Spring 3-2021

Traversing NAT: A Problem

Tyler Flaagan
Dakota State University

Traversing NAT: A Problem

A dissertation submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Cyber Operations

March 2021

By
Tyler Flaagan

Dissertation Committee:

Dr. Kyle L. Cronin
Dr. Michael J. Ham
Dr. Mark L. Hawkes

DAKOTA STATE
UNIVERSITY

DISSERTATION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Philosophy degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: Tyler Flaagan

Dissertation Title: Traversing NAT: A Problem

Dissertation Chair/Co-Chair: Kyle Cronin Date: April 1, 2021
Name: Kyle Cronin

Dissertation Chair/Co-Chair: __________ Date: __________
Name:

Committee member: Michael Ham Date: April 1, 2021
Name: Michael Ham

Committee member: Mark Hawkes Date: April 1, 2021
Name: Mark Hawkes

Committee member: __________ Date: __________
Name:

Committee member: __________ Date: __________
Name:

Original to Office of Graduate Studies and Research
Acid-free copies with written reports to library

Abstract

This quasi-experimental before-and-after study measured and analyzed the impacts of adding security to a new bi-directional Network Address Translation (NAT). Literature revolves around various types of NAT, their advantages and disadvantages, their security models, and networking technologies' adoption. The study of the newly created secure bi-directional model of NAT showed statistically significant changes in the variables than another model using port forwarding. Future research of how data will traverse networks is crucial in an ever-changing world of technology.

DEDICATION

I dedicate this dissertation to my parents, without which I would not be who I am today. Thank you, and I love you both.

ACKNOWLEDGEMENTS

Completing this dissertation has been one of the most significant challenges in my life. I would not have completed it without many people's support before I even started and along the way. Writing these acknowledgments to all those that have helped is not an easy task on its own. To all my committee members for helping me through the process, committee chair Kyle Cronin, thank you for you all your support throughout the past few years. During the challenging times, you helped me keep moving towards the goal. Mike, thank you for always digging deeper and providing feedback faster than what should be expected. Mark, thank you for all your time and input and for putting up with all the technical jargon along the way.

To the East Hall 6 team, without whose help, I would not be writing this section. Whether it was bouncing around ideas, taking a second look at code, supporting my infrastructure needs, and everything in between, your help along the way is very much appreciated. I hope that we can keep the winning tradition alive and continue to create successes.

To all my other DSU friends, I can confidently say I would not be where I am today without your help along the way. Between the classes I have taken from you, the advice you have given, general support, and even all the lectures I got about just getting it done. I know that I would not be at the end of this journey without your help. Tom, Josh, Wayne, Rob, Kathy, Brent, and Austin, thank you for all the help for what has now been almost ten years at DSU as a student and faculty member.

The list above is not a comprehensive list of those that have helped me get here. To those I have missed, know that you have contributed to this project’s success and beyond. I have a debt of gratitude to all those mentioned above and those missed.

It has been a long journey looking back at where I started and where I have ended up. I hope that this completion leads to further successes of teams in the many areas that we have done

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Tyler Flaagan

Tyler Flaagan

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The Internet's design was to pass information to and from systems interconnected by a global network using the Internet Protocol's functionality (IP) (Postel, 1981). The Request For Comments (RFC) for IP proposed that hosts meant to send and receive datagrams from one another would be identified by a fixed-length address that would be 32 bits in size (Postel, 1981). A fixed-length addressing scheme limits the number of addresses to a finite number since it is not expandable. Originally unforeseen challenges were introduced due to the rapid growth in technology regarding the depletion of IP addresses (Beeharry & Nowbutsing, 2016). The exhaustion of IP addresses led to a solution that would allow systems to have access to the Internet still but be logically separated. Network Address Translation (NAT) is commonly used to connect devices with a private network address to the public Internet to use publicly available resources (Srisuresh & Holdrege, 1999). The standard was built with a bi-directional traversal option but is limited in its configuration and security mechanisms. Chapter 2 will further explore these limitations. Bi-directional traversal allows for the initiation of sessions from either side of a device providing NAT services. The addition of security may cause the existing process to incur additional overhead. This study documents the development of a new method of NAT traversal that provides dynamic authentication. The introduction of dynamic authentication improves security for both a client and server and enables them to initiate an Internet traversing conversation with one another from either direction. The purpose of this study was to determine the impact on CPU usage, memory usage, and round trip time of packets of added authentication methods in a new approach to bi-directional NAT traversal.

Chapter 1 will explore a new model's proposed study to traverse NAT bi-directionally with added authentication mechanisms. The chapter will include the background, purpose, significance, design, assumptions, and scope of the study, emphasizing the problem to be solved and the research questions that drove the study.

Background of the Study

The rapid growth of the Internet brought new challenges, such as depleting globally unique addresses (Beeharry & Nowbutsing, 2016). These challenges have caused the redesign of various methods, protocols, and services. RFC memorandum 1918, Address Allocation for Private Internets*,* states that the motivation for creating private addresses was due to the unanticipated proliferation of the Internet by its creators (Rekhter, Moskowitz, Karrenberg, de Groot, & Lear, 1996). This continual growth presented new issues that required attention to allow the continued evolution of the Internet. The first challenge cited is that globally unique Internet Protocol (IP) addresses will be exhausted (Rekhter et al., 1996). RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan was released in 2006 and echoed this issue. It stated that CIDR's intention was not to slow the consumption of globally unique Internet Protocol version 4 (IPv4) addresses, requiring an improved and more long-term solution (Fuller & Li, 2006). The implementation of RFC 1918 addresses, otherwise known as private addresses, alone did not allow private resources to connect to the Internet. The RFC for private addresses describes that an organization or entity that uses private addresses loses its flexibility to connect to the Internet (Rekhter et al., 1996). Later, RFC 1918 specifies that if a host connected to the network via a private address needed access to the Internet, it would require renumbering. Subsequently, if a system was connected using a public address and is no longer needed or the organization required a different system connected to the Internet, renumbering would also be required (Rekhter et al., 1996). Readdressing or renumbering every time there is a change in the network would cause a significant amount of overhead to keep an organization connected to the Internet.

RFC 2663, IP Network Address Translator (NAT) Terminology and Considerations proposed a solution to private addresses not having the ability to be used outside of their internal private network (Srisuresh & Holdrege, 1999). RFC 2663 proposed NAT as a solution for private addresses to communicate with the public Internet without using a globally unique address. According to its RFC memorandum, NAT is a method that maps IP addresses from one addressing scheme to another. One of NAT's common uses is to connect devices on a network that implements RFC 1918, or private addresses, to addresses that are globally unique and publicly available on the Internet (Rekhter et al., 1996). RFC 2663 goes further to describe the variants available (Srisuresh & Holdrege, 1999).

The variants of NAT described in RFC 2663 include basic NAT, network address port translation (NAPT), bi-directional or two-way NAT, twice NAT, and multihomed NAT (Srisuresh & Holdrege, 1999). While each of the previously mentioned types of NAT provides a slightly different feature set than the last, they all perform a similar function of providing a transparent routing solution even when different networks are used (Srisuresh & Holdrege. Separately, Suzuki, Goto, and Watanabe suggest that there are three categories of NAT. These categories include behavior-based NAT, control-based NAT, and a third type described as NAT-less (Suzuki, Goto, & Watanabe, 2007).

The first type, behavior-based NAT, includes protocols such as Session Traversal Utilities for NAT (STUN). Traversal Using Relays around NAT (TURN) and Relay Extensions to Session Traversal Utilities for NAT (NAT STUN). STUN and TURN NAT traversal work by allowing applications to discover the type of NATs and firewalls that separate them from the Internet using third-party servers. TURN NAT traversal more uses the external server explicitly as a relay between itself and another host. When both the internal host and the external host reside behind separate NATs, STUN requires the TURN extension (Mahy, Matthews, & Rosenberg, 2010). In both examples, the traversal mechanisms are exiting the network to learn more about the networking device and its configuration that controls their traffic instead of making modifications (Rosenberg, Weinberger, Huitema, & Mahy, 2003). The second type, control-based NAT, is a form of port forwarding done automatically by the internal device (Suzuki et al., 2007). The primary user of control-based NAT is Universal Plug and Play (UPnP) Internet Gateway Device – Port Control Protocol Interworking Function (IGD-PCP IWF) (Boucadair, Penno, & Wing, 2013). UPnP is a control-based NAT because the internal client on the private network is permitted to connect to the IGD using Port Control Protocol (PCP) to make modifications to the NAT table. The designers of UPnP considered security features during its implementation. However, none focus on solving the NAT traversal problem and instead focus on stopping malicious activity originating on the client device destined for the IGD (Boucadair et al., 2013).

Since the security features of UPnP are focused on stopping malicious activity within a network and do not focus on solving the NAT traversal problem, they are out of the scope of comparison for this study (Boucadair et al., 2013). Another control-based

NAT traversal technology, NAT-free, is proposed by Suzuki et al. (2007). NAT-free is similar to the original bi-directional NAT proposed by the NAT RFC (Srisuresh & Holdrege, 1999). It is similar because DNS, an external system, is still required to connect the internal client and the external entity (Suzuki et al., 2007).

The final type described by Suzuki et al. (2007) is NAT-less. There are a few methods that fit into this category. This category's primary qualification relies on modifying the IP headers and requires changes in how the IGD routes traffic (Suzuki et al., 2007). Chapter 2 will further discuss the various forms of NAT and their implementations.

An IGD or Internet Gateway Device is commonly known as a router in network architecture. A router routes traffic from any connection that it has access to and sends traffic to either the correct destination or its default gateway. This routing works by the router reading the headers of an incoming packet and directing the packets to the correct destination based on its rules and the packet's header. If the router's rules prevent a packet from crossing a boundary based on any criteria found within the packet, the router will discard the packet. There is a configuration of NAT where the external interface has a single public IP address and a single internal interface responsible for an entire private local area network (LAN). In this case, a rule may block all incoming traffic on the external interface attempting to access the internal network.

Statement of the Problem

Since its conception, the Internet's evolution has been aggressive and has grown beyond many expectations (Rekhter et al., 1996). It evolved from a simple way to send messages to a multi-faceted conglomerate of services. During its evolution, the changes brought new attention from individuals and groups with malicious intent.

NAT is a protocol that is an example of the lack of security consideration during its inception (Rekhter et al., 1996). Without concerns made for security in the design, NAT relies on other services or protocols for security, such as IPSec (Srisuresh & Holdrege, 1999). Chapter 2 will discuss in further detail the various types of NAT and their configurations.

The initial memorandum describing NAT does not provide security considerations; the memorandum mentions that in the recommended configuration where there is an external connection, the network should filter any private networks from inbound routing information (Rekhter et al., 1996). As a result, a system external to the network receiving or attempting to send data to an internal system only sees the Internet Gateway Device (IGD) (Rekhter et al., 1996). Since an external device can only see the external address of the IGD, it cannot directly connect to internal devices without other changes to the network. The previous section outlined a few of the initial solutions to the problem.

Gaps exist in the previous NAT types that do not account for security and allow traffic to flow across an IGD in both directions easily (Keranen, Holmberg, & Rosenberg, 2018; Novo, 2018; Yang & Lei, 2016). Designing and implementing a new variant of NAT includes a security layer, and bi-directional traffic closes that gap. This authenticated NAT allows traffic to traverse bi-directionally across an IGD to enable services required by a device residing on a private network and does not require a third-party service to help identify network configuration. Various systems stand to benefit from the bi-directional communication and the added forms of security that result from a new method of traversing NAT.

Purpose of the Study

The experiment studied the impacts of added authentication methods that rely on cryptography to securely allow bi-directional communication across an IGD without imposing a significant adverse effect on the network. The impacts studied were the CPU consumption, memory consumption, and the round-trip time of data that traverses the network. This NAT implementation allows external entities to authenticate through an IGD to communicate with an internal entity or vice versa. According to Kumar (2019), a quasi-experimental study has characteristics from both an experimental and non-experimental study. A non-experimental study is that the researcher does not have complete control over every variable in the study. The experimental characteristic is the researcher will be introducing what will be assumed to be the cause of change in the network (Kumar, 2019).

Significance of the Study

NAT is a popular solution in many current networking solutions due to the lack of globally unique IPv4 addresses and the relatively slow transition to more permanent IPv6 addresses (Beeharry & Nowbutsing, 2016; Zhang, Zhu, Han, Zhang, & Feng, 2016). It allows for significant flexibility in designing a network and mitigates challenges in a rapidly changing network environment. A common use of NAT consists of an internal entity reaching out from a private network to an external entity on the Internet. This application is limited in the flexibility it provides a network as it only allows for creating a session in one direction across an IGD. This limitation has been the reason for the design of alternative versions of NAT that will be discussing in detail in Chapter 2.

Before the exhaustion of IPv4 addresses, researchers identified that the Internet's growth would eventually consume the available (Beeharry & Nowbutsing, 2016). The solution to an ever-increasing number of connections globally is a new addressing scheme, IPv6. Even with the increasing adoption rates for IPv6 (Beeharry & Nowbutsing, 2016), While IPv4 addresses are in use, there is still a need for NAT in many traditional networks. A NAT solution that allows for many new features also could support IPv6 addresses in various ways while still providing connections for IPv4 tenants.

A significance of this study is that it allows traversal of network borders while enabling security features. These security features could prevent various attacks such as a distributed denial of service, unverified third-party compromise via port openings, and others. Enabling this security protects the end-user transparently.

Nature of the Study

At the beginning of the research process, the researcher must decide on the type of research conducted. The kind of research to be undertaken is chosen based on the perspective of the researcher. While not mutually exclusive, the researcher's view will decide which type of research best lends itself to the analysis performed. While this research could focus on the application perspective or the objectives perspective, the mode of inquiry perspective seems most suitable (Kumar, 2019).

Within the mode of inquiry perspective, there are three approaches available to the researcher: quantitative or structured in approach, qualitative or unstructured in approach, and mixed methods, which have qualities of both. The quantitative approach is most apt for adoption considering the highly technical nature of this study and electronics' precision. Creswell and Creswell (2018) also suggest using a quantitative view in circumstances where the study's data is predetermined. In this case, the data under study is the overhead created by modifications made to NAT. Overhead is the amount of CPU usage, memory usage, and round-trip time of packets in the experiment.

