunately none of the show commands regarding

NAT would work after the 5 minute period. This can be attributed to the extremely large number of translations present on the device.

This number can be deceptive however. The Vyatta show commands do not show how many entries are active in the table currently, only how many have matched the rule to be translated. But since TCP entries seem to last for a very long time, it can be assumed that in this case there are in fact hundreds of thousands of entries in the table. The lack of effect from the attack can most likely be attributed to the relative power of the machine. This device was originally a desktop computer and therefore, has a large amount of memory and processing power when compared to the other devices in this test. The only negative effect the attack appeared to have had was that the *show nat translations* command would lock up due to the large number of entries in the table.

Note: Show commands for the NAT translations did not produce any output for this test

Machine Comparison

Platform	**RAM**	**CPU Speed**
Cisco 2651	64MB	Unavailable
Extreme Summit 200	128MB	Unavailable
Linksys WRT120N	32MB	400MHz
VMWare NAT	N/A (Process)	N/A (Process)
Vyatta	512MB	1.6GHz

Figure 13: Machine Specification Comparison

3.7 UDP NAT Attack Methodology

```
#!/bin/bash

For ((i=0; i<300; i++))

Do

    Sudo nmap -sU 74.74.74.3-254 1>/dev/null 2/dev/null &

Done
```

The UDP attack script is a bash script that initiates 300 instances of NMAP to run. The flag -sU means that it is going to send out UDP packets to the range of IP address from 74.74.74.3 to 74.74.74.254. Since NMAP is used to scan for ports, it will be sending out UDP packets to every port for every IP address in the range.

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. All tests in this portion of the experiment utilized the default timers on all devices.

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.

3.8 UDP Attack Observations

3.8.1 Cisco 2651 Results

The first few times I ran the test script, the entries ended up showing up in the NAT translation table as ICMP entries, and therefore would expire after 60 seconds. In order to get around this issue, I ran the following script instead.

```
#!/bin/bash
```

```
For ((i=0; i<300; i++))

Do

    Sudo nmap -sU 74.74.74.2 1>/dev/null 2/dev/null &

Done
```

During my experiments I found that the Cisco platform would not register UDP packets as UDP unless they went to a host that existed on the outside network. This script is identical to the script run on all of the other platforms except that it had one host as a target to scan on the outside as opposed to a range. The rate of translations entered remained similar to the other tests and the translation table filled with the expected entries while using this script.

After a little more than three minutes (3:20 on average) after the UDP attack was started, the table on the 2651 became full and unable to accept more entries. The maximum number of entries in the table for this device ended up being around 117,668. This resulted in massive amounts of memory errors as can be seen in Figure 14. Also, other functionality of the router was affected while the memory was full of translation entries. The auto-complete and question mark functions no longer worked and the router process for routing protocols could not be invoked either as can be seen in Figure 15.

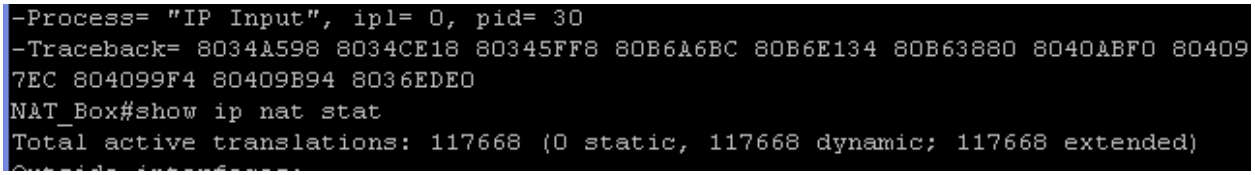


Figure 14: Memory Traceback errors due to full translation table

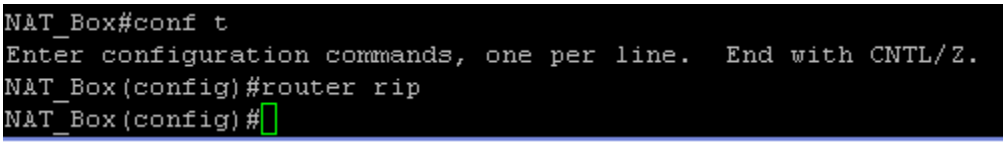


Figure 15: Normal services on the router could not be invoked

During the attack, the CPU utilization spiked to between 80 and 100% and internal hosts were unable to reach the outside or the gateway as can be seen in Figure 16. However, internal network connectivity (aside from the gateway) remained unaffected. Because of the loss of connectivity, FTP and TFTP transfers failed during the attack.

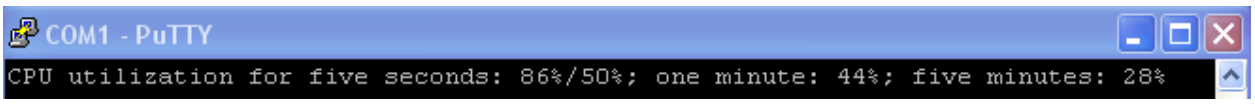


Figure 16: High CPU usage during the attack

After the active part of the attack stopped, the table remained full for a short period (until the active entries started to expire) and during this time FTP and TFTP services remained unavailable. After the entries started to expire, new entries could be added to the table and full connectivity was regained.

3.8.2 Extreme Summit 200 Results

After 5 minutes of running the UDP attack against the Extreme device only about 44,000 active entries were able to be maintained. This is due to the aggressive two minute UDP default timer that the Extreme device had in place. Despite the timers however, the table did get fairly sizable as can be seen in Figure 17.

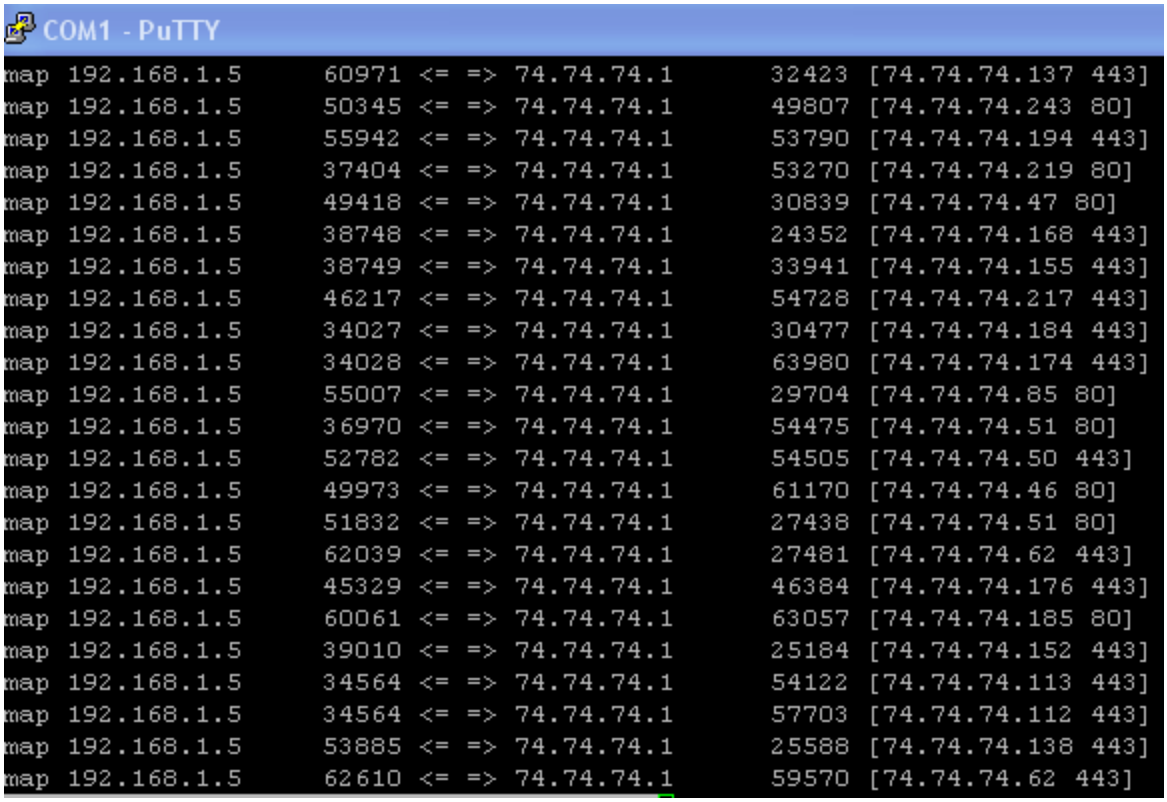


Figure 17: Extreme translation table (partial) after 5 minutes

Much like the Cisco test and the TCP tests before it, CPU utilization spiked during the attack. Also during the attack, inside hosts lost connectivity to the outside host. TCP and TFTP transfers could not be established during the attack either. After the attack completed, full connectivity was restored. At this point neither TCP or UDP attacks were able to fill the translation table on the Extreme, however the CPU utilization during the attack occurs in both.

3.8.3 Linksys WRT120N Wired/Wireless Router/AP Results

Much like during the TCP attack, the UDP attack has immediate and catastrophic results when the Linksys device is targeted. All connectivity between the internal hosts and the external host is lost as soon as the attack begins. Additionally, FTP and TFTP transfers do not work for the duration of the attack. Unfortunately, the firmware doesn't provide very much information as to the NAT process on the router. There aren't any commands to show active entries or to keep count of the total translations. The only results that could be gleaned came from the translation logs which are of a limited size. Therefore, total entries cannot be determined from the number of entries in the table. A portion of the UDP log can be found in Figure 18.

```
192.168.1.5 to 129.21.3.17:DOMAIN is accepted
192.168.1.5 to 74.74.74.2:34892 is accepted
192.168.1.5 to 64.132.49.139:WWW is accepted
192.168.1.5 to 74.74.74.2:49207 is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
192.168.1.3 to 10.200.200.17:402 is accepted
192.168.1.5 to 74.74.74.2:539 is accepted
192.168.1.5 to 129.21.3.17:DOMAIN is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
192.168.1.5 to 74.74.74.2:37843 is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
192.168.1.5 to 129.21.3.17:DOMAIN is accepted
192.168.1.5 to 74.74.74.2:1457 is accepted
192.168.1.5 to 129.21.3.17:DOMAIN is accepted
192.168.1.5 to 74.74.74.2:21358 is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
192.168.1.5 to 129.21.3.17:DOMAIN is accepted
192.168.1.5 to 74.74.74.2:54114 is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
192.168.1.5 to 74.74.74.2:25931 is accepted
192.168.1.5 to 74.74.74.2:20525 is accepted
192.168.1.5 to 74.74.74.2:443 is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
192.168.1.5 to 129.21.3.17:DOMAIN is accepted
192.168.1.5 to 129.21.4.18:DOMAIN is accepted
```

Figure 18: Portion of the UDP log from the Linksys device

The problem of retrieving information on this platform is exacerbated by the fact that all management is done in-band and cannot be accessed during the execution of the attack.

After the attack has completed, full connectivity is restored after about 10 seconds. Full Jperf tests were done before the attack phase like during the TCP segment. Connectivity was not affected during the TCP or UDP jperf tests and no buffer problems were encountered. These preliminary results make overutilization and buffer issues due to the UDP NAT attack unlikely.

3.8.4 VMWare Workstation Results

Unlike the TCP attack which had devastating effects on the VMWare NAT process, the UDP attack had little noticeable effect. The latency increased slightly during the attack but all of the devices retained connectivity to the outside and the FTP and TFTP transfers were successful every time they were attempted. This can largely be attributed to the default 30 second UDP timer specified in the vmnetnat.conf file as can be seen in Figure 19.

```
[udp]
# Timeout in seconds, 0 = no timeout, default = 30; real value might
# be up to 100% longer
timeout = 30
```

Figure 19: Configurable UDP portion of the vmnetnat.conf file.

Because the UDP timer was set to 30 seconds (unlike TCP which didn't have a timer setting) we can conclude that the entries that the false entries that were being added were expiring every thirty seconds. This prevented the VMWare NAT process from being overloaded with translations and hanging up. The VMWare NAT process faired significantly better in this test than the previous TCP test. Therefore, we can conclude that having the ability to configure the TCP timer could have helped minimize the effects of this attack.

3.8.5 Vyatta 5.0.2

Like the TCP settings, Vyatta doesn't appear to have any way to configure the UDP settings, but UDP entries appear to expire after 30 seconds by default. During the attack, there was no loss of connectivity and FTP and TFTP transfers did not fail during the test period. After 5 minutes around 268,000 entries appeared to match the NAT rule as can be seen in Figure 20. Unfortunately there is no way to tell how many of those entries were active at the end of the testing period. Since the expiration timer is so low the number of active entries in the table was most likely a significantly smaller number. The lack of effectiveness of this attack on this platform can largely be attributed to the relatively powerful hardware that this platform was hosted on, but also to the short UDP expirations that are the default on this platform. The only difference in results between the TCP and UDP attacks was that the NAT commands would not work during the TCP attack.

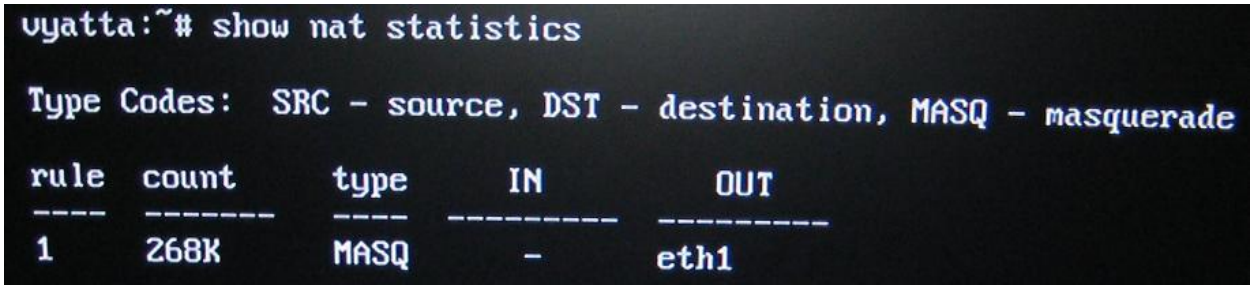


Figure 20: Entries that matched the NAT rule after 5 minutes

3.9 ICMP NAT Attack Methodology

```
#!/bin/bash

For ((i=0; i<300; i++))

Do

        Sudo nmap -sP 74.74.74.3-254 1>/dev/null 2/dev/null &

Done
```

The ICMP attack script is a bash script that initiates 300 instances of NMAP to run. The flag -sP means that it is going to send out ICMP echo requests to the range of IP address from 74.74.74.3 to 74.74.74.254. Unlike the TCP and UDP tests, the ICMP tests only sends echo requests instead of scanning for ports.

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. All tests in this portion of the experiment utilized the default timers on all devices.

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.

3.9 ICMP Attack Observations

3.9.1 Cisco 2651 Results

Similar to the TCP SYN settings, ICMP translations expire in 30 seconds by default. After 5 minutes the number of active translations averaged around 45,000 entries. However, this attack did not seem to affect the router as badly as the TCP or UDP attacks. Figure 21 shows that while the CPU utilization is high (and does occasionally spike very high) during the attack, it is not nearly as high on average as the previous two attacks.

Figure 21: CPU usage on the 2651 5 minutes into the ICMP attack.

As previously discussed, unlike TCP or UDP, there are no ports to translate. Therefore, the NAT device uses the ICMP identifiers to map internal requests to the external address. This can be seen in action in Figure 22.

```
NAT_Box#show ip nat trans
Pro  Inside global      Inside local       Outside local      Outside global
icmp 74.74.74.1:7081    192.168.1.5:7081   74.74.74.25:7081   74.74.74.25:7081
icmp 74.74.74.1:51120   192.168.1.5:51120  74.74.74.9:51120   74.74.74.9:51120
icmp 74.74.74.1:10164   192.168.1.5:10164  74.74.74.3:10164   74.74.74.3:10164
icmp 74.74.74.1:18353   192.168.1.5:18353  74.74.74.9:18353   74.74.74.9:18353
icmp 74.74.74.1:43929   192.168.1.5:43929  74.74.74.57:43929  74.74.74.57:43929
```

Figure 22: ICMP Translations on the Cisco 2651

Unlike the TCP and UDP attacks, there was very little negative impact on the device during this attack. None of the devices lost connectivity and FTP and TFTP services remained unaffected during this attack. Additionally, latency increases were minimal as can be seen in Figure 23.