A quantitative approach attempts to measure variables objectively evaluate the variation in the phenomenon induced by the researcher communicates findings analytically and places significance on the validity and reliability of conclusions (Kumar, 2019).

According to Kumar (2019), a quantitative research study can be classified based on perspective. Three considerations must be taken into account to decide on the study design. The first of which is how many contacts the researcher has with the study population. The second is the reference period of the study. Moreover, the final consideration is the nature of the investigation. This study measured data produced in an environment, applied a change to the environment, then reran the same test. This experimental nature places the survey under the third category; studies based on the type of investigation (Kumar, 2019).

Kumar (2019) explains that there are various study designs based on the nature of the investigation. These designs are experimental, non-experimental, and quasi- or semi-experimental designs. The decision to use one over the other can be decided based on how the relationship is studied. If it is examined by observing a phenomenon, then searching for the cause, the experiment can be considered non-experimental. If the opposite is true, wherein the researcher induces the phenomenon by introducing the cause, the study is experimental. A combination of these two is quasi-experimental or semi-experimental. In this study, the researcher induced the environment's change, resulting in either quasi-experimental or experimental. Since the study will lack the population's randomization, it is quasi-experimental (Kumar, 2019).

Within experimental studies, there are a variety of designs to be considered for use. The quasi-experimental study has properties of both experimental and non-experimental studies. The experimental design was the most appropriate due to its technical nature compared to using a non-experimental design. The best-suited model for this study is the before-and-after experimental design. This design is the best choice for the study as the researcher did not have to construct the original observation as it was available retrospectively. The reason for choosing this design over the control group design, another design that has measurements for both before and after, is that there are no extraneous variables to be accounted for using a control group in the study (Kumar, 2019).

Research Questions

According to Kumar (2019), objectives are what the researcher sets out to gather in their study, and that wording the objectives is essential. The researcher's objectives guide the study as they are concerned with the study's overall direction and any relationships the researcher seeks to establish (Kumar, 2019). This study's primary objective was to inquire about the change in overhead due to adding security mechanisms to NAT traversal in networks that allow for and bi-directional traversal. The research question that guides this objective is as follows:

> What are the impacts of additional authentication methods that rely on cryptography that allow bi-directional communication across an IGD conducting NAT traversal, and do those authentication methods cause enough overhead to the end devices IGD to impose a negative impact on the network as a whole?

Sub-objectives will also be defined to support the primary objective. These sub-objectives will support the primary objective but give further clarity to the direction of the study. The sub-objectives of this study are as follows:

1. Determine the extent of additional security to existing protocols and methods of NAT traversal.
2. Determine if the added security allows for bi-directional communication across the IGD providing NAT services.
3. Ascertain the amount of CPU usage, memory usage, and the round-trip time of packets.

These objective and subsequent sub-objectives drive the variables under analysis in the study. Chapter 4 will further detail these different variables.

Theoretical Framework

Kumar (2019) suggests that constructing a system based on theories found in the literature shapes the research's theoretical framework. A loosely characterized framework guides the literature review. A review of a small amount of literature helps to understand the theories that directly or indirectly impact the research topic to create this loose framework. Their theme can sort these theories regarding the research topic to help from the literature review.

This study’s objectives revolve around introducing security mechanisms in NAT techniques that allow bi-directional communication between clients and servers. NAT is still an essential piece to many networks today (Zhang et al., 2016), but NAT’s initial implementation does not work bi-directionally. Single direction NAT limits its possibilities. Other implementations attempt to allow bi-directional traffic, such as STUN (Rosenberg et al., 2003). These implementations require a third-party server on the public Internet to assist in the creation of the session. This third-party server responds to requests from clients to inform them of their public networking settings.

A VPN is also a mechanism used to allow traffic to flow into a network that resides behind NAT. VPN’s often do have additional authentication methods. Tailscale is a product that allows the traversal of traffic into a network where forms of NAT may be enabled. A Wireguard VPN is the mechanism that allows access through the firewall into the network (Anderson, 2020). Before a user can initiate the session with the internal network, they must authenticate to the IGD or other device providing VPN services.

This study improved upon this by adding authentication methods similar to those of services that use a VPN but do not require the same pre-configuration for each new connection made. The new model does not require the third-party server on the public Internet to inform clients behind NAT regarding their external network settings. These new model modifications caused an increased overhead, just as other studies saw (Yang & Lei, 2016).

Definitions

Bi-Directional Traversal: Traversal of network traffic can be initiated from either side of an Internet Gateway Device.

External Entity: A system connected to the IGD through the Internet.

Internal Entity: A system that is on the inside network segment of the IGD.

Internet Gateway Device (IGD): A device hosting NAT services connecting a private network to the Internet.

Overhead: The measurement of additional Round-trip Time, CPU usage, and memory usage.

Private Address: An address as defined by RFC 1918 (Rekhter et al., 1996).

Private Network: A network consisting of private addresses from RFC 1918 that are not globally unique (Rekhter et al., 1996).

Round-trip Time: The time taken for a packet to reach its destination and return to its source.

Assumptions

The first assumption made is that the measurements taken were the cause of the researcher's change. This assumption results from the environment remaining unchanged during the tests before and after introducing the network's change. The use of a segregated network allows the researcher to limit non-essential traffic on the network.

This study used an open-source version of NAT to help make modifications without accessing closed source software. There are various assumptions when choosing open-source software. The first is that the software package's original creator implemented all of the protocols involved to their specifications. Software not written to that standard could have detrimental impacts on the outcome of this study. It would not be representative of a solution using protocols written correctly to the specifications.

Other studies creating new forms of NAT have used open-source projects to expect no issues (Yang & Lei, 2016).

The following assumption is that the tools used to measure performance will accurately measure the desired variables. Without accurate measurement, the study would not have produced a conclusive result. Discussion over the tools used to measure the desired variables occurs in the Instrumentation, Reliability, and Validity sections of Chapter 3.

Scope, Limitations, and Delimitations

Scope

This study measured the overhead of a new form of bi-directional NAT that has added authentication measures. When testing this solution, the traffic sent to the IGD was under the control of the researcher. Control of the traffic reduces the variability of what an IGD could encounter while connected to the Internet. Since the primary focus of this study is NAT, IPv4 is the only version under consideration. IPv6 allows every device to be given a globally unique IP address on the Internet and therefore does not explicitly require NAT.

Networks have variations in how they are implemented and maintained; therefore, designing a NAT version that would work for every network is a challenge. This study provides a version of NAT that applies to a few circumstances.

Large organizations’ enterprise networks and networks are outside this study’s scope due to their difference in technology used and needs.

Limitations

This study ran in an entirely virtual environment with all operating systems and software running on a hypervisor. A virtual environment allowed the researcher to experiment without the use of dedicated hardware per system. Using dedicated hardware in a different configuration could produce different results as any change in hardware specifications could. In some cases, the hardware could be built explicitly for the software running on it. In this configuration, the environment is virtual. It is difficult to account for all other processes running on each operating system and that they are the same from test to test in a virtualized environment. Chapter 3 will further discuss the nature of the virtualized environment.

The following limitation of this study is that the code developed will be created to work in a specific manner in a single operating system. Code developed only for one operating system means no variance in how it deploys to a given system. Testing and measuring the working code was wholly controlled. In a production environment, there could be many different devices that perform NAT services for a network. Each of the different devices that could run the service may run a similar service slightly differently. Different abilities to run the code means that using a different device to replicate this study may not have the same results.

The study was conducted in a virtually segregated environment away from all other systems and networks. Virtually separating the environment reduced the amount of what would be considered normal traffic that the IGD may process during normal operations from external sources. Attempting to place the device in a typical production environment for this study's proposed solution would cause it to encounter the traffic required by the experiment and the additional regular network traffic, traffic from malicious users on the Internet, and others. Being placed on a segregated network also eliminates the need for routers and other networking equipment between the IGD and servers in a configuration connected to the Internet. Removing intermediate networking devices not required for the experiment limited any other possible variables introduced by other devices.

Delimitations

One of this study's goals was to add a security layer to an existing implementation to create a final product in its entirety secure. Completing this study in a virtual environment eases reproducibility, but by doing so, reproducing the experiment could yield different results. Although replicating the systems under test from one virtual environment to another is possible, the environments themselves may not be the same. There are many possible differences in the virtual environment, such as differences in the configuration of the virtual appliances, underlying hardware running the hypervisors, and the load put on the hypervisors by other users during measurement times. Attempting to reproduce the experiment in a non-virtual environment may not result in the same outcome as the hardware's factors could be different from virtualized hardware.

Summary

Chapter 1 introduced this study beginning with its background and impact, then presented the problem and the study's purpose. The purpose was to study the impact of added authentication methods that rely on cryptography that allow bi-directional communication across an IGD that is conducting NAT traversal and if those authentication methods cause enough overhead for participating devices. It then followed up with sections diving into the study's significance and its nature, the research questions at hand, and the assumptions. Based on the study, its questions, and objectives, the best-suited research design is quasi-experimental before-and-after. Last discussed were the scope of the study, the limitations, and the delimitations.

The theoretical framework was also presented in this chapter and laid the outline for Chapter 2. Chapter 2 will present the literature review for the study and include a summary of NAT, its terminology, considerations, and a comprehensive review of NAT's variants. This section details the different features, security configurations, and deficiencies of the variants, following the investigation of NAT variants, a review of the literature involving the evaluation techniques of networking protocols regarding performance and their application to NAT in a network. The literature review will also provide ideal performance for a new version of NAT and the non-ideal performance concerning performance factors' impact. There will be a discussion on network security issues about network protocols and NAT along with performance factors.

CHAPTER 2: LITERATURE REVIEW

Chapter 1 began by introducing the topic of this study and its objectives. This study aimed to measure the change in resource overhead while changing NAT to allow for bi-directional traversal. This objective exhibited in Chapter 1 derives from the primary research question that inquires the impacts of additional authentication methods that rely on secure methods to allow bi-directional NAT traversal across a networking device. Chapter 2 adds depth to topics briefly mentioned during Chapter 1. Chapter 2 begins with an overview of the original NAT Request for Comments documentation, as this lays the groundwork for how NAT works and the reason for its development. Discussion following the NAT overview will cover the similarities and differences in solving the NAT traversal problem. After analyzing how other NAT operate methods, the conversation will transition into the overhead of adding NAT to allow for traversal across a networking device and how that overhead is measured. After discussing performance, the conversation will transition to network security and its role when considering NAT or new versions of NAT. Finally, Chapter 2 will examine the challenges of adopting a new version of NAT and adoptions of other NAT versions.

Internet Protocol Overview

According to RFC 791, the Internet Protocol created a system for interconnected networks that allows the sending of datagrams (Postel, 1981). These datagrams were to be sent from a source to a destination using addresses fixed in length. RFC 791 alone implements addressing and fragmentation of datagrams, and the Internet protocol treats each of these datagrams as an independent entity (Postel, 1981). RFC 791 is updated by many RFC's such as RFC1349, RFC2474, and RFC6864 (Postel, 1981). These updated

RFC’s update sections of the original Internet Protocol bring the Internet to what it is today, supporting a conglomerate of services to many consumers.

The Internet Protocol also described what a datagram, or packet, would look like as it traversed a given network. For this study, the header format is of significant value as NAT must modify it in some circumstances. The header’s notable contents regarding NAT are the version, protocol, source address, and destination address. These are typically more significant to IGD’s as they process datagrams through NAT rules.

As stated by RFC 791, data can split into multiple datagrams sent across networks that have limits on datagram sizes. IP treats each datagram as an entity unrelated to other datagrams (Postel, 1981). This management of entities will further complicate NAT during its development as it does not only have a single datagram per connection to handle. Upon reception of the data on the other side, the receiving host puts all data stored in the datagrams back together.

As discussed earlier, the introduction of IP did provide for systems to connect to the Internet but lacked foresight for the Internet’s upcoming growth. The addresses described in RFC 791 were only 32 bits in length, with the address beginning with a network number followed by a local address otherwise known as the host field (Postel, 1981). RFC 791 also described using classes A, B, and C as primary spaces for users to set public addresses. The introduction of RFC 791 did not propose private addresses or registries; both attempts at slowing the address exhaustion, so all addresses were considered public. It was not until the proposal of RFC 1918 that presented private addresses that could communicate on local area networks (LANs) (Rekhter et al., 1996).

NAT Overview

The primary technology under study is Network Address Translation. It is essential to evaluate the original writing of RFC 2663, IP Network Address Translator (NAT) Terminology and Considerations. This technology allows transparent routing to hosts that are behind an Internet Gateway Device. Transparent routing works by allowing the networking device or IGD to map one realm of addresses to another. When there is an initial connection attempt by a device to send traffic to a different network, generally, the traffic's first destination is the IGD. Once received by the IGD, that networking device may modify the IPv4 headers if it is needed. Not every NAT situation requires the headers to be modified. If the headers are modified, they are modified to reflect the source and destination address external to the original sender (Srisuresh & Holdrege, 1999).

Once the datagram is ready for transmission, it is then sent to the next destination, decided by the device's routing protocol and routing tables. From the perspective of the IGD, the traversal path is out of scope after the data leaves. This exclusion includes the routing protocols that determine the path the traffic takes to the destination and the networking configuration or NAT traversal implemented by the destination host. The destination host can reside behind a separate NAT as well as the source. However, since NAT is to perform transparent routing, that is unknown within the original networking device's scope. In TCP communication, some packets return from the original destination host. After packets have been sent outbound through the IGD, it may expect a response. The packets then return from their destination. They may have a new source address, which would have been the original destination address, and a new destination address of the IGD currently in scope. When the addresses change, it is also essential to identify the cascading effect of their change. The change of an address or port will also require the change of applicable checksums for the data. For example, a device that receives IP traffic will verify that the checksums are correct on each packet. If the checksum on a received packet is incorrect, it must be silently discarded (Braden, 1989). NAT's previous description is of RFC 2663 calls Traditional NAT or Outbound NAT (Srisuresh & Holdrege, 1999).

If the addresses were changed, this is considered a form of destination NAT. One of the limitations of this form of destination NAT is that it only has the capability for sessions to be initiated from one direction. However, it shows the translation process as they cross an IGD to travel to other networks. Another limitation in the first description of traditional NAT is that it does not describe a traffic translation mechanism to multiple hosts. The Network Address Port Translation (NAPT) section of the RFC describes a mechanism using ports (Srisuresh & Holdrege, 1999).