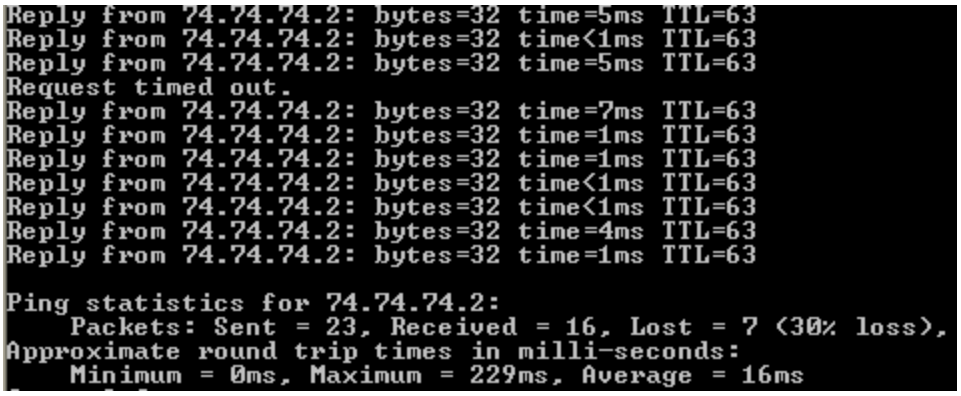


Figure 23: Latency test 3 minutes into the attack

Despite the fact that ICMP does have a checksum that needs to be recalculated and rewritten just like TCP and UDP, this attack didn't appear to have any substantial negative effects on the Cisco device. However, CPU utilization did go up enough that problems could have occurred had the router had more load on it from other processes.

3.9.2 Extreme Summit 200 Results

By default, the NAPT configuration on the Extreme did not translate ICMP to the outside address like the other platforms. Therefore another rule had to be added to match all ICMP traffic and translate it to the external IP address. This portion of the configuration can be seen in Figure 24. The first line caused both TCP and UDP to be translated and the second line creates a dynamic mapping between the internal hosts and the outside address for ICMP packets.

```
configure nat add "n_outside" map source any to 74.74.74.1/32  both portmap
configure nat add "n_outside" map source any to 74.74.74.1/32
```

Figure 24: Second dynamic entry added to the process in order to translate ICMP requests

Even though there was a second rule added and the NAT device still had to keep track of / rewrite the ICMP packets, there appeared to be little to no impact on the device during the ICMP attack. After 5 minutes there were less than 10,000 translations active in the table. This was due to the extremely low 3 second ICMP timeout that was the default on the device. Additionally, CPU utilization appeared to be relatively low when compared to the TCP and UDP attack, hovering around 40%. There was no loss of connectivity during the attack, FTP and TFTP services remained available during and after the attack, and there didn't appear to be any negative effects on the device during the ICMP attack.

Although the 3 second timer was effective in limiting the number of ICMP entries in the table, the low amount of time makes it more likely that translations will expire before the device can receive a response. This could make troubleshooting in high latency / low available bandwidth situations difficult since the translations can expire before ICMP packets can complete the round trip journey.

Since none of the three attacks were able to fill the translation table with the default timers, another set of experiments was designed and executed with unlimited timers in order to determine the effect of a full translation table on the Extreme device. The methodology and results of these experiments will be discussed in a later section.

3.9.3 Linksys WRT120N Wired/Wireless Router/AP Results

The results of the ICMP test on the Linksys platform exactly mirror the results of the TCP and UDP tests. Once the attack begins, connectivity between the inside and outside is disrupted and remains disrupted for the duration of the attack. Figure 25 shows connectivity beginning to fail at the start of the attack and completely failing shortly thereafter.

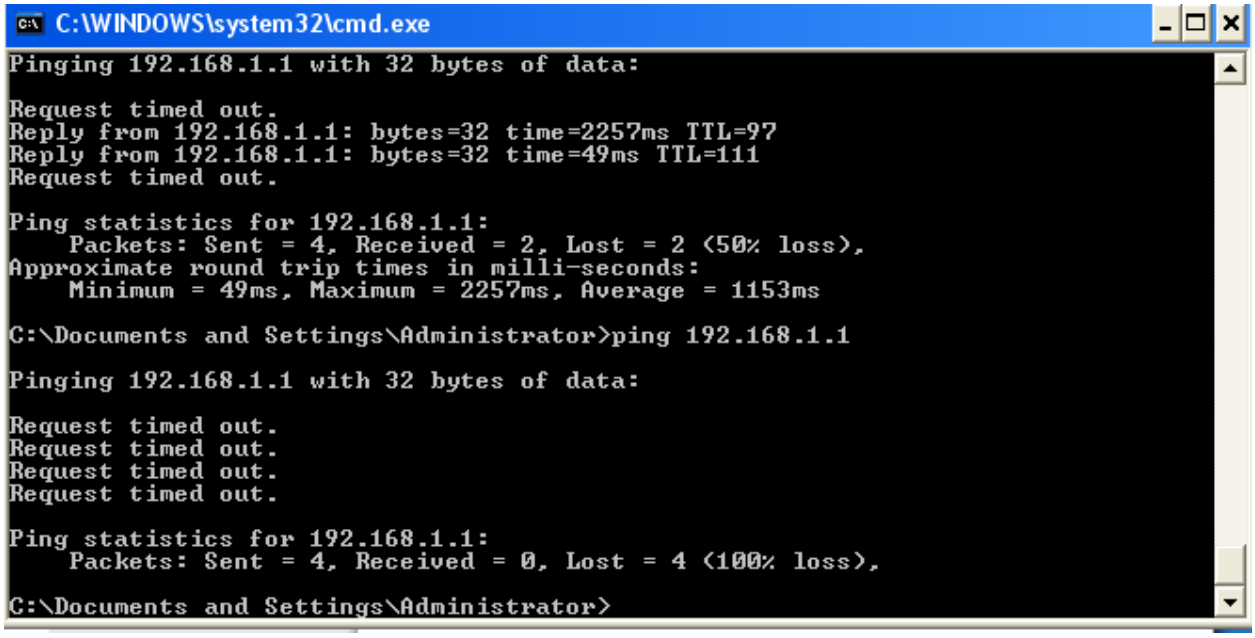


Figure 25: ICMP requests begin failing almost immediately during the ICMP attack

Unlike the TCP and UDP translations, the Linksys logs do not appear to keep track of ICMP translations. This makes tracking the connections through the Linksys device even more difficult than usual since all management is done in-band and cannot be used during the attack. However, unlike the other platforms, the ICMP attack has a catastrophic effect when executed on the Linksys platform.

3.9.4 VMWare Workstation Results

Unlike UDP, there is no configurable ICMP setting present in the vmnetnat.conf file. This paired with the inability to execute debugging commands on the NAT service make it almost impossible to determine the default timer for the ICMP process. Although there was no setting present, the VMWare NAT process translated ICMP packets by default and did not require additional configuration like the Extreme device did.

Much like the other platforms (excluding the Linksys), there was no noticeable impact on network connectivity during the attack. Internal devices could contact the external device during the attack and both FTP and TFTP transfers occurred successfully during the attack.

3.9.5 Vyatta 5.0.2 Results

As with all the other Vyatta tests, there are no configurable parameters for the ICMP timeout. Additionally, the defaults are not listed in the documentation, however when a show nat translations monitor detail is shown, we can see the entries being added in real time. This shows that the ICMP default timeout is 30 seconds.