NAPT takes translation a step further by identifying a mechanism that allows multiple hosts on the internal realm to translate to one address in the external realm. NAPT uses identifiers from the packets that it receives while working in this mode and using them to track NAT mappings. The data used for identifiers depends on the type of traffic queued. The port number is the identifier for TCP and UDP traffic. For ICMP traffic, the ICMP query ID is the identifier. It is also possible to combine NAPT with other variations of NAT. For example, NAPT combined with outbound NAT allows multiple hosts on the private network to connect to external clients with a single external IP address on the IGD. This connection works by separating traffic sent by the internal hosts by port numbers assigned by the IGD with the session's creation's identifiers. This method of NAT is formally known as Traditional NAT and is outlined separately in RFC 3022, Traditional IP Network Address Translator (Traditional NAT) (Srisuresh & Egevang, 2001). NAPT's effectiveness in separating traffic is used through other implementations of NAT traversal such as STUN and TURN, Port Control Protocol, Complete Cone Symmetric Temporary NAT, among others (Cheshire, Boucadair, Penno, & Selkirk, 2013; Flores & Santisteban, 2017; Rosenberg, Mahy, Matthews, & Wing, 2008).

Srisuresh and Holdrege (1999) explain NAT's variation in RFC 2663, IP Network Address Translator (NAT) Terminology and Considerations, bi-directional or two-way NAT. This version of NAT allows hosts to initiate connections from the inside of the network, leaving the network and hosts to initiate connections from outside the network entering it. While this does allow for bi-directional communication, it is limited in its design. For bi-directional NAT to operate, it requires Domain Name System Application Layer Gateway (DNS-ALG). This DNS-ALG must allow DNS queries to traverse between the private and public realms. The IGD is required to host this service so that when an external entity wants to initiate a connection to the internal network, it must first perform a DNS request to get the FQDN of the internal device. Once the FQDN is available, the IGD can reply to the initial DNS query by the external device, sending traffic to the internal network device (Srisuresh & Holdrege, 1999). This mechanism of traversing NAT requires multiple prerequisite configurations before it can accomplish its goal of bi-directional communication. The first is that an internal host has an FQDN assigned to it. The second is that the DNS-ALG must reply to DNS requests for the

FQDN of the internal device, meaning that DNS queries must be allowed to traverse from the internal network to the external network and vice-versa. After those prerequisites, there is no consideration for security. The ALG will reply to an external device that can send the DNS query. Once the FQDN has returned to the external device, it can send requests to the internal client (Srisuresh & Holdrege, 1999).

The next variation of NAT described by RFC 2663 is Multihomed NAT. Multihomed NAT allows the network border to consist of more than one networking device. While having a single border device may ease the NAT process, it prevents network redundancy if the border device fails. Having multiple border devices presents additional concerns for address translation. For example, if an internal host were to initiate a connection to a host external to the network, the traffic would leave the network and translate through one of the border devices. The border device that the initial session traverses through will maintain the information for that session (Srisuresh & Holdrege, 1999).

Twice NAT is the final form of NAT described in the original RFC. Twice NAT is a form of NAT designed for when both the source and destination address a packet. The typical use for this method of NAT is when address spaces from both sides of the device that provide NAT overlap (Srisuresh & Holdrege, 1999).

According to the original RFC, NAT itself is an intensive process. When packets arrive at an IGD, each packet is subject to a NAT lookup. Even with a checksum involved to help speed up the lookup process, NAT is considered intensive (Srisuresh & Holdrege, 1999). After completing the lookup, the IGD can decide whether to forward the packet or drop the packet resulting in more cycles. Processing each packet as it arrives at the IGD requires the IGD to perform lookups and thus cause CPU cycles. The above descriptions and variations of NAT also show that to perform additional NAT-related operations, NAT's cost rises. While adding other features to NAT, these features may impact every packet that the IGD receives, thus slowing down the IGD's ability to process incoming data.

Security considerations from the original RFC discussed that a NAT router could become a target for attacks since they are Internet hosts (Srisuresh & Holdrege, 1999). When discussing NAT traversal and the devices that provide this service, security is a common thread. For instance, using a virtual private network (VPN) to connect through an Internet-connected device providing NAT services is implemented with security in mind (Deshmukh & Iyer, 2017).

In the original forms of NAT, some methods allow for different types of NAT traversal. These original methods outline several limitations and security concerns. These limitations and issues lead to developing other versions of NAT to supplement the originals. The upcoming section surveys additional implementations on how they solve NAT traversal and addresses the original design's limitations.

Survey of Existing NAT Solutions

Outside of traditional NAT and its uses described in RFC 2663, various NAT versions have been proposed and implemented. This section explores other models that allow traffic to flow from one domain to another. This study does not consider any NAT versions used as an intermediary between IP versions. There are many different types of NAT with different purposes: Configuration options ranging from those that require manual setup per instance to those that automatically work once configured.

Full Cone NAT – Static NAT

Technologies that allow traversal through an IGD are not always automated. Some require manual configuration. A common mechanism to allow inbound traffic through a networking device to an internal device is to use Full Cone NAT. This technique is also known as one-to-one NAT or manual port forwarding. This configuration allows connections to be initiated in either direction across an IGD based on the device's manual configuration (Cheshire & Krochmal, 2013).

The configuration of the networking device allows for connections initiated from external devices to the IGD. After establishing the connection to the IGD, the IGD acts as a proxy for those requests. The IGD will receive the inbound data and then perform NAT translation into the internal network. This form of NAT requires an external server to allow data to flow from the external network to the internal network but requires manual configuration. Once the manual configuration is complete, the mapping stays in place until manually removed.

Restricted Cone NAT

Restricted Cone NAT is a form of NAT where all requests from an internal client's IP address and port map to the same external IP address and port once a connection begins. Following the beginning of this connection, an external host can send a packet to the internal host using the same port. This external to internal connection requires that the internal host be the first to initiate the connection (Flores & Santisteban, 2017; Zhang et al., 2016).

Port Restricted Cone NAT

Port Restricted Cone NAT is similar to that of Restricted Cone NAT. It comes with the inclusion of restricting the port number. An external host could send a packet to the internal host only if the internal host had previously sent a packet to the external host. If the external host attempts to connect back to a different port, the connection will fail. The return connection must come from the same port that the original host sent. Using this form of NAT could cause issues for certain types of servers that receive connections on one port but may reply from a dynamic port (Flores & Santisteban, 2017).

Complete Cone Symmetric Temporary NAT

Complete Cone Symmetric Temporary NAT is a NAT traversal solution that allows two peers behind NAT to connect. The process starts with the first client reaching out to a relay server that does not reside behind NAT. The first client provides the relay server with the information required to initiate a connection, such as the public IP address and the first client's public port. Next, the second client will connect to the relay server with the same information. The email address of the peer that the client is attempting to connect to is another essential piece of information needed to connect. Once the relay server has received all the information it requires from both peers, it will send the information for the connection to each client. Finally, the clients will request the NAT mapping to be made by their local IGD so that communication can begin. Once communication is complete, both clients will request their respective IGD release the NAT mapping (Flores & Santisteban, 2017).

This type of NAT solution shares similar themes with other discussed NAT traversal schemes. The first is the use of an intermediary server that does not exist behind

NAT. A NAT configuration implementing STUN, TURN, and ICE also uses a third-party server to establish connections between two hosts that are behind IGDs (Keranen et al., 2018). One difference in this form of NAT traversal is using the email address to establish that two peers will connect.

TCP Hole Punching

TCP hole punching is another mechanism that allows for bi-directional NAT to occur. Hole punching has been used previously in peer-to-peer networking configurations (Ford, Kegel, & Srisuresh, 2009). There are multiple requirements for this to work in a given networking configuration and limitations of how the traversal mechanism will function.

Hole punching works by two hosts behind a NAT device attempting to connect to each via outbound TCP connections. Once the device sends the SYN packet for the TCP connection, the NAT device will have open external ports for the clients to connect. This feature is unique to this form of NAT traversal. There are various mechanisms in which the two clients may attempt to connect, including simultaneous TCP open and sequential hole punching (Ford et al., 2009).

A restriction to TCP hole punching is that after establishing the NAT mapping, the external entity must know the externally available port. This restriction creates limitations when using TCP hole punching. A limitation created by the restrictions is that if a particular application uses a designated port and one instance is already using the designated port, another instance may not have the ability to use it. Hole punching will not work with all forms of NAT as they all do not operate the same way. The form of NAT in place must be compliant with the restrictions and operating procedures defined above; otherwise, TCP hole punching will not work as intended. A variant of this NAT traversal mechanism is UDP hole punching. UDP hole punching works similarly to TCP hole punching but uses a rendezvous server external to both clients (Ford et al., 2009). Using a rendezvous server is also used in other forms of NAT traversal.

STUN and TURN

The following solution is a combination of separate protocols to create a complete solution for NAT traversal. Session Traversal Utilities for NAT (STUN) is another protocol that aids in network traversal. STUN is no longer considered a complete solution to NAT as it was in its original RFC. The original STUN design was a complete solution for NAT traversal (Rosenberg et al., 2003). An updated version is only a partial solution to network traversal, which requires multiple other protocols for a complete solution, such as the Interactive Connectivity Establishment (Rosenberg et al., 2008). ICE uses the STUN and its extension protocol Traversal Using Relays around NAT (TURN) for NAT traversal in primarily UDP-based communication. ICE has also changed to support TCP traffic to support a wider variety of applications and protocols (Keranen et al., 2018).

To begin communication with one another, two clients must first discover their networking configuration to choose an appropriate communication mechanism. The clients themselves are unaware of their network's possible NAT configuration due to NAT's transparent nature. The agents begin by connecting to a signaling server that resides on the public Internet. Once the connection establishes to the signaling server, it can determine public IP and port information from the traffic that it has received. Once each client residing behind NAT has connected to the signaling server, the server can distribute the information required to connect to the opposite client attempting to establish a connection (Rosenberg et al., 2008). A benefit to using this type of NAT traversal technique is that it works under many networks due to its flexibility from the discovery mechanism (Santos, Kantola, Beijar, & Leppaaho, 2013).

One issue with this method of NAT traversal is the extra burden it puts on a device. It requires extra code to run that is not related to the task the application or device performs. The device must continue to send keepalive messages to keep the NAT mapping alive. A third drawback to using this type of traversal is the possible delay in the session setup. During the initial phase of a connection, a device must wait until the first option has timed out before using the second method. This extra time in configuration may lower the system's quality and possibly may not be acceptable for the application that is implementing this form of NAT (Santos et al., 2013). A downfall of using a rendezvous server requires extra configuration and maintenance. The extra steps required to initiate a connection between two hosts that reside behind NAT using a rendezvous server on the public Internet introduces complexity for a single connection in both configuration and troubleshooting.

SIP and ICE

A solution presented by Yang and Lei in 2016 proposed that the combination of Session Initiation Protocol (SIP) and ICE (Yang & Lei, 2016). Their solution showed promise by allowing all clients to connect to the peer across NAT in different cases. The three cases under experiment were as follows: both peers located behind the same NAT, one peer located behind NAT, and the other located on the public Internet, and the third case presented both peers behind separate NAT's. The solution showed promise because all clients could connect (Yang & Lei, 2016) successfully.

Their research is also limited in two primary ways: if their solution fails, TURN servers support the new solution. Using TURN servers introduces significant overhead with heavy network traffic showing that performance was also an important consideration. The second limitation cited was that the study only used PCs during testing, which is limited considering the range of technology that could use a new form of NAT (Yang & Lei, 2016).

This solution relies on a third-party server on the open Internet to ensure that connections work correctly. It shares this similarity with many of the solutions presented in this chapter.

Virtual Private Networks

Another commonly used solution to allow traffic into a network through an IGD providing NAT services is a virtual private network (VPN). A VPN is a virtual network created on top of existing physical networks (Frankel, Hoffmann, Orebaugh, & Park, 2008). A VPN can create a tunnel between a client and a server. VPN's are different from previous models of NAT traversal because they do not allow traffic to cross an IGD, but they create an entire tunneled network to send all traffic.

One of the security features that a VPN provides is privacy. This privacy prevents users that may be in between the endpoints from viewing or changing packet data. Security is a common theme in protocols that transfer data over the Internet and require a layered approach with multiple security features to protect data. The next feature provided by a VPN is authentication. Authentication offers verification that a user makes the connection to the network with valid credentials to the network. Another feature is data integrity. Data integrity of traffic passing over a VPN verifies that no data modification has occurred during transmission. The final feature is that data sent over a VPN is not re-playable. Nonrepayable traffic means intermediate users cannot resend packets sent by a legitimate user (Deshmukh & Iyer, 2017). These security features upgrade from some of NAT's previous implementations where systems can send traffic through an IGD without being sent through encrypted means like an encrypted tunnel for security. The original NAT specification lacks any acknowledgment of encryption as a security mechanism (Srisuresh & Holdrege, 1999). Though, not all of these security features may be required in all situations when implementing NAT in a network.

An example of using a VPN to gain access across a boundary is a solution called Tailscale. Tailscale attempts to eliminate as much of the configuration as possible while still using a VPN solution to access an internal device from an external device. The Tailscale approach also requires a "Magic DNS" component that the administrator configures. This "Magic DNS" component acts similarly to the Private Realm Gateway explored below by creating DNS names for the internal devices ("Tailscale," 2021).

Port Control Protocol

Port Control Protocol (PCP) allows an application to flexibly manage IP addressing mappings and policies on NAT devices and firewalls on the local network (Cullen, Hartman, Zhang, & Reddy, 2015). To do this, PCP has two primary functions. The first is to allow packets to be received from the Internet and sent to a host on a network, and the second function is to reduce keepalive messages sent from a host to a server. Port Control Protocol is defined via RFC 6887, Port Control Protocol (PCP), and is designed for use when a Carrier-Grade NAT is in place outside of the network or within a small network. IPv6 transition scenarios also warrant the use of PCP. PCP is flexible in its uses as it is helpful in scenarios where the NAT mapping is short or long-lived (Cheshire et al., 2013).

During operation, a client sending a PCP message will send its request over UDP. PCP does not require a reliable protocol as every message sent will generate a response from the PCP server. This mechanism means that the PCP client is responsible for verifying that the PCP services its request. If a response is not received, the client will resend the PCP messages requesting a NAT mapping, thus making the protocol more resilient (Cheshire et al., 2013).

Since its inception, Port Control Protocol has evolved through updates. One update specified a form of authentication for the protocol. Allowing any host connected to the internal network to generate or delete port mappings can lead to security concerns. This method defines a mechanism that allows a PCP client to authenticate to a PCP server to securely modify, create, or delete inbound or outbound mappings (Cullen et al., 2015). Adding an authentication system to a mechanism that allows for NAT traversal is not entirely original. For example, using a VPN to traverse through an IGD can enforce authentication, but the original NAT design does not itself consider authentication (Frankel et al., 2008; Srisuresh & Holdrege, 1999).

The predecessor to PCP is the Port Mapping Protocol (PMP). PMP is laid out in RFC 6886 NAT Port Mapping Protocol (NAT-PMP) (Cheshire & Krochmal, 2013). Although Port Control Protocol has updated NAT-PMP, some networking software such as pfSense still supports NAT-PMP (Netgate, 2019). NAT-PMP is the basis for a protocol that allows the automation of port mappings and functionality to allow a client to gather information such as the external address of the network it is residing on (Cheshire

& Krochmal, 2013). PCP supports other NAT traversal mechanisms as well. Universal Plug and Play (UPnP) is a system that is embedded in an IGD or other NAT device and allows the transparent control of NAT (Boucadair et al., 2013).

Application Layer Gateways

Application Layer Gateways (ALG) are another mechanism to traverse NAT. ALG's are components of networking devices that help to route transparently. Not all application traffic easily adapts to using traditional NAT mechanisms. When an application places IP addresses or port information in the packet's payload, traditional IGD's will not interpret the information and correctly implement any port change (Srisuresh & Holdrege, 1999).

ALG's typically do not use any additional protocols to communicate with the IGD. The ALG will work directly with NAT to modify state information for the application traffic. The original NAT RFC mentioned DNS-ALG's as a mechanism to allow bi-directional traffic across an IGD. The DNS-ALG allows the traversal of DNS requests to internal network resources (Srisuresh & Holdrege, 1999).

A limitation of an ALG is that a given ALG only supports the specific applications and protocols configured to support it. Therefore, an application requires a specific ALG configured (Novo, 2018).

Private Realm Gateway

Private Realm Gateway (PRGW) is another network traversal technique proposed that does not rely on existing NAT but replaces it. PRGW aims to create a scalable model that can use a limited number of public addresses and equipment to support end-to-end communications with existing protocols. It also attempts to allow bi-directional communication of various protocols (Santos et al., 2013).

Internal hosts' outbound connections act very similarly to that of the original NAT specification and therefore do not receive as much attention as inbound connections. Inbound connections work by first performing a name resolution for the FQDN of the private host. Following the DNS query's reception, the PRGW will use a public IP address from its public address pool and uses that in the DNS response. The PRGW will then create a mapping that will receive data from the external host and forward the traffic to only the host that the original DNS request was made (Santos et al., 2013).

One similarity of this technique is its use of DNS names for the private hosts to multiple NAT traversal mechanisms such as Customer Edge Switching and the original description of bi-directional NAT (Kantola, 2010; Srisuresh & Holdrege, 1999). This form of NAT requires a single public IP address per resolution, representing a limitation (Santos et al., 2013).

Customer Edge Switching

Customer Edge Switching (CES) is a replacement for traditional NAT devices. This traversal form requires replacing the hardware device on the trust boundary where a traditional IGD would reside. CES solves the reachability problem by implementing a PRGW. This implementation allows hosts either on the Internet or on external private networks to initiate connections to a host on a separate private network (Amir, Goulart, & Kantola, 2016). This form of traversal works by publishing unique identity tags for users or applications. CES systems can create unique identity tags from unique names such as FQDNs (Kantola, 2010).

Upon creating a unique identification tag for a resource, a client may start communication with a DNS query routed through a CES system. A CES system contains an enhanced DNS proxy that allows it to reply to DNS requests for CES resources. CES then maps the identity tag to a local IP address and local MAC address. After gathering this information, the host sends the message to the CES with the IP address it previously received, and the CES will modify the packet to then forward on to the provider edge node (Kantola, 2010).

One way in which CES is unique from other solutions is in the way it allows hosts from one private network to communicate with hosts from another private network without having globally unique addresses. Similar to other forms of NAT, when an application requires sending address information in the packet's data section, an ALG supports that information by decoding the data section's information. The following way this form of traversal is unique is its ability to invalidate the unique addresses, thus preventing the permanent use of addresses (Kantola, 2010).

Summary

Each of the previous solutions sets out to solve the NAT traversal problem, also known as the reachability problem. Themes emerge from the previous solutions. One of which is that protocols that transmit host or protocol information in the packet's body introduce extra challenges to NAT traversal. The next is that additional hardware is often required to create a fully working NAT traversal mechanism. STUN, CES, and possibly even VPNs require additional software to successfully traverse NAT (Leppaaho, Beijar, Kantola, & Santos, 2013; Rosenberg et al., 2008). Another theme throughout multiple instantiations of NAT traversal mechanisms is using an FQDN to locate a resource behind a NAT device (Kantola, 2010; Santos et al., 2013; Srisuresh & Holdrege, 1999).

NAT Performance and Measurement

NAT can be an intensive process performing NAT translations on networking devices (Srisuresh & Holdrege, 1999). The performance of a device that is a single entry point to a network is crucial. If the NAT process overloads the device, it could cause the device to slow down and impact the performance of any traffic that is required to traverse NAT. Since NAT executes on a networking device on the edge of a network, it is the primary measurement point. The first measurement taken into account is the CPU cycles. These cycles are the underlying foundation of every action a device will produce. If a new version of NAT requires too many cycles, the processor could be overloaded and cause a queueing effect, thus slowing the device down (Novo, 2018).

The subsequent performance measurement is the round-trip time (RTT) of the packets traveling across the IGD. Round-trip time (RTT) is a measure of how long it takes data to move from endpoint A to endpoint B and a return acknowledgment from endpoint A (Zhao & Gao, 2015). RTT is a helpful measure when analyzing network performance because not all networks link speeds are symmetric—using the tool produced gave the researcher insight into RTT.

The third measurement is the memory usage on the networking device. The change in NAT traversal technique presented by Novo (2018) also takes note of this measurement when testing a new NAT traversal form. Novo (2018) explains that it is essential to understand the memory footprint that the software makes on the device that is translating data in a constrained environment.

A consideration but not a variable that will be measured is the configuration delay or the time it takes to begin communication. Some NAT traversal models, such as using STUN, TURN, and ICE to create a complete NAT solution, may take extra time to set up a connection (Santos et al., 2013). A significant delay in the initiation of a connection may prove to be unusable in many circumstances. When dealing with two hosts behind NAT devices, some of the models above could cause a significant delay. Future research using the created model could compare the relative configuration delay between dynamic models. However, a significant difference may not negatively impact the new model's overall performance.

Network Security

It is essential to consider network security when changing or adding services that could receive traffic from malicious users with rising security requirements. Under security considerations, RFC 2663 mentions that NAT devices are Internet hosts, which makes them a potential target of malicious attacks, and that a device running NAT should have protection to the same degree as that of any other server that resides on the Internet (Srisuresh & Holdrege, 1999).

There is a secondary consideration for network security in the case where NAT is in use. One of the primary functions of bi-directional NAT is to receive traffic from external devices to forward the traffic to the internal network. In most scenarios where the external network is the Internet, there is potential for unsolicited traffic from malicious users. Suppose malicious traffic was to be received and forwarded to the internal network. In that case, the malicious users send unsolicited traffic to devices on the internal network.

NAT itself creates a false sense of security when it comes to routing traffic on the Internet. When a private network is behind a networking device or IGD performing NAT, the internal addresses are not directly accessible from the external network. This lack of connectivity creates a reachability problem for the network (Santos et al., 2013). Since the internal hosts are not directly routable, there must be a NAT mapping for traffic to enter the network. NAT is, however, not considered to be firewall functionality. In the case of IPv6, this is not the same as in the definition of IPv6, where all addresses are globally routable (Deering & Hinden, 2017).

Opening a port to the Internet presents a risk as well. Upon creating a port mapping to allow the IGD to process traffic, there is a risk that it is exploitable. Once the functionality of an IGD is exploitable, attackers may have a way to gain further access to the network and circumvent NAT or possibly the firewall configuration. Even exposing functionality that would allow the modification of NAT rules internally to hosts is cause for concern. RFC 7652, Port Control Protocol (PCP) Authentication mechanism discusses this issue and provides a solution. The RFC states that not all hosts may be authorized to modify mapping information. Adding a form of in-band authentication to the Port Control Protocol gives refined security control over the ability to create or modify address and port mapping information (Cullen et al., 2015).

A method of exploiting NAT called “NAT Slipstreaming” has been discovered. NAT Slipstreaming allows a malicious actor to access remote ports on a system that is internal to a running IGD. The method allows the attacker to bypass the firewall and NAT system after the victim on the internal host visits a website (Kamkar, 2020).

With NAT previously described as a compute-intensive process, it is essential to consider the risk of denial of service (DoS) attacks on the device itself. If a port is open on the IGD for either the initial use of allowing a connection to an internal device or itself, the IGD will process packets that it receives. As NAT is already an intensive process, the additional overhead of new packets may impact the IGD's performance as it processes legitimate data. RFC 2663 mentions that NAT devices are Internet hosts, which makes them a potential target for multiple types of attacks and that they should have the same amount of protection that any other Internet-facing server would have (Srisuresh & Holdrege, 1999).

Although denial of service attacks can be very useful, techniques are available to mitigate the attacks. One such mitigation uses a technique to use the device's firewall functionality to block all transmissions from a sender. Denial of service attacks has shown to be very useful in the past and can scale to massive proportions that a simple firewall feature would not block (Etherington & Conger, 2016). Protecting against such an attack is outside the scope of this study.

Adopting NAT

Adoption of new technology is challenging, and adopting networking technology into existing networks is no exception. An example of this is IPv6. The first introduction of IPv6 was in 1995, with revisions published in 1998 and 2017 (Deering & Hinden, 2017). There are multiple reasons for the challenge of implementing new versions of any networking technology. One of those is that change is not required. Take, for example, IPv6. IPv4 is still available to many users worldwide thanks to RFC 1918 private addresses and NAT (Rekhter et al., 1996). Without an absolute requirement to change, it is difficult to force change. These challenges relate to creating a new version of NAT. Though various technologies and applications could benefit from a new version of NAT, other mechanisms are already in place.

Even though the software can be easily updated and, in some cases, even easily propagated, it can be challenging to deliver that technology out to the required devices. With a new NAT technology, IGD's would require an update even though they may not be updated often. Although some technology may be difficult to update, some have introduced more natural update mechanisms. An example of this is the software firewall pfSense. pfSense includes a mechanism to automatically update its software with the push of a button and very short downtime (Netgate, 2019). pfSense is only one example of this type of update system. Even though applications or users could benefit from a new version, it may be challenging to implement.

A difficulty with adopting newly developed technology is that it may not always have the intended effect even if implemented in a single place. Securing BGP traffic via IPv6 extension headers is an example of this (Ham, 2017). Even though a new technology might show promise to be beneficial, it must have broad adoption to have the intended effect. In this example, attacks against the BGP protocol continue to occur. A new version of NAT cannot be implemented on a single networking device at the edge of a network and improves traversal applications or users. Another example of this challenge is CES. Without deploying CES to multiple locations, a user or application cannot benefit (Kantola, 2010).

The applications and systems that are sending data across the IGD must also be capable of using a new form of NAT. Although some forms of NAT are automated and transparent, some others required manual configuration or intervention. This configuration could be dynamic and not require any modification by the user but could require the systems to make changes. NAT variations reviewed above demonstrated dynamic configuration. One such example being STUN, TURN, and ICE. In this combination, traversing NAT was not as transparent as in the standard NAT version defined by RFC 2663 (Santos et al., 2013; Srisuresh & Holdrege, 1999). ICE first must communicate with the various systems to determine their external addresses and then communicate that information back to the systems that are attempting to communicate across networks that have NAT implemented.

In summary, there are many roadblocks in implementing a new networking technology that is widely adopted. A new NAT must be both beneficial and easily adaptable to have a slight chance to be adopted into production. It is also easier to adopt if it is entirely transparent to the applications that send data across a NAT device.

Summary

Chapter 2 began by reviewing the original NAT RFC and its numerous options, modes of operation, and other considerations. This review was necessary to supply background information and compare and contrast the other NAT traversal models. After the initial NAT mechanism review, newer models and methods for solving the NAT traversal problem and the reachability problem were surveyed, demonstrating similarities and differences in their design and feature set. Various themes emerged in how each of the solutions could traverse NAT that often included using additional hardware. Next, a review of NAT performance variables occurred and how they affected other models of NAT. Finally, an overview of the challenges of adopting new technology forms that affect NAT models' production affects other networking and technology facets.

CHAPTER 3: RESEARCH METHODS

As Chapter 2 surveyed the literature around NAT, Chapter 3 will begin by presenting the research methods applied during this study. After the sections on method and design, the chapter will move into the research question, hypothesis, and variables that are the basis of the study. Exploring the population and sampling of that population will produce mixed results. Next, connections will be drawn between the data to be collected, its collection method using research instruments, and how those instruments can be considered valid and reliable. The final section of Chapter 3 will highlight data analysis execution and why the researcher chose those methods over others.