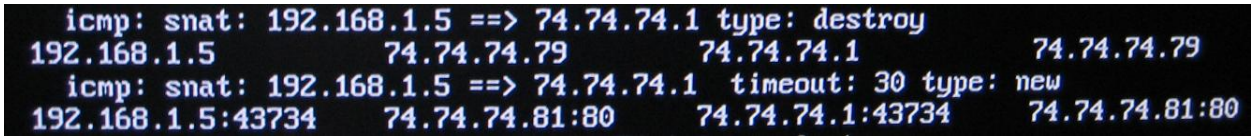


Figure 26: ICMP Entries being aged out and added in

Unfortunately the show nat statistics command only shows how many entries have matched the rule and not how many translations are currently active. However, after 5 minutes, a whopping 560,000 entries were translated. But due to the lack of output, determining how many were active was impossible. Due to the relatively fast expiration of these entries, it is likely that the number of active entries at one time was relatively minimal.

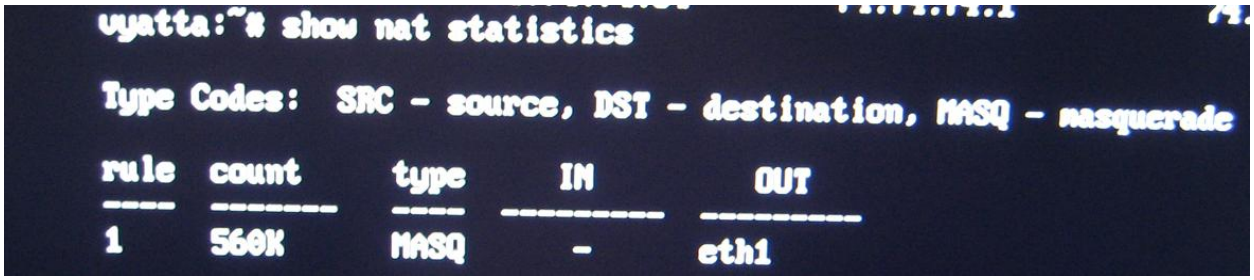


Figure 27: Number of entries matching the rule after 5 minutes

During the attack, there was no loss of connectivity between the inside hosts and the outside. Both FTP and TFTP transfers were successful during and after the attack. There didn't appear to be any adverse consequences to normal service throughout the entirety of this attack.

4. Additional Experiments

4.1 Extreme TCP / UDP Unlimited Timers Test Methodology

Since none of the tests (TCP, UDP, or ICMP) were able to fill up the translation table when the default timers were set, another set of tests was designed to run with the timers set to a extremely high number. This way, the effects of a full translation table on the Extreme platform could be observed. This test did not need to be executed on the other platforms because the effects of a

full translation table could be seen using the default timers (Cisco, Linksys, VMWare TCP). Unfortunately this test could not be performed on the Vyatta device because the timers were not configurable.

```
#!/bin/bash

For ((i=0; i<300; i++))

Do

    Sudo nmap -sU 74.74.74.3-254 1>/dev/null 2/dev/null &

Done
```

```
#!/bin/bash

For ((i=0; i<300; i++))

Do

    Sudo nmap -sS 74.74.74.3-254 1>/dev/null 2/dev/null &

Done
```

The same scripts that were used in the previous rounds of testing would still be used in order to see if there were any differences in the effects of a TCP vs a UDP attack on the device.

The topology for each test was set up as shown in previous experiments with three PCs connected to a switch behind a NAT device which then connected to an external 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 the Extreme device would be tested in these experiments. The Extreme device ran 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.

4.2 Extreme Unlimited Timers Experimental Results

4.2.1 Extreme Unlimited Timers TCP Results

For this test, the TCP timeout, syn timeout, and fin/rst timeout were set to 655555555 minutes to allow ample time to fill the translation table with entries. During the attack, hosts lose connectivity to the outside like in previous attacks and the CPU usage goes almost to 100 percent. Since none of the entries would expire, after two minutes and 30 seconds (on average) the Extreme device would stop adding new entries into the table. The maximum number achievable was 64,512 entries every time as can be seen in Figure 28.

```
* Summit200-24:143 # show nat stats
mapped  in      3       out     64515
added           64513
expired         1
no memory       0       bad nat 0
inuse           64512
rules           1
```

Figure 28: Maximum TCP entries for one outside address

Since there didn't appear to be any memory failures or other warnings from the Extreme device, this appears to be a predetermined maximum set by Extreme. This number is significantly lower than the maximum possible in the Cisco device. When the table was full, no new TCP connections could be established and UDP transfers using TFTP failed as well. After the table became full, no new packets could be seen on the outside network, meaning that the device not only did not make new translations, it dropped the packets that could not be translated as well. After the table was cleared, all devices operated normally.

4.2.2 Extreme Unlimited Timers UDP Results

For this test, the UDP timeout was set to 655555555 to allow ample time to fill the translation table with entries. During the attack there was a loss of connectivity between the inside and the outside hosts much like the other attacks. After 2 minutes and 30 seconds (on average) the translation table would fail to translate any new entries. The maximum number of entries in the table at this time was 64,512 which is exactly the same number as the TCP attack with large timers. Unlike the Cisco device, there were no memory failures and the router did not seem to be affected in any other way when the translation table was filled. After the attack stopped (and the table remained full), no new TCP or UDP sessions could be established between the inside and outside. The outside hosts did not even see any packets from the inside network once the table was filled. This means that the Extreme device not only does not translate new entries, it also drops the packets that cannot be translated. This is an extremely important, because it means that if the translation table can be filled by an attacker, all communication between the inside and outside will cease. The fact that the number of translations stops at exactly 64,512 for UDP and TCP points to the fact that this is a predefined number by the vendor and not a memory limitation unlike the Cisco device.

```
* Summit200-24:265 # show nat stats
mapped  in       1        out      64711
added            64512
expired          0
no memory        0        bad nat 0
inuse            64512
rules            1
```

Figure 29: Table full at 64,512 UDP entries

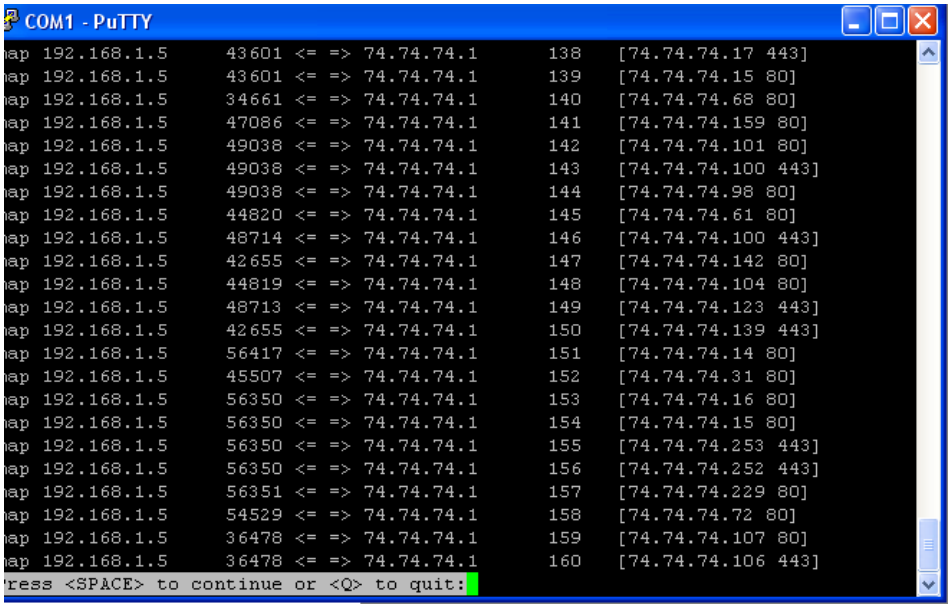
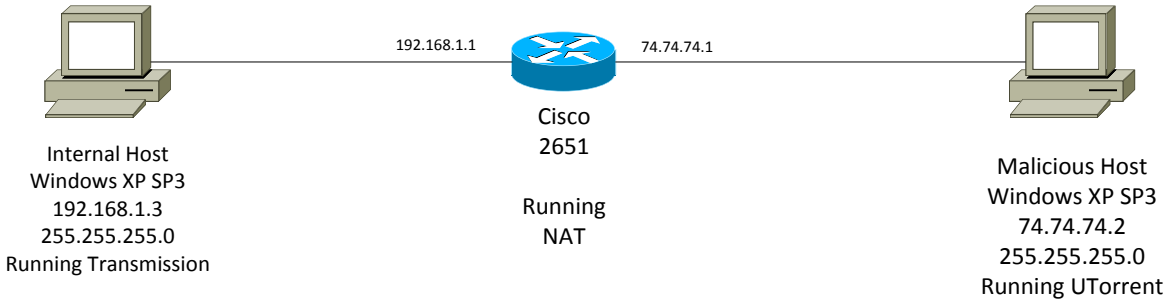


Figure 30: Portion of the UDP translation table

4.3 Methodology for Navigating the Translation Table to the Inside

Once a connection is set up in the translation table, any packets matching the specific entry in the table can pass back through to the inside. Generally this isn't a problem because the an attacker on the outside would have to somehow guess the outside destination address, port, source port and outside source address to gain entry. However if a node on the inside were to establish a connection to the attacker, the attacker could gather enough information to craft malicious packets and push them back through.

In order to simulate these circumstances the following topology was created:



The external host created a private torrent which pointed to itself instead of a tracker and in the real world would have had to distribute it through the web. The configuration to do this is present in Figure 31. The private torrent was chosen to eliminate extraneous variables introduced by the tracker, however a torrent using a tracker could be used to achieve the same results.

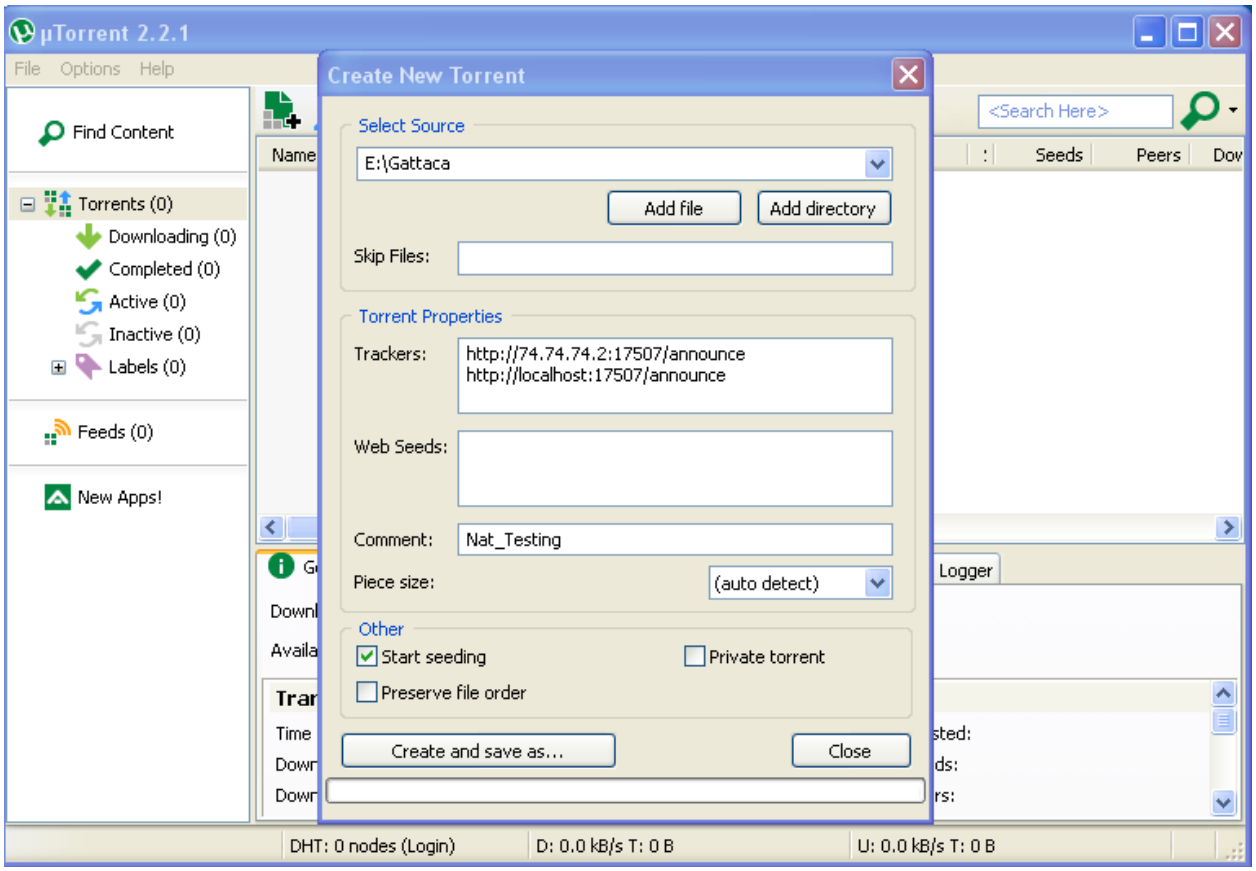


Figure 31: Private torrent configuration

The victim computer then ran the torrent file and established a session to the malicious host to start downloading the file. The upload speed from the malicious attacker was limited to 50Kbps in order to prolong the transfer and maintain the download session. The malicious attacker used wireshark to examine the packets and determine the ports / addresses to forge and then used Colasoft Packet Builder to craft various packets to send back through.

Additionally, since this is a private torrent a port had to be opened in the NAT device to allow the internal host to get out, however this would not be needed if a tracker was being used. However, this experiment assumes that the attacker has no knowledge of the forwarded port and must gain the information through packet analysis.

4.4 Results for Navigating the Translation Table to the Inside

After the internal host ran the torrent file, the attacker (running wireshark) could gather a large amount of information from the packet captures. Figure 32 shows the setup of the bittorrent connection between the inside and outside host.

1	0.000000	74.74.74.2	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
2	9.166941	74.74.74.1	74.74.74.2	TCP	43433 > 17507 [SYN] Seq=0 Win=4380 Len=0 MSS=1460 SACK_PERM=1 TSV=878127 TSER=0 WS=6
3	9.167011	74.74.74.2	74.74.74.1	TCP	17507 > 43433 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=0 TSV=0 TSER=0 SACK_PERM=1
4	9.167329	74.74.74.1	74.74.74.2	TCP	43433 > 17507 [ACK] Seq=1 Ack=1 Win=4416 Len=0 TSV=878127 TSER=0
5	9.167494	74.74.74.1	74.74.74.2	TCP	43433 > 17507 [PSH, ACK] Seq=1 Ack=1 Win=4416 Len=347 TSV=878127 TSER=0
6	9.167992	74.74.74.2	74.74.74.1	TCP	17507 > 43433 [PSH, ACK] Seq=1 Ack=348 Win=65188 Len=248 TSV=39483 TSER=878127
7	9.168336	74.74.74.1	74.74.74.2	TCP	43433 > 17507 [ACK] Seq=348 Ack=249 Win=4224 Len=0 TSV=878127 TSER=39483
8	9.168533	74.74.74.1	74.74.74.2	TCP	43433 > 17507 [FIN, ACK] Seq=348 Ack=249 Win=4224 Len=0 TSV=878127 TSER=39483
9	9.168558	74.74.74.2	74.74.74.1	TCP	17507 > 43433 [ACK] Seq=249 Ack=349 Win=65188 Len=0 TSV=39483 TSER=878127
10	9.168688	74.74.74.2	74.74.74.1	TCP	17507 > 43433 [FIN, ACK] Seq=249 Ack=349 Win=65188 Len=0 TSV=39483 TSER=878127
11	9.169306	74.74.74.1	74.74.74.2	TCP	43433 > 17507 [ACK] Seq=349 Ack=250 Win=4224 Len=0 TSV=878127 TSER=39483
12	14.991681	74.74.74.2	74.74.74.1	UDP	Source port: 17507 Destination port: 33731
13	14.992333	74.74.74.1	74.74.74.2	UDP	Source port: 33731 Destination port: 17507
14	14.992459	74.74.74.2	74.74.74.1	UDP	Source port: 17507 Destination port: 33731
15	14.992600	74.74.74.2	74.74.74.1	UDP	Source port: 17507 Destination port: 33731

Figure 32: Bittorrent Setup

The connection starts out with a normal three way handshake, however the push packets that are sent before the connection ends give the attacker another helpful hint of information before the TCP connection is torn down.