Research Method and Design Appropriateness

At the beginning of the research process, the research method is chosen and guided decisions throughout the process. This method comes from the perspective of the researcher on the study. Kumar (2019) states there are various perspectives that a researcher could maintain while performing research. The first is the application perspective. The application perspective splits into two categories: pure research and applied research. Pure research focuses on changing research methodology, techniques, tools, practices, methods, and others to assist other research types. The other category, applied research, applies data methods to be useful in other ways (Kumar, 2019). The application perspective does not lend itself to help the researcher meet the previously stated goals to measure the impact of a change on the network. The application perspective is a type of research for forming new research methodologies that can be further applied. The following perspective is the objectives perspective. There are varied study types within the objective’s perspective, such as the descriptive study, which attempts to describe a situation. Another study within the objective’s perspective is the correlational study, which attempts to identify a relationship between two aspects of a situation. An additional descriptive study within the objective’s perspective attempts to describe why and how there is a relationship between two aspects of a situation. The final type of research within the objective’s perspective is exploratory research. Exploratory research attempts to investigate an area that there is little known or previously researched. None of the prior studies within the perspective of the objectives attempt to numerically measure identified variables where a researcher modifies a study’s environment. Therefore the objectives perspective is not the best fit for this study based on the study’s objectives. Within the final perspective of research, mode of inquiry, multiple approaches could be used (Kumar, 2019).

The three approaches available to the researcher from the mode of inquiry perspective are quantitative, qualitative, or mixed methods (Kumar, 2019). The quantitative approach is structured, whereas the qualitative is an unstructured approach that allows more flexibility to the researcher. According to Kumar (2019), if the study's purpose is primarily to describe a situation based on measurable variables through nominal or ordinal scales, and if the analysis of that data finds the situation's variation without quantifying it, then the best research approach for the study is qualitative. If the research aims to measure a phenomenon's extent, then the quantitative approach will be used (Lazar, Feng, & Hochheiser, 2017). The study that was defined earlier proposed that the data gathered would measure the extent that devices on a network would be impacted by creating and implementing a new version of NAT with new features that would allow for authentication. In this quantitative approach, there must be a way to measure the described variables. Instruments were be used to measure the variables and will be described later in Chapter 3. Creswell and Creswell (2018) suggests using predetermined variables with a quantitative research approach.

Kumar (2019) explains three considerations when applying a research design to a quantitative study. The first of which is how many contacts the researcher has with the study population. The next is the reference period of study; this design focuses on events that have occurred in the past. Since this was a live experiment, a study design based on the reference period is inadequate for this study. The final consideration is the nature of the investigation and is best suited as the design for this research. The investigation's nature is the best-suited design for the study because the researcher introduced an intervention to the environment and observed the changes. Within study designs based on the investigation's nature, there are three options for a researcher: experimental, non-experimental, and quasi-experimental. The actions that compose this study are the researcher implementing new technology and introducing phenomena into a network. An experimental design starts from the cause of a relationship and intends to determine the effects. A non-experimental design is the opposite; it begins from the effects and attempts to determine the cause. The design is classified as a quasi-experimental design if both the experimental and non-experimental designs do not fit (Kumar, 2019).

Within the investigation study design, there are various models to choose from to implement the study. A quasi-experimental study has properties of both an experimental and non-experimental study. These two types of studies differ in the way that the cause and effect relationship is studied. In the experimental study, the relationship is studied, starting from the cause and establishing the effects. The non-experimental study is the reverse, starting from the effects and working back to the cause (Kumar, 2019). The researcher has decided to use a primarily experimental design instead of a primarily non-experimental study design because it allows the researcher's intervention to occur and then be studied.

The goal of this study was for the researcher to introduce a new method of bi-directional network address traversal that allows for authentication. The best suited experimental study design based on this information is the before and after experimental design (Kumar, 2019). There are many other study designs with an experimental nature that do not lend themselves to this study. For example, the after-only experimental design could work for this experiment, requiring the intervention already being in place and studied. The control group design bases itself on having multiple groups and using one as control and one as experimental. This experiment style does not lend itself to the control group design because there is no possibility for the control or the before data to change. There are also many design types, such as the comparative design, that do not fit well because they may have a different number of population groups.

During the study's execution, the researcher took an existing situation and added a modified version of NAT, resulting in a before and after experimental design study. This model fits the study best as it does not require the researcher to understand the situation before the addition to the network. The network can be observed and measured before introducing the new model of NAT and then re-measured through the same process after the intervention.

Research Question, Hypotheses, and Variables

The basis of the research question is: *What are the impacts of additional authentication methods that rely on cryptography that allow bi-directional communication across an IGD conducting NAT traversal, and do those authentication methods cause enough overhead to the end devices IGD to impose a negative impact on the network as a whole?* An attempt to answer this research question by implementing the new NAT traversal mechanism and overhead measurement.

Drawing a hypothesis from this research question is possible. The hypothesis is an assumption or assertion made by the researcher before conducting the research based on a situation and the researcher's observations. This assumption becomes the basis of an inquiry in a study (Kumar, 2019). This research study's hypothesis is: *A new method of NAT traversal implemented such that traffic can traverse an IGD bi-directionally with added authentication mechanisms to further secure traffic traversing in and out of a having minimal impact on the overhead of the IGD and the network.*

According to Creswell and Creswell (2018), a variable is a characteristic of something that varies and can be studied. Kumar (2019) suggests that a variable is measurable on a scale with varying precision levels. From the hypothesis above, the variables for this study emerge. The hypothesis states that the overhead is what is to be measured. The overhead of the IGD, in this case, can be further defined as the CPU usage, memory usage, and round-trip time of the packets sent from hosts on either end of the networking device. The CPU usage and memory usage were taken from the IGD as that is where a new version of NAT could have the most significant performance impact.

CPU usage might significantly impact performance due to additional actions requiring cryptography (Redzovic, Smiljanic, & Savic, 2017).

Population

The population of a study will provide the answers to the research question (Kumar, 2019). In this study, the research questions' answers will measure variables on systems running within an environment. Virtualization made these machines as similar as possible even though they ran different operating systems. All systems running on the same virtualization system will remove any variability from outside the operating systems (Rahman, Wang, Chen, & Jiang, 2018).

A hypervisor is software used to create, run, and manage virtual machines and be known as a Virtual Machine Manager (VMM) (Iqbal, Pattinson, & Kor, 2015; Timcenko, Djordjevic, Rakas, & Davidovic, 2014). The virtualization of this study used a hypervisor instead of systems running on bare metal. Running all systems on the same hypervisor allows for consistency between devices and is easier to test and replicate. There are different types of hypervisors, most notably Type-1 and Type-2 hypervisors. A Type-1 hypervisor being native and running on bare-metal versus a Type-2 hypervisor hosted on a system (Vojnak, Eordevic, Timcenko, & Strbac, 2019). The use of a Type-1 hypervisor eliminates the need for underlying software running the VMM. A Type-2 hypervisor would require an underlying operating system that could introduce unknown variables into the study.

This study's virtual machines that emulated the internal and external hosts will be running the Ubuntu Linux operating system. Ubuntu Linux is an open-source operating system that can run systems that range from a standard desktop running inside a network to a host providing services on the Internet to an Internet-connected device known as an IoT device (Canonical Ltd, 2016). This flexibility of an operating system lends itself to this research. An operating system with so many possible uses makes it an ideal candidate for this study. It limits the variance of using multiple operating systems for the study and is a reasonable emulation of which systems could run similar production software. The version of Ubuntu used in this system is 20.04.2. The Ubuntu system connected to the internal portion of the network, and the Ubuntu system connected to the network's external portion used the same version. Each machine had four cores of a processor, four gigabytes of memory, and 68 gigabytes of storage. Each machine's cores ran on a Type 1 hypervisor with a Xeon E5-2630 running at 2.30GHz.

The virtual machine that ran as the gateway device running the new version of NAT ran the pfSense operating system. pfSense is an operating system from a free and open-source firewall project based on FreeBSD and offers free third-party software packages (Netgate, 2018). The version of pfSense used was the latest stable build at the time of testing. The pfSense version number was 2.5.0. FreeBSD is an operating system based on the development of a large community. It is a platform for servers, desktops, and embedded systems used throughout the Internet ("FreeBSD," 2021). A free firewall operating system allows the researcher to conduct the study without purchasing software or hardware and easing replication for future research. The open-source operating systems and subsequent packages also allowed for more straightforward software modification and research options to introduce new functionality into the device. Netgate (2018) also claims that the pfSense firewall has become so popular that it has replaced many other brands of commercial firewalls in numerous installations worldwide. This popularity in production networks is another reason pfSense was an appropriate choice over other options, such as using a barebones Linux system running only NAT functionality and being open source and easily modifiable. PfSense ran with a single core on an Intel Xeon CPU E5-2630 at 2.30GHz, four gigabytes of memory, and 36 gigabytes of storage. As previously mentioned in Chapter 2, any form of widescale adoption can be a challenge, especially when it comes to networking. Two examples are the widescale adoption of Internet Protocol version 6 and Border Gateway Protocol (Beeharry & Nowbutsing, 2016; Ham, 2017). Using a free and open-source operating system found in commercial and non-commercial installations worldwide lends itself to be an option for research to aid in adoption speeds.

A single hypervisor hosted the entire experiment to limit the effects of traversing across a network to another instance of the hypervisor. Using a single hypervisor eliminated any effects that networking hardware could introduce. If the virtual machines used multiple hypervisors, additional overhead could be sent from one to the other. Also, the test hosts were the only hosts using the hypervisor at the measurement time. The hypervisor used was VMware ESXi 6.7.0 17167734 running on a Dell ProLiant DL360p Generation 8 server. The Dell ProLiant server ran an Intel Xeon CPU E5-2630 at 2.30GHz with 24 logical processors. Additional hosts using the hypervisor during measurement times could lead to inaccurate measurements due to unknown operations occurring on the hypervisor from other virtual machines. All of the previous efforts when setting up the experiment in a virtualized environment limit outside influences to produce the best results possible in a given situation.

Research Model and Design

The model created for this study added additional functionality to a host firewall that allowed traffic to traverse bi-directionally dynamically. The new software ran on pfSense and included a web application programming interface (API) that is reachable from any connected network. The model used the network shown in Figure 1.

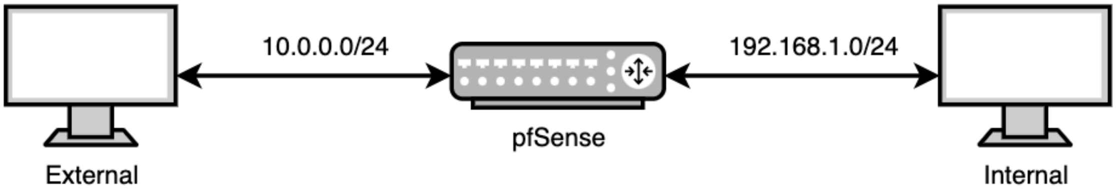


Figure 1. Network Diagram

The variables under study were measured at various points within Figure 1. For this study, vmstat measured both CPU and RAM on the pfSense system every half second. For round-trip time, the measurement was on the external host. However, this measurement required the external host to wait for a request to traverse the pfSense system and receive a response from the Internal host. While both tests produced different results, including different vmstat data, the measuring was done the same in both trials.

Typically, an external device cannot traverse an IGD to send a message to the Internal machine. In this case, pfSense was the IGD in use. This model allows a connection after the initial configuration period. Once connected to the network, the Internal device informs pfSense and the external system of its presence and generates a key. This connection is allowed through automatic outbound NAT. PfSense then stores the generated key for later when the External system attempts to open a port. The key and ID of the Internal system must be known. After the External system successfully creates an opening, it can then send messages to the Internal device. These messages could be anything from simple text to software updates. Once the communication completes, the External device tells pfSense to close the port and end the connection

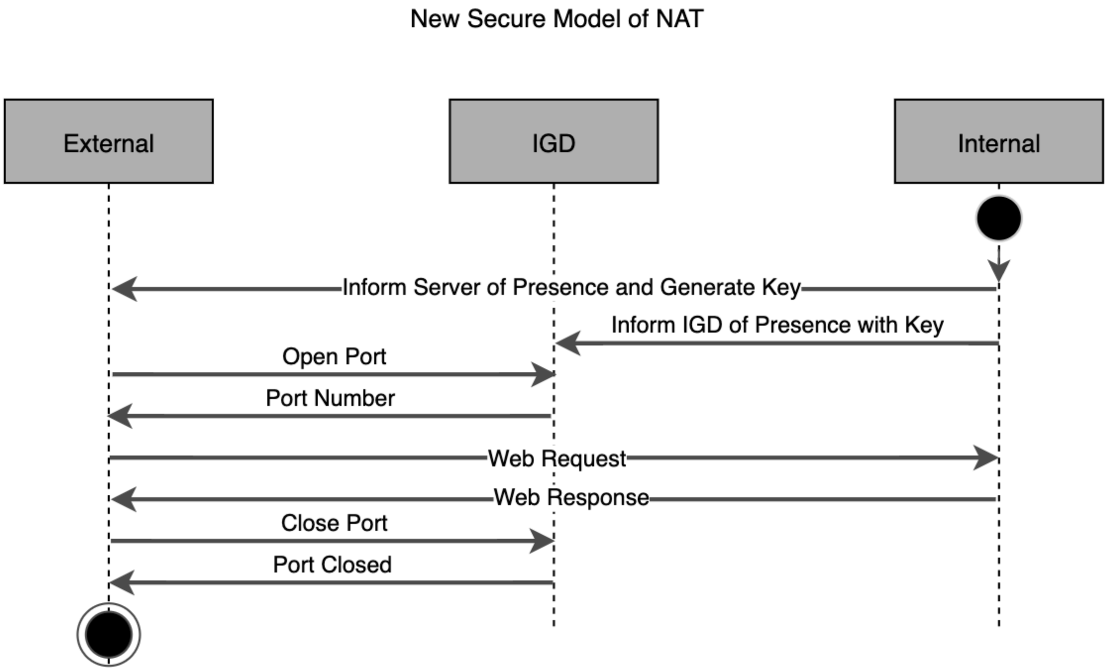


Figure 2. New Secure Model of NAT

Based on the proposed research model shown, the study itself is repeatable. Consideration for other hardware types is needed due to the challenging aspect of having the same hardware used in this study. This study is repeatable on other hardware and software, although the results may vary. One example of this is the modification of the processor in use. There are variants, such as using a processor with a higher base clock speed, allocating more cores of a processor, or providing more processors to the IGD to handle the new model more efficiently. The previous examples would likely close the gap between the CPU results of the port forwarding model and the proposed model but would require further testing. Another consideration is running the same trials on an IGD that contains an application-specific integrated circuit (ASIC) chip. These examples may lead to different CPU performance results and cannot be estimated based on this study alone.

Sampling Frame

Selecting a sample in quantitative studies aims to achieve the highest precision with the given sample size (Kumar, 2019). An important consideration when using sampling is to avoid bias while selecting a sample. Based on this study's nature, it is impossible to gather a sample of routers connected to the Internet to evaluate the theories presented. Due to this limitation, a non-random sampling design was the best choice for this study (Kumar, 2019). Within this category, there are many different types of non-random sampling available.

For this study, judgmental sampling was the most appropriate form of non-random sampling. Judgmental sampling allows the researcher to use their best judgment to decide on the sampling to best achieve the study's objectives. Because of the study's design, quota sampling was not available because the researcher does not have access to other routers actively using the Internet. Quota sampling allows the researcher to access the most convenient population until the population meets the sample size. Accidental and convenience sampling are not available because access to other routers is not an option. Accidental sampling has similarities to quota sampling in its convenience. However, instead of being guided by a visible characteristic, there is no guidance, and convenience sampling is only convenient to the researcher. Snowball sampling is not available because the population does not consist of an entity that can identify other entities for sampling. Snowball sampling provides a sample by creating networks from known entities (Kumar, 2019). For example, in a study using IGD’s, the researcher would identify IGD’s connected and sample those until the population reached the correct number of IGD’s.

Within judgmental sampling, there are multiple methods to consider. According to Ilker, Musa, and Alkassim (2017), they are maximum variation sampling, homogeneous sampling, typical case sampling, extreme or deviant case sampling, critical case sampling, total population sampling, and expert sampling. Using maximum variation sampling is not needed because the study will only study a limited number of variables. Homogeneous sampling is not ideal because of its focus on the similarity of candidates. Typical case sampling could be an option for this study, but identifying what is typical across various vendors would prove challenging. Extreme or deviant case sampling does not fit because it focuses on the exact opposite of a typical case and would not account for the variance in operating systems as it is not linear. Critical case sampling seems to be most aptly suited for this study because it allows the researcher to select a predetermined number of critical cases. The assumption that if the phenomenon can happen in the critical case, it can happen in other cases. Since much of this study based itself on standardized protocols, this added to reproducibility. Total population sampling is not available because the total population of routers connected to the Internet is not available to the researcher. The final sampling method is expert sampling. This method may work in the test but does not fit the critical case sampling method. Based on these considerations, this study's best-suited sampling method was a non-random judgmental, critical case method (Ilker, Musa, & Alkassim, 2016).

Data Collection

The researcher collected data from the virtual machines participating in the experiment. Data collection occurred using various operating system tools that are readily available. The data collected are the variables defined in the research question and the previously designated objectives. The first variable was the CPU usage of the networking device with the new version of NAT. The addition of code handling data produces additional CPU cycles during the execution of a new method of NAT. These extra cycles are measurable via operating system tools defined in instrumentation. The following variable collected was the memory usage of the networking device. Tables in memory keep track of NAT mappings. In this new NAT model, more data the mappings held more data, such as the mappings created by external hosts initiating connections to the internal hosts and authentication and authorization information. The last piece of data measured was the round-trip time of traffic. Measuring the round-trip time was essential and provided insight into how the new NAT model's intervention will affect users. Negligible additional CPU cycles and additional memory usage may be transparent to the user. This data is essential because it will show how the intervention impacts the traffic traversing speed of the IGD.

Guided by the research questions and the study's objectives, gathered data answered the research questions presented. Kumar (2019) suggests that the analysis of data should be appropriate for the study's readers. Based on the questions and objectives described earlier, the types of data gathered are primarily performance-based.

Performance of an IGD while using NAT is crucial because it can be the gateway to the public Internet for many devices. Significantly impacting an IGD's performance as the gateway for many devices could impose adverse effects on the network's overall performance, potentially resulting in an unusable network for users. It is essential to take this under consideration when making changes to the IGD. Indicators for how changes will impact the IGD include CPU and memory utilization. Suppose the new NAT's operational use discovered a significant impact after adding security mechanisms to include configuration changes and cryptography. In that case, using the secured implementation may not be feasible. The impact to the IGD is not the only consideration required with the changes proposed. Studying the round-trip time of the data provided insight into how the change might affect the end-user.

The data was collected from within the virtual machines themselves during the test. Once collected, the data was removed from the virtual machine and collocated with all tests' results for analysis.

Instrumentation

Instruments are the tools used to collect data during the data collection phase of the experiment. Collection occurred before and after the researcher intervenes. Collecting data pre-intervention and post-intervention causes the instruments to be used multiple times throughout the process, requiring them to be consistent (Lazar et al., 2017).

The instrument was the code created to implement the new variation of NAT itself to collect round-trip time data. Other tools such as PING have been used in the past to measure round-trip time data (Kaup et al., 2015). Vmstat measured the CPU cycles of both user and system time and the amount of memory used during the test. Vmstat reports specific kernel statistics regarding processes, memory, disk usage, CPU usage, and others (Ham, 2017).

This study was a before and after study requiring observing variables before and after the researcher's intervention. The instruments used in this study were required to observe the variables cited in the research questions and the sub-objectives. Considering the established use of the tools as mentioned above, no pilot testing of the tools occurred. The study's objective is to measure the change after the intervention; the instruments do not need to predict the change. Therefore, predictive validity was not a good fit for this study (Kumar, 2019).

Validity and Reliability

Validity

Validity is the concept of having a situation where the instruments measure what they are supposed to measure according to the study's objectives (Kumar, 2019). Creswell and Creswell (2018) and Kumar (2019) state three different validity forms for an instrument. The first is the content validity, better described as whether or not the instruments correct information (Creswell & Creswell, 2018). Suppose an instrument used during this quantitative test was to incorrectly measure a variable directly related to the hypothesis or problem statement. In that case, the conclusions drawn from that information could be incorrect. The second form of validity is predictive or concurrent validity. In other words, do the results correlate with other results, and do they predict a criterion measure. Predictive validity measures how well a research instrument can forecast an outcome (Kumar, 2019). Since the study's nature measures the change after the intervention in the environment, predictive validity is not a valid form of validity test. Constructed validity is the third form of validity. The construct validity determines whether or not the items construct hypothetical constructs or concepts (Creswell &

Creswell, 2018). The most aptly applied to this study was the content validity for each instrument used during testing with these three validity forms.

Kumar (2019) suggests two approaches used in finding the validity of an instrument in quantitative testing. The first is to establish a logical link between a study's objectives and the research questions used in the instrument. (Ware & Frédérick, n.d.). Since the tool directly measures the variables under study, it creates a logical link to the research problem variables. The third variable under study, round-trip time, is also directly measured by the tool created to implement the secure version of NAT that measured round-trip time, creating a logical link. A logical link for the tools in use for measuring the variables under study produces valid tools, according to Kumar (2019). Using the tool created also uses the Linux “time” tool. The logical link created was the output of time used and the need for measuring time from the start of the script until the end of the script. The logical link between the output of the tool and the required data provides a valid tool (Kumar, 2019).

Reliability

Crewswell (2018) and Kumar (2019) define reliability as the consistency or repeatability of an instrument (Creswell & Creswell, 2018; Kumar, 2019). Reliability comes in two forms, internal reliability, and external reliability. Determining the reliability of an instrument was essential, as well as determining its validity. If a research tool is consistent and stable, providing predictable and accurate results can be reliable. Given this information, a researcher's view on an instrument has two different perspectives: how reliable it is and how unreliable it is (Kumar, 2019). Due to the study's technical nature, some of the factors or reliability listed by Kumar (2019) do not apply.

Cresswell (2018) says that results from past use of an instrument demonstrate acceptable reliability. Ham (2017) used vmstat to measure kernel statistics on a pfSense router similar to the one used in this study (Ham, 2017). This previous use of vmstat to measure CPU performance and RAM utilization demonstrates acceptable reliability for the instrument.

The proposed system does not have proven reliability to measure round-trip time, such as in the previous case of testing like vmstat. Kumar (2019) explains: to establish the reliability of an instrument; there are various methods. One such method is the test and retest method. This method takes the results from an instrument administered twice and compares the first results to the second results. The difference between the two tests indicates the reliability of the instrument. There are advantages and disadvantages to this type of procedure. Comparing the instrument against itself is one of the significant advantages of this procedure. Comparing the results of one tool to another could create inconsistencies (Kumar, 2019). Kumar (2019) lists multiple disadvantages of this method. However, many do not apply to this particular instrument as it measured a technical procedure and not attitudinal data. Another disadvantage of this is that the first results may impact the second set of results in tests where subjects have the memory of the first test (Kumar, 2019). The instrument's implementation attempted to negate previous runs by removing any records created and allowing the state table to reset before rerunning the test. Appendix D shows the test/retest method results to verify the reliability. The test ran using a request number of ten thousand. The second test ran using the same number of requests and returned an almost identical number. The first test resulted in an average of 1.8437 seconds, while the second test returned an average of

1.8504 seconds. The testing of the created product was combined with the Linux “time” tool to measure the amount of time a script took to run. Kumar (2019) says that the smaller the test and retest difference is, the higher the reliability of the instrument. These test results show that the average response is accurate to the tenth of a second after running two tests of ten thousand requests. The hundredth of a second was off by one in the average time of the two sets of ten thousand requests.

Based on the previous arguments made, results derived from testing should both be valid and reliable. These results will then input into the data analysis phase. After analysis, observations will be made in the following chapter.

Data Analysis

Once testing is complete, data was removed from the systems and collected to a central location. After collection, the data was cleaned and categorized for further analysis. Creswell and Creswell (2018) describe that the researcher should report the descriptive statistics and indicate the inferential statistics. The descriptive statistics include means and standard deviations, which will apply to the variables under observation listed as the CPU consumption, memory consumption, and round-trip time of the data sent (Creswell & Creswell, 2018). The CPU consumption and memory consumption statistics come from the IGD in the testing network. The round-trip time was the time taken for the external machine to create the port opening, send a message, receive a response, and tear down the port opening. This process was every step that an external entity would go through to get a message and response from a system internal to the IGD.

Chapter 4 will detail the data analysis, which will allow for testing of the hypothesis. Conclusions can then be drawn based on the analysis of data and the hypothesis tested.

Summary

Chapter 3 started by diving deeper into the design and method of the study. Chapter 3 reiterated the research question, the hypothesis, and the variables outlined by the hypothesis. Chapter 3 also described the population, sampling frame, data collection, instrumentation, viability, reliability, and data analysis. All of which provides a plan of how data will be collected and further analyzed in Chapter4. Chapter 4 will examine the study results and expand on the measurements taken and apply them to the study.

CHAPTER 4: RESULTS

The purpose of this quasi-experimental before-and-after study was to measure the impact that a secure model of NAT could have on a network. NAT's secure model would contain features such as sending data across an IGD through NAT in a bi-directional sense. The performance metrics measured were the CPU usage and memory usage of the IGD, the modified version of pfSense. The round-trip time measured comes from the external system using custom Python code to generate requests to the internal device. The requests sent across were Hypertext Transfer Protocol Secure (HTTPS) messages with a small amount of text in them. Kumar (2019) describes the factors affecting the inferences drawn from a sample to be the sample size and the extent of variation in the sampling population. First, the sample size, findings based on larger sample sizes have more certainty than those with small sample sizes. The second is the variation in the sampling population (Kumar, 2019). These factors affected the study in how large the population based on the variation. The purposive, non-random sampling of this study led to the researcher choosing a predetermined number of tests executed for the quantitative study. Each test ran one thousand requests, and the test ran three times.

The rest of Chapter 4 will describe these measurements in terms of the environment and their results.

Data Collection

As described in Chapter 3, multiple devices were running across different network segments in a virtualized environment. These devices ran a series of tests to create port openings and send a message through an IGD to another device inside the network. As described in the previous chapter, the testing was done in an isolated environment to eliminate any external factors affecting the testing results.

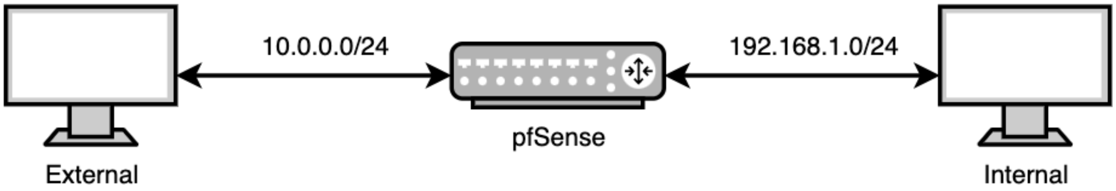


Figure 3. Network Diagram

Figure 3 shows a simplistic network diagram of the devices used to perform the tests. The vmstat instrument ran every half a second during the test to collect the CPU and RAM usage on the IGD, otherwise known as pfSense. This time interval was deemed acceptable in previous studies using pfSense as an IGD while collecting CPU and RAM usage (Ham, 2017). Once the data was collected and downloaded from the modeled network, tools removed the extra data not necessary for the results to provide a clean working data set. This data includes header information and additional data produced by the tools used.

Vmstat provided measurements of CPU and RAM on the IGD while the connections were taking place. As previously mentioned, vmstat ran on half-second intervals only while the connections were taking place. Once all one thousand connections concluded, vmstat stopped tracking the results. One significant difference in the results between the two versions of the test was the number of times that vmstat ran. In the test that used the port forwarding model compared against the new model of NAT, vmstat ran significantly fewer iterations in the former than the latter. The explanation for this lies in the differences in the tests. Since one test used connections that took significantly longer to run, the Vmstat tool ran more times than the shorter connection test. The way that vmstat ran was through a script that relied on having SSH access to the IGD. When the script started, it started running vmstat, and it started another script written in Python on the external host. The Python script ran for the prescribed number of iterations making connections to the internal host. Once the Python script completed the prescribed amount of iterations, it would stop the vmstat tool on the IGD. Vmstat output to a file on the local machine and was then downloaded after the test.