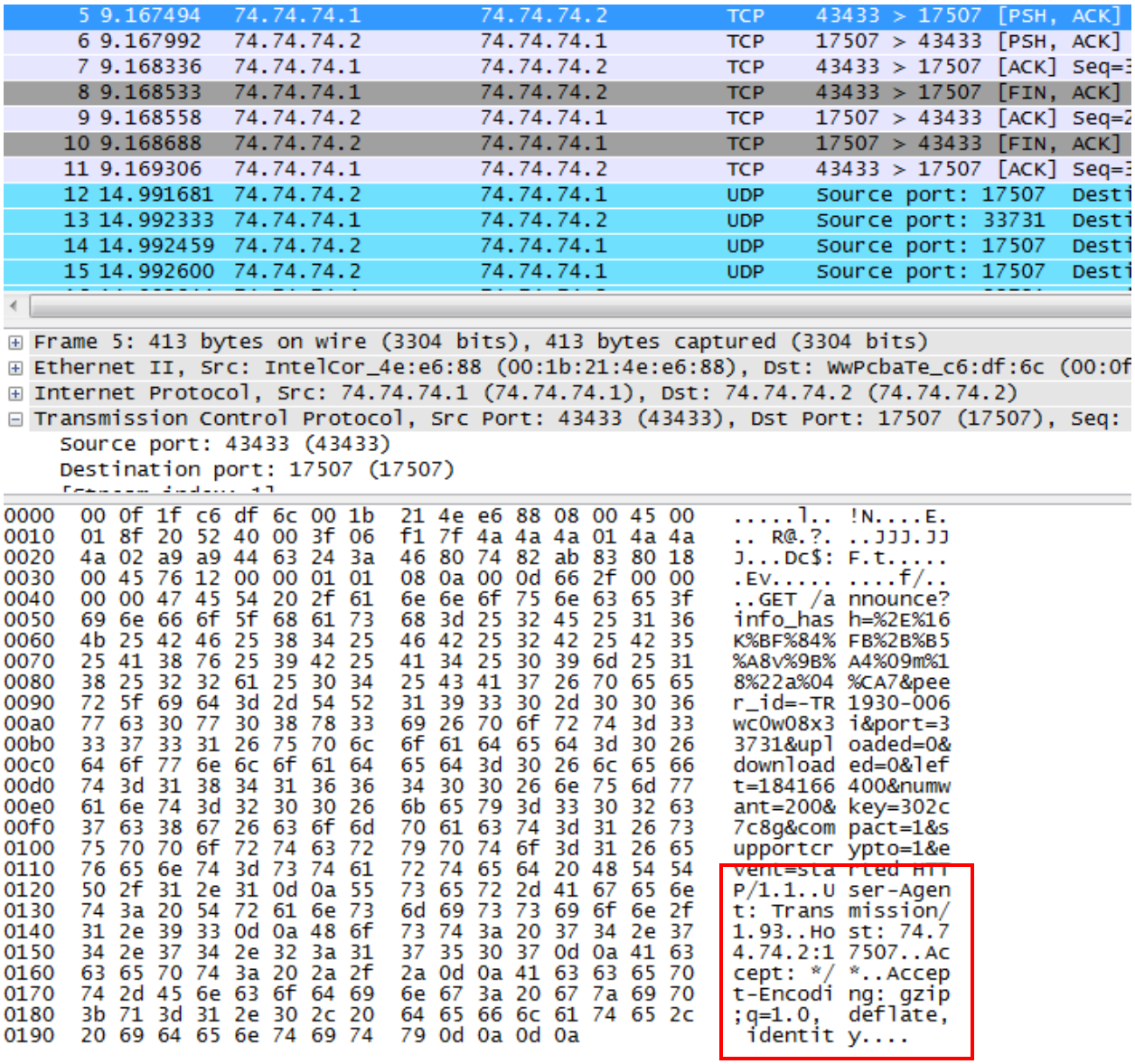


Figure 33: The first push packet

Figure 33 shows the data that is inside the first push packet, which is sent from the internal host. This contains the UDP port information (unfortunately not human readable) for the outside host to connect to and the software and version that the user is using to connect. The second push packet is from the external host confirming the ports and stating it's version.

After the initial TCP exchange, TCP and UDP packets are used for data transfer and since parts of the transfer do not have sequence numbers (UDP), it is much easier to craft packets for this stream using packet building software. In this experiment, Colasoft Packet Builder was used to edit UDP packets and Colasoft Packet Player was used to send the packets into the network, however any software such as hping or scappy could be used to customize packets to send back through the network.

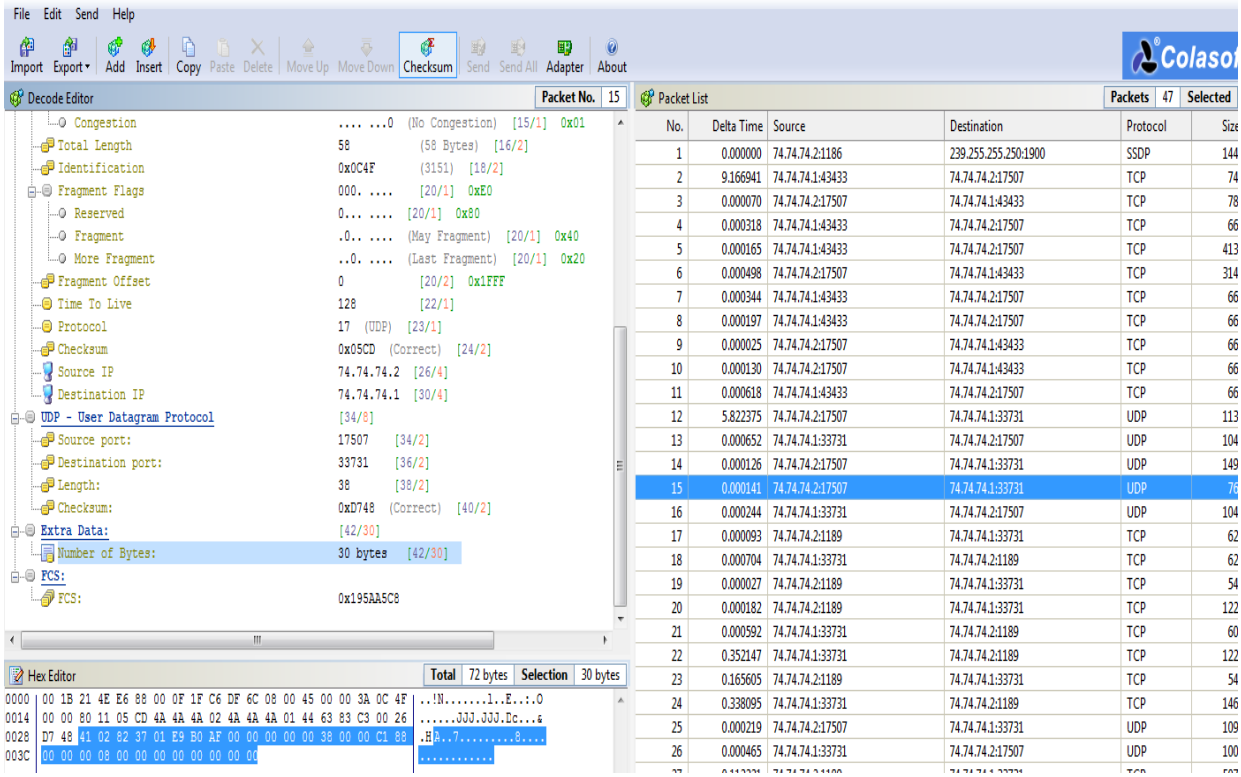


Figure 34: Colasoft Packet Builder

In order to make sure that the packet was received on the inside of the network, the payload was made into an easily recognizable patter as shown in Figure 35.