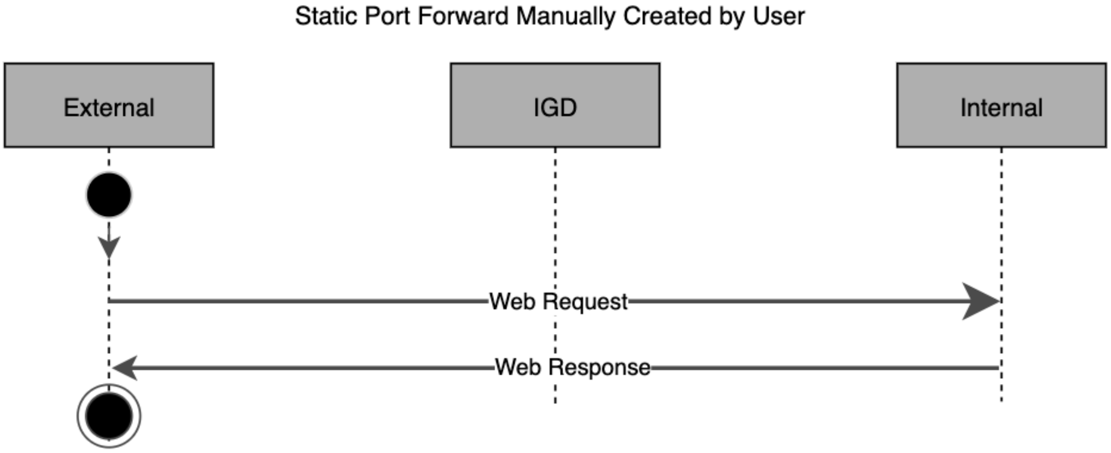


Figure 4. Port Forward Manually Created by User

The round trip time (RTT) of the messages was measured concurrently with the CPU and RAM measurements. RTT measurements resulted from tracking the time it took to send a message from the external device through the IGD to the internal device and back. Inside of the run script previously mentioned, the Python script ran through commands. For the original test and the test that included the researcher's intervention, different Python scripts existed. The researcher created a rule for port forwarding and NAT translation for the first set of tests done without NAT's newly created model. The Python script used to create messages for these tests only sent a message through the forwarded port and then received the response. In the test with the researcher's intervention, the Python script sent a message through a port forward and dynamically created the firewall rule and forwarding rule that allowed it to get traffic through. Once the script received a response from the internal host, the Python script also closed the dynamically opened port and then stopped the timing. The reader should consider this difference while looking at the results, statistical findings, and interpretations.

The sample sizes varied per measurement but were all based on having one thousand requests made across the IGD. Meaning that while the RTT times will have an equal number of samples for each test, the results from vmstat (CPU and RAM utilization will not) The averages from the CPU and RAM utilization numbers were running. Since the new model of NAT increased the number of actions it performed in its script, its times measured were longer than its counterpart, meaning that there are more vmstat measurements since vmstat still ran every half second.

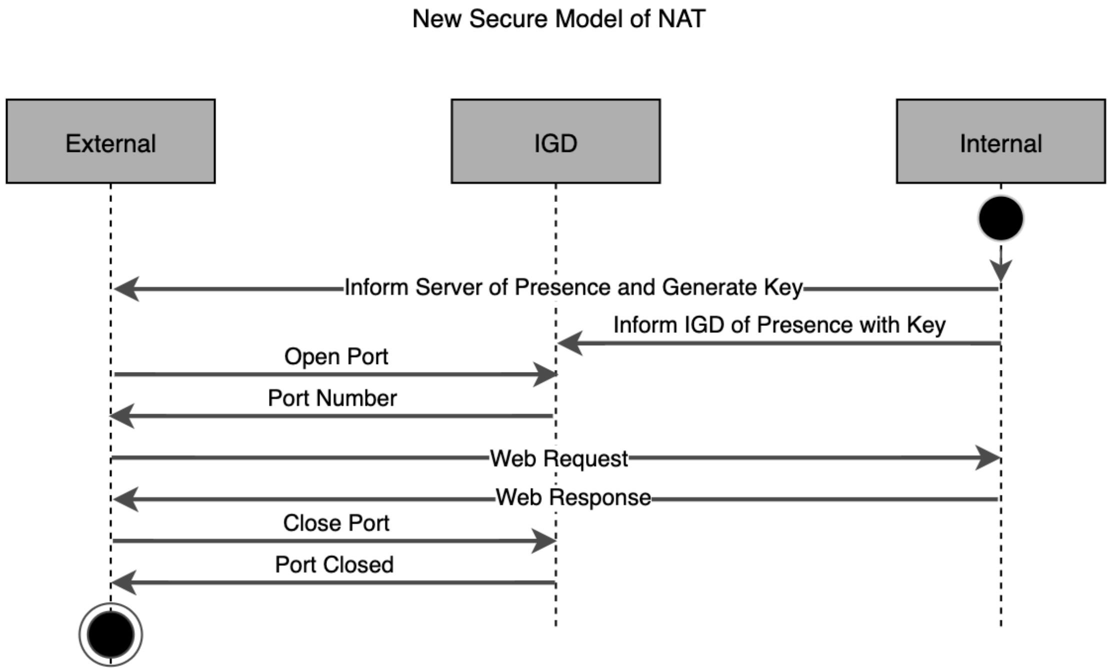

New Secure Model of NAT

Figure 5. New Secure Model of NAT

In conclusion, the new model of NAT took more actions on the IGD than that of a manual port forward. This increase in actions is by design as the new model of NAT is dynamic and should be viewed through that lens.

Results

A researcher interprets data from a quantitative study (Creswell & Creswell, 2018). This statement means that the researcher draws conclusions from the results and applies them to the research questions and hypothesis. This application leads to the more significant meaning of the results as a whole. Chapter 5 contains this meaning and the interpretation of the data. Statistically speaking, the researcher's intervention added a considerable amount of overhead in the test environment.

The results of the tests were conclusive that the models do have different costs associated with them. The average CPU time in the original model averaged slightly above 1%, and the new secure model of NAT was roughly 74% during the tests. RAM utilization was much closer as the averages difference was insignificant. The RTT average difference's significance depends on the application and will be discussed further in Chapter 5.

As previously mentioned, the script with the researcher's intervention took more actions on the IGD than the script that sent web requests through the static port forward. This difference is of note when reading the descriptive observations of each of the following measurements.

Descriptive Observations: CPU Performance

CPU performance was the first variable studied. CPU performance measured as the user and system percentage of usage. Vmstat recorded this data running on 0.5-

second intervals. Appendix B displays the averages of the tests and comparisons. Vmstat outputs the user time, system time, and idle time as percentages of the total process time (Ware & Frédérick, n.d.). Combining the user and system time leaves only idle time and creates the utilization percentage of the device.

An IGD only port forwarding requests show an average of 1.44% used. After adding the researcher's intervention, the average was 75.68% processor time. This change is a significant increase in processor time used by the IGD. This change demonstrates that the additional functionality of opening the port and closing the port by using a web server on the IGD creates significant overhead. Reloading the filter, running a webserver, and accepting cryptographically secure communications on the IGD are all additions made in the new model of NAT. The combination of these processes causes a significant amount of overhead to operate the model.

Descriptive Observations: Memory Utilization

Memory or RAM was the following variable under analysis during the trials. Vmstat reported RAM measurements on usage. Similar to the CPU measurements, vmstat ran at 0.5-second intervals measuring memory utilization. The RAM and CPU variables come from the execution of vmstat. Chapter 5 holds a more detailed chart of these results.

An IGD only port forwarding requests shows an average of 734.84KB of memory consumed while in use. The new model of NAT averages 839.27KB averaged across the three tests. This additional use is a mild uptick in the amount of memory consumed on the IGD relative to the amount of memory available. These numbers show that the new model consumes slightly more memory, but the memory does not cause NAT's new model to become infeasible to run.

Descriptive Observations: Round-Trip Time Analysis

The round-trip time is the only variable that does not come from the vmstat output. The round-trip time was used by measuring the systems clock and the time Unix tool. This measurement is using real-time measurement versus monotonic. Since a program ran instead of a single command, the time command was deemed the best fit. As explained above, for the test that relied on manual port forwarding, the script executed fewer actions than the script that executed NAT's new model. Appendix B serves as a more descriptive comparison of the tests.

The average RTT of a message while using static port forwarding across three tests of one thousand requests shows a result of 0.268 seconds. The RTT of the new model of NAT shows an average of 1.398 seconds. This difference shows a decrease in speed by over five times the difference. This amount may or may not be significant to a user of the system.

Statistical Analysis

A researcher concludes the study results to attempt to answer the research questions and validate the hypothesis. The statistical tests hope to determine that the results or observed scores reflect a pattern rather than chance (Creswell & Creswell, 2018). Kumar (2019) says that statistics have a primary function to act as a test to confirm or contradict the conclusions drawn based on the data at hand. Kumar (2019) further describes that the first step in processing data is to ensure that it is clean and free of inconsistencies (Kumar, 2019). The raw data collected from the systems were cleaned manually by removing any information other than the statistical analysis data.

According to Lazar, Feng, Hochheiser (2017), the hypothesis is the foundation of an experiment and the basis of statistical significance testing (Lazar et al., 2017). The hypothesis was stated in Chapter 3 as *A new method of NAT traversal implemented such that traffic can traverse an IGD bi-directionally with added authentication mechanisms to further secure traffic traversing in and out of a having minimal impact on the overhead of the IGD and the network.* In the experiment, the null hypothesis would be that there is no difference between the two models. If the models returned identical results, the null hypothesis could be accepted; it otherwise is rejected, stating that there is statistical evidence to support the difference in results (Lazar et al., 2017).

Identifying a Method to Demonstrate Statistical Significance

There was not a method proposed to determine statistical significance before this point. Experimental research allows for identifying relationships of events by observing dependent variables and control of independent variables (Lazar et al., 2017). When there is a group comparison, and the test yields a comparison of two groups in terms of outcomes, the statistical test used is a t-test (Creswell & Creswell, 2018).

The variance of the samples was determined using an F-Test. The results of the F-Test showed that the two samples had unequal variances in each of the categories. These F-tests led to using a two-sample assuming unequal variances t-test. The CPU tests' variances showed that the new model had an average of 406.39098, while the variance for the port forwarding tests resulted from 14.76070. The RAM also showed significant differences with the new model's variance at 1853354560.21 and the port forwarding model at 4795879.42. Lastly, the variances for the round-trip time were 0.0048829, while the port forwarding model showed 0.0021332. The differences in these variances suggest that the averages are different and that a t-test using unequal variances should calculate statistical significance.

Calculation and Evaluation of Statistical Significance

According to Lazar, Feng, and Hochheiser (2017), almost all experimental studies use significance tests. Without significance tests, it is possible to misinterpret the results of a given study. When conducting these tests, a commonly used P-Value is 0.05, or the probability of making a Type 1 error; using the 0.05 value limits Type 1 errors. A Type 1 error is when the null hypothesis is rejected, and it should not be; otherwise known as a false positive (Lazar et al., 2017). When a study contains two related samples, a commonly used test to compare the samples' means is the t-test (Boslaugh, 2013; Lazar et al., 2017).

Microsoft Excel calculated the t-test data results. For RTT, the arrays used for variables were the 3,000 total messages sent for each test. A t critical two-tail value of 1.96042. Additionally, the p-value resulted in zero, significantly less than that of the alpha level of 0.05 used. Therefore, the null hypothesis is rejected, and the difference between sample means shows the increase in RTT while using the new secure model of NAT.

Summary

Chapter 4 presented the quantitative results generated by the study. The first set of results derived from the study included an unmodified system using port forwarding on an IGD to allow traffic to pass into a network from an external system and return. The second set resulted from tests on the modified system, which involved creating the port forward dynamically, sending the traffic, and closing the port. Chapter 5 will continue by interpreting the study's data and relating it to the research questions and objectives.

CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS

This chapter summarizes the numeric findings presented in Chapter 4. The following metrics are of primary concern as they relate directly to the research questions and hypothesis: CPU performance, RAM utilization, and a message's round-trip time. Limitations will also be discussed, along with recommendations for future research.

Limitations

The researcher made attempts to eliminate any outside variables from affecting the study while taking measurements. As mentioned in Chapter 3, a Type 1 hypervisor was in use, and the researcher was the sole user of the hypervisor without any other systems running. It is still possible that the underlying hypervisor introduced a random variable into the environment while measuring, such as a process that does not run consistently, only occasionally.

The following limitation was the keys generation. For testing the proof of concept, the keys used to authenticate the external device to the IGD to allow the messages into the network were hardcoded, meaning that they were static. Although generating keys were in place, the proof of concept used static keys. Moving to a more production-ready version would require using the dynamically generated keys to ensure that the IGD was secure.

The third limitation was the data sent. In a production environment, the new model would allow anything from messages to software or firmware updates to be sent from the external system and received by the internal system. While testing both models, the script sent only a tiny text string as data inside the web request. This minimal data means that there were not likely multiple packets sent for each message. Further testing with more extensive data could slightly alter the results produced.

Finally, the skills of the research may have imposed restrictions on the study and caused validity issues. The researcher may not have had the skills required to develop such a package to create consistent and valid results. The final assumption is that the researcher did possess the technical and research skills to answer the research questions.

Findings and Interpretations

Chapter 2 displayed many different frameworks and tools attempting to ease the traffic traversal burden across IGD's. This study attempted to solve that problem as well. The following findings are described within the lens of the research question:

> What are the impacts of additional authentication methods that rely on cryptography that allow bi-directional communication across an IGD conducting NAT traversal, and do those authentication methods cause enough overhead to the end devices IGD to impose a negative impact on the network as a whole?

They also must be viewed with the sub-objectives in mind. The following sub-objectives support the primary research question.

1. Determine the extent of additional security to existing protocols and methods of NAT traversal.
2. Determine if the added security allows for bi-directional communication across the IGD providing NAT services.
3. Ascertain the amount of CPU usage, memory usage, and the round-trip time of packets.

These objective and subsequent sub-objectives drove the variables under analysis. First, addressing the study's sub-objectives, security features were added to existing protocols to allow NAT traversal. HTTPS was the method of transport in all communications. Once a device connected to the Internal network, it generated a key from the server to give to the IGD and its address so that when the external device is needed to connect to the internal device, it could use that key. Once the IGD received the request to open a port, it would only allow traffic to the correct host if the key was correct. If the key were incorrect, the port mapping would not be issued. This usage of a key to allow a port mapping proves that the second sub-objective is also possible.

The following sections answer the third sub-objective with interpretations of that data.