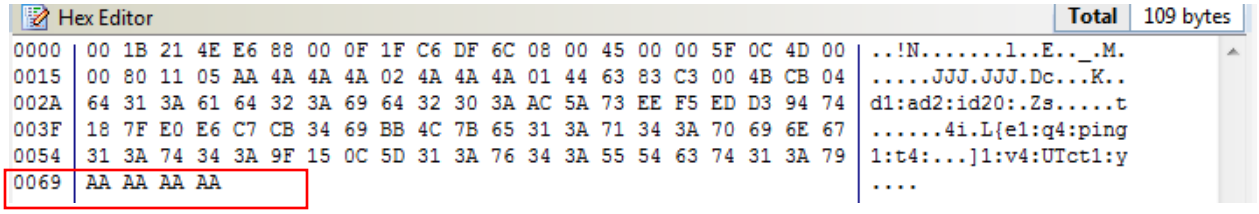


Figure 35: Easily recognizable UDP payload pattern

While the bittorrent transfer was running, the attacking computer sent several of these type packets through the open connection in the NAT translation table and they were received by the internal host as can be seen in Figure 36.

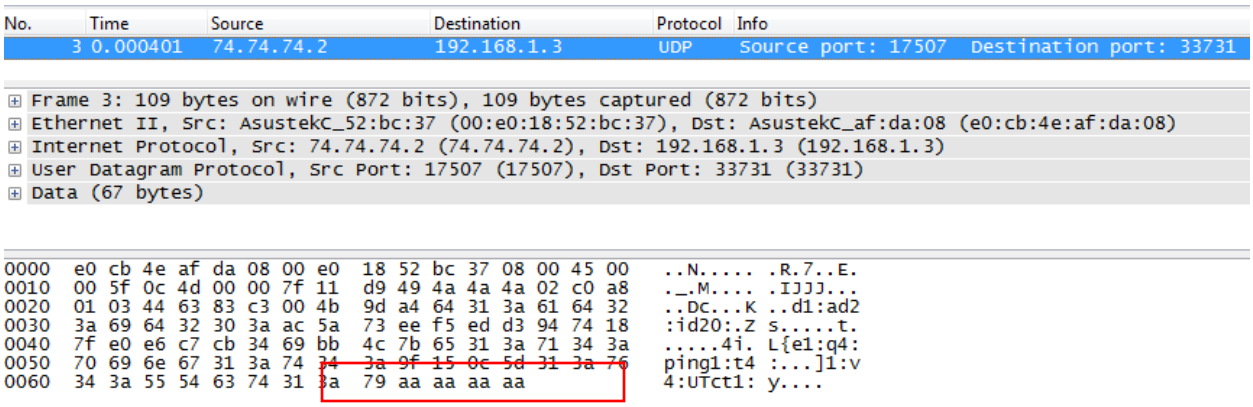


Figure 36: Crafted Packet (Internal Network)

These results show that an attacker can use connection data to successfully map a connection between the inside and outside of the NAT network and send crafted packets through. Additionally, since bittorrent clients send out the software type and version when initiating the session, this leaves open the possibility of attackers using these connections to exploit vulnerable torrenting clients after a connection has been established.

5. Experiment Summary

	TCP Attack	UDP Attack	ICMP Attack	Unlimited Timers
Cisco 2651	-Entries expired quickly -High CPU during attack -Connectivity Loss during attack -Connectivity restored after attack	-Entries took time to expire -Translation table filled -High CPU during attack -Connectivity lost during attack -Connectivity lost after attack, until entries expired	-Entries expired quickly -High CPU during attack -No loss of connectivity during or after the attack	N/A
Extreme Summit 200	-Entries expired quickly -High CPU during attack -Connectivity Loss during attack	-Entries expired quickly -High CPU during attack -Connectivity Loss during attack	-Entries expired quickly -High CPU during attack -No loss of connectivity	-Translation table filled -High CPU during attack -Connectivity lost during attack

	-Connectivity restored after attack	-Connectivity restored after attack	during or after the attack	-Connectivity lost after attack, table entries never expired
Linksys	-Immediate loss of connectivity during the attack -Connectivity restored after the attack	-Immediate loss of connectivity during the attack -Connectivity restored after the attack	-Immediate loss of connectivity during the attack	N/A
VMWare	-Entries took time to expire -Translation table filled -Connectivity lost during attack -ICMP connectivity regained after attack -TCP / UDP connectivity never restored	-Entries expired quickly -Slight latency increase -No loss of connectivity during or after the attack	-No loss of connectivity during or after the attack	N/A
Vyatta	-No loss of connectivity during or after the attack	-No loss of connectivity during or after the attack	-No loss of connectivity during or after the attack	N/A

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.

Despite the differences in implementation, all of the devices except for the Vyatta device experienced extremely high CPU utilization as a result of the NAT translations. This actually proved as effective if not more effective at denying service to internal users than filling up the translation tables did. The high CPU utilization ended up causing the network outage long

before the tables were filled up on these devices. While larger and more robust devices would be able to handle a higher load, these larger devices are generally under a heavy load in an enterprise setting. Therefore, creating a large number of translations is an effective strategy to deny service in even very large network settings.

In addition to the denial of service attacks, an attacker can also craft packets that can navigate through the NAT Translation table. This becomes much more likely in scenarios where inside hosts are establishing active connections to the attacking node (in this case during a torrent). Despite the fact that it requires relatively little effort to do, this type of attack is overly complex and begs the question of why an attacker would use such a method when they could just disguise a malware file as the file the user is downloading from them.

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.

6. Mitigation Techniques

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. Therefore grouping networks into lists would be the best option to limit the amount of users that can be denied service by a single node. However even this method would become incredibly cumbersome on a large network with many subnets. Additionally, having to keep track of the number of connections each list currently has requires processing time and could adversely impact the performance of the device.

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.

The ideal type of configuration strategy to safeguard against this type of attack would be a command that could limit the number of translations allowed per user (that doesn't have to be specifically defined) to a specified amount. This way the administrator can determine what constitutes a reasonable number of translations per host and put the policy into place quickly and easily. However this technique would still require processing power on the part of the NAT device, and could degrade network performance. Additionally, this technique would not prevent an attacker from spoofing different source addresses, but it would make it significantly more difficult to execute the attack.

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. Despite being able to prevent the table from filling up, the Extreme device was taken down due to the high CPU utilization of the NAT attack. Therefore, the best solution to preventing this type of attack is to use a combination of rate limiting and carefully adjusted timers. This way legitimate users can still make connections, stale connections can age out in a timely fashion, and the number of connections that can be made by a single host can be limited to a reasonable amount.

Another important device that could help limit or at least minimize the effects of this attack is an IDS device. These kinds of devices analyze the network to establish a baseline and can detect when traffic patterns deviate significantly from the norm. This device could alert an administrator when a compromised node is executing this attack and the administrator could then take action to remediate the problem. However, these devices need to be calibrated so that they do not confuse normal behavior of legitimate hosts with that of an attacker, otherwise the false positives could cause a delay when trying to pinpoint the source of an attack. These mitigation techniques work well in theory, however experiments to gauge the effectiveness of these techniques were not tested in this study.

7. Final Conclusions

While NAT with Port Translation is widely deployed and is in fact, necessary for internet connectivity in the modern age, many of the drawbacks of using NAT with Port Translation are often overlooked. These experiments show that as a consequence of this fact, equipment used to create NAT are vulnerable to attacks that can severely impact network connectivity to a large number of users. The NAT translation table creates a single point of failure that will block out all legitimate traffic if it is filled. While the consequences of filling the table differ in severity depending on the vendor, each case where this was accomplished negatively impacted the network in a substantial way.

While there are several RFCs that establish NAT and attempt to standardize NAT behavior, these experiments show that NAT implementations can vary significantly between Vendors. A detail that is left unclear in the RFCs is what the maximum translation table size should be. This experiment showed that this detail also varies from vendor to vendor. The translation table on the Cisco device is limited only by the amount of memory that can be filled with translation data while the maximum number of translations on the Extreme device is limited by the manufacturer to a specific number of translations per outside IP address. The Vyatta device appeared to act like the Cisco device, however I was unable to fill the translation table to verify this. The experimental tests on the Linksys and VMWare were inconclusive as to the maximum size of their translation tables due to the lack of output available from each of these solutions.

Some of the other features that tended to vary were the default timers for various types of traffic, the configurability of these timers, and TCP session tracking. These experiments show that the inclusion of TCP session tracking in order to expire unfinished TCP connections not only effectively mitigates the TCP DOS attack that was demonstrated, but also helps to keep the table size to a minimum during normal operation. However, despite the fact that both Cisco and Extreme include this feature, their default timers end sessions extremely fast when compared to the recommendations in RFC 2663. While this does age out stale entries rapidly, this could result in application issues due to the graceless nature in which connections would be closed.

Additionally, this experiment also showed that the ability to adjust timers is critical in controlling NAT translation table size. RFC 2663 notes recommends that vendors include this feature for this very reason. While adjusting timers to suit the network needs is an integral part of NAT operation, network administrators must be extremely careful when doing so to avoid terminating legitimate sessions prematurely. Even though the lack of the feature could be expected with a personal use device like the Linksys router which was tested, it was surprising to find that VMWare workstation lacked the capability to adjust timers for TCP sessions and that Vyatta lacks any options to configure NAT timers.

A few traits were fairly consistent on almost all of the platforms. This experiment showed that the process of creating, maintaining, and expiring NAT translations is extremely CPU intensive. All platforms except the Vyatta device were severely affected during the attack. Despite the fact that the translation tables were not full on the Cisco and Extreme devices, all network connectivity between the inside and outside was lost when the attack was ongoing. Additionally the Linksys device suffered immediate network failure during the attack because of this fact. The VMWare device was also impacted during the attack but less so than the other platforms. The Vyatta device is the only outlier and this can be attributed to the overwhelmingly powerful hardware that was used when compared to the other devices.

An additional fact that was discovered during the process of this experiment was the fact that unlike when the CAM becomes full on a switch, the router does not "fail open" like switches do. When the translation table becomes full on a NAT device, the device will not translate any traffic that is not already in the table until an entry ages out and a new entry can be added in. Furthermore, the router will also drop any new traffic if new translations cannot be made. It does not send out the traffic un-translated as is the case when traffic does not match the translation rule.

This experiment also showed that an attacker could utilize knowledge of current NAT translations to send packets back to the inside network. Torrent programs in particular establish many connections to many different hosts, thereby opening many avenues for external hosts to get to the inside network. However, this attack requires that an inside host make some sort of connection or transfer with the attacker, which makes executing this attack somewhat convoluted

considering the attacker could just have the inside host download a malware file instead. Therefore this attack isn't very likely to occur in a real world scenario.

From these experiments it can be concluded that network administrators need to become familiar with the equipment they are deploying in their networks when using NAT with Port Translation. Since each vendor has their own way of doing things, and different capabilities, administrators must configure their devices based on need in order to ensure optimum efficiency and to help mitigate possible threats from attackers targeting the NAT. In order to further protect the network, administrators should also employ IDPS devices as an early warning and to help isolate malicious users before they can cause a large network outage.

8. Future Work

Since these experiments were limited to the devices available in the RIT labs with a limited amount of hosts, future work could involve performing these experiments in an environment with more active traffic to determine if it becomes easier or more difficult to interrupt legitimate traffic. Additionally, future work could be done to determine if other mainstream platforms such as Juniper Networks are able to be compromised in the same manner as these experiments. Furthermore, while this experiment proved that knowledge of the NAT translation table could be used to navigate to the inside, more research could be done in order to determine if certain applications are more vulnerable to being compromised using this method than others. IDS and IDPS devices could also be used in order to determine their effectiveness at detecting and mitigating the attacks that were demonstrated in this experiment.

9. Works Cited

[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

10. Appendix

10.1 Cisco Basic NAT Configuration

```
interface FastEthernet0/0
 ip address 74.74.74.1 255.255.255.0
 ip nat outside
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
!
interface FastEthernet0/1
 ip address 192.168.1.1 255.255.255.0
 ip nat inside
 duplex auto
 speed auto
!
ip nat inside source list translate interface FastEthernet0/0 overload
ip classless
ip http server
!
!
ip access-list standard translate
 permit any
```

10.2 Extreme Basic NAT Configuration

```
# Configuration Mode
create vlan "n_inside"
create vlan "n_outside"

# Config information for VLAN n_inside.
configure vlan "n_inside" ipaddress 192.168.1.1 255.255.255.0
configure vlan "n_inside" add port 1 untagged

configure vlan "n_inside" add port 2 untagged

#
# Config information for VLAN n_outside.
configure vlan "n_outside" ipaddress 74.74.74.1 255.255.255.0
configure vlan "n_outside" add port 24 untagged


# NAT configuration
configure nat add "n_outside" map source any to 74.74.74.1/32  both portmap
configure nat add "n_outside" map source any to 74.74.74.1/32
```

```
configure nat n_outside outside
configure nat "n_inside" inside
enable nat
```

10.3 Cisco NAT Timer Commands

```
ip nat translation udp-timeout 655555555
ip nat translation tcp-timeout 655555555
ip nat translation icmp-timeout 65555555
ip nat translation syn-timeout 655555555
ip nat translation finrst-timeout 65555555
```

10.4 Extreme NAT Timer Commands

```
configure nat timeout 655555555
configure nat tcp-timeout 655555555
configure nat udp-timeout 655555555
configure nat icmp-timeout 655555555
configure nat finrst-timeout 655555555
configure nat syn-timeout 655555555
```

10.5 VMNetnat.conf

```
# Windows NAT configuration file

[host]

# NAT gateway address
ip = 192.168.40.2/24
hostMAC = 00:50:56:C0:00:08

# enable configuration; disabled by default for security reasons
#configport = 33445

# VMnet device if not specified on command line
device = vmnet8

# Allow PORT/EPRT FTP commands (they need incoming TCP stream...)
activeFTP = 1

# Allows the source to have any OUI.  Turn this one if you change the OUI
# in the MAC address of your virtual machines.
allowAnyOUI = 1

[udp]
# Timeout in seconds, 0 = no timeout, default = 30; real value might
# be up to 100% longer
timeout = 30

[dns]
# This section applies only to Windows.
```

```
#
# Policy to use for DNS forwarding.  Accepted values include order,
# rotate, burst.
#
# order: send one DNS request at a time in order of the name servers
# rotate: send one DNS request at a time, rotate through the DNS servers
# burst: send to three servers and wait for the first one to respond
policy = order

# Timeout in seconds before retrying DNS request.
timeout = 2

# Retries before giving up on DNS request
retries = 3

# Automatically detect the DNS servers (not supported in Windows NT)
autodetect = 1

# List of DNS servers to use.  Up to three may be specified
#nameserver1 = 198.41.0.4
#nameserver2 = 192.36.148.17
#nameserver3 = 202.12.27.33

[netbios]
# This section applies only to Windows.

# Timeout for NBNS queries.
nbnsTimeout = 2

# Number of retries for each NBNS query.
nbnsRetries = 3

# Timeout for NBDS queries.
nbdsTimeout = 3

[incomingtcp]
# Use these with care - anyone can enter into your virtual machine through these...

# FTP (both active and passive FTP is always enabled)
#      ftp localhost 8887
#8887 = 192.168.27.128:21

# WEB (make sure that if you are using named webhosting, names point to
#     your host, not to guest... And if you are forwarding port other
#     than 80 make sure that your server copes with mismatched port
#     number in Host: header)
#      lynx http://localhost:8888
#8888 = 192.168.27.128:80

# SSH
#      ssh -p 8889 root@localhost
```

```
#8889 = 192.168.27.128:22

[incomingudp]
# UDP port forwarding example
#6000 = 192.168.27.128:6001

[PrivilegedTCP]
autodetect = 1

[PrivilegedUDP]
autodetect = 1
```

10.6 Vyatta Basic NAT configuration

```
Service {
  NAT {
       Rule 1 {
            Outbound-interface eth1
            Source {
                 192.168.1.0/24
            }
             Type masquerade
       }
   }
}
```