CPU Performance

The CPU performance was a primary variable to the hypothesis and research question. The data used for averages came from three separate tests using one thousand requests per test, totaling three thousand requests for each model. Comparing the static port forwarding test versus the new NAT model tests showed a massive increase in CPU usage. The difference between each model's best tests was roughly 74%, noting that both models were running on a virtualized IGD with a single core of a processor. The model compared against used slightly over 1% of the CPU, while the new model averaged consistently between 74% and 75%.

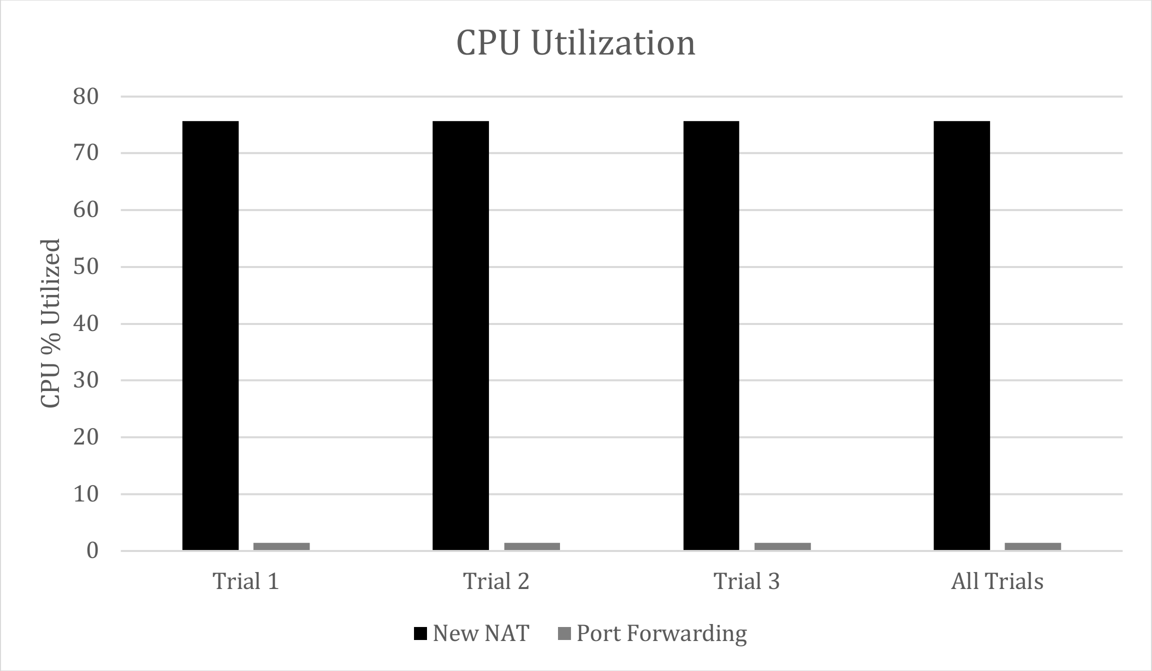


Figure 6. CPU Utilization

T-tests evaluated the statistical significance of the findings and showed a strong indication of statistical significance. The observed increase in CPU time could be related to multiple different factors of the new model of NAT. Reloading the filter, running a webserver, and accepting cryptographically secure communications on the IGD are all additions made in the new model of NAT. The combination of these processes causes a significant amount of overhead to operate the model.

Memory Utilization

The following primary variable was the memory utilization or “RAM.” This data came from the same vmstat output as the previous CPU results. It was again, taking measurements at half-second intervals for the duration of the one thousand requests sent. Comparing the two models showed that NAT's new model did consume more RAM than the previous tests, although the difference was not as significant as the CPU measurements' results. An IGD only port forwarding requests shows an average of

734.84KB of memory consumed while in use. The new model of NAT averages 839.27KB averaged across the three tests. T-tests evaluated the statistical significance of the findings and showed a strong indication of statistical significance. Running an additional web server with a database for the dynamic ports opened accounts for the slight additional use in memory

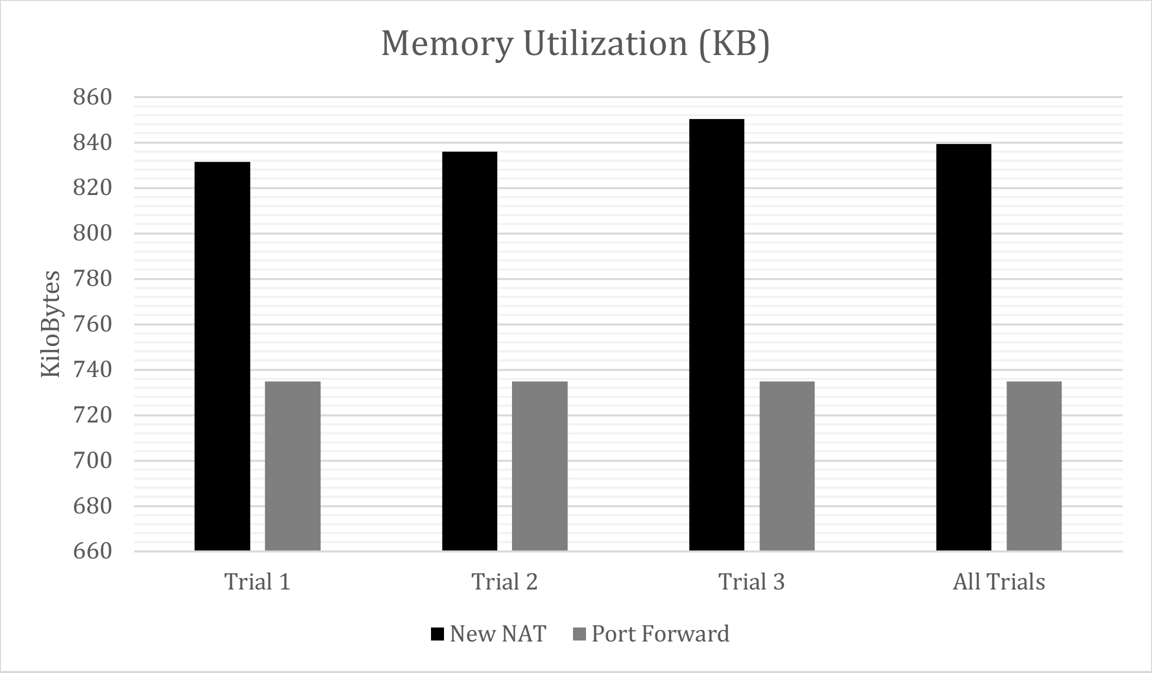


Figure 7. Memory Utilization

Round-Trip Time

The RTT of the messages was measured separately from CPU and RAM utilization. The external entity ran a Python script that timed to find how long it took for the message to return from the internal device. For each trial, the scripts sent one thousand requests. After completing all three trials and averaging all of the results, the port forwarding model had an average of .268 seconds. The new model of secure NAT had an average of 1.398 seconds.

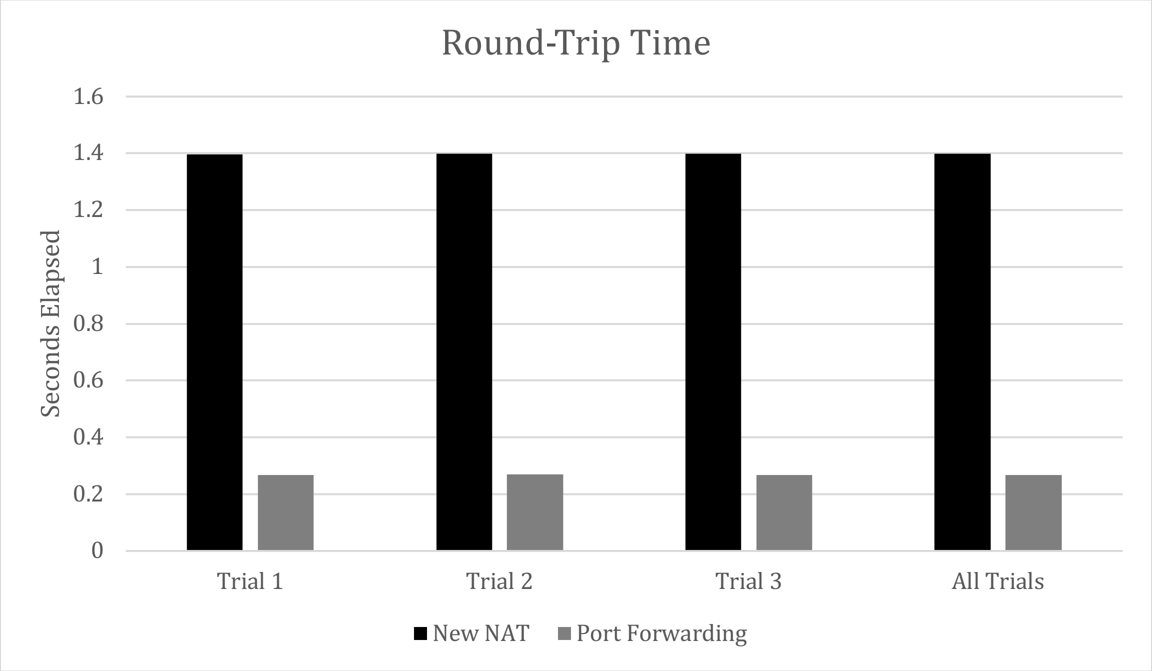


Figure 8. Round Trip Times

T-tests evaluated the statistical significance of the findings and showed a strong indication of statistical significance. The additional time found in the new NAT model resulted from the additional overhead created by sending multiple messages to the IGD and the internal device. The new NAT model also had to wait for the IGD filter to reload on each dynamic opening or closing.

Recommendations

NAT has come to be commonplace in many networks, as shown by the literature presented in Chapter 2. Implementing NAT adds challenges to traverse networks in specific scenarios. Only specific scenarios as there are still implementations where NAT is transparent to the user. Take, for example, a home network where a user uses a computer and browses the Internet, or in simpler terms connecting to servers that are external to its network. Outgoing NAT enabled on the IGD used for that home network would allow for the user to

Based on the literature review, design of the study and model, and descriptive observations, the following recommendations come forth. The following section will address recommendations such as adding authentication to certain forms of NAT, replacing some NAT traversal methods, reducing performance costs in new models such as those presented in this study, and the need for future research on the continuing advancement of networking technologies. Given that networking is an ever-evolving technology, there will be a need for future research. With the advent of IPv6, the need for NAT in many networks may dwindle. The discussion by Beeharry and Nowbutsing (2016) showed that the adoption of IPv6 was slow but growing (Beeharry & Nowbutsing, 2016). Further IPv6 adoption will surely change NAT's usage, and the researcher does not underestimate the elimination of NAT altogether in the future.

Using Authentication with Bi-Directional NAT

As mentioned in Chapters 1 and 2, bi-directional NAT has been around since NAT’s inception (Müller, Evans, Grothoff, & Kamkar, 2010; Srisuresh & Holdrege, 1999). Chapter 2 also introduced the need for security when working with any external entity to the IGD. There are malicious actors always attempting to cause harm on any device they can access. With any new NAT model that allows for traffic to dynamically make its way to the internal network, there must be some authentication form. As seen with the overhead produced with this new form of NAT, it introduced associated costs due to multiple aspects.

Additional Authorization with Bi-Directional NAT

Like adding authentication, authorization is another consideration with multiple aspects to provide security for the network at hand. First, a device authorized to make new connections to the internal network should not connect any device on that internal network. This new model of NAT implements this by matching unique names and keys. These names and keys should be dynamically generated and not guessable to move from a proof of concept to a production environment. In the new model of NAT, an external entity can only communicate with entities set up within the database on the IGD.

Second, a feature such as this new model of NAT should be disabled by default on any device as users of a network may want more control over their network and how traffic traverses it.

Reducing Performance Costs of the New Model

This new model of NAT is proof of the concept of how new NAT models could develop. During development, the performance was not a concern. There is likely room for performance improvement with more time and expertise with specific technologies. One such place that the researcher

Performance might also be reduced in individual sections outside of the developer's scope with modification made within pfSense, the IGD used for testing. If, for instance, pfSense modified its filtering to reload portions of the filter without reloading the entire filter where the NAT rules exist. This partial reloading could significantly shorten the time for each dynamically generated rule.

Using different hardware could also mitigate some of the performance costs of the new secure model of NAT. For example, the IGD in this study only used one processer core of a Xeon E5-2630 running at 2.30GHz. Adding more cores or processors to a production device would mitigate some of the processor costs. Additionally, using a processor with a higher base clock rate would also mitigate some of the CPU cost.

Taking more time to develop and focusing on performance could decrease the overall RAM utilization. The RAM's overall increase is not as concerning as the CPU usage. Although the new secure model uses roughly 105KB on average RAM, this is not significant to the system itself as it ran with 4GB of ram. The additional RAM used was insignificant to the amount that was free on the system.

Replacing Other NAT Traversal Methods

This study showed that this method could successfully bi-directionally traverse NAT with added authentication. This technology can replace certain other types of NAT traversing technologies. This model would not replace all other NAT forms as its primary enhancements do not lend themselves to NAT used on many networks. Many of the variations viewed in Chapter 2 used a third-party server to tell another client about the NAT they were behind. This model has similarities to that in the traditional client-server model, but a slight modification could add that feature.

Customer Edge Switching (CES), as discussed in Chapter 2, was a novel idea that allowed traffic to traverse from end to end with the possibility of both endpoints being behind NAT gateways. While the idea of this new model of NAT could add in features to allow this, it would take continued research. This instance is another where the new study does not replace a current solution.

IPv6 Adaption

IPv6 migration is slow but is increasing in speed each year (Beeharry & Nowbutsing, 2016). Therefore, while this new model primarily uses IPv4 addresses, consideration must be made for the future. This model could already substitute IPv6 addresses for some IPv4 addresses and carry on in most cases. However, if an entire network changes to IPv6 and every address is exposed publicly, this model would need refinement. There would no longer be a need for port forwarding as the new IPv6 address would be globally routable. However, the part of the model that creates an opening in the firewall could still be relevant even in a world full of IPv6 devices. Another possible scenario is that many external IP addresses switch to using an IPv6 address while still using NAT and an IPv4 private scheme on the inside of the network. That is another case where this new model of NAT would still find use.

Security Audits

Any form of NAT following the research should have source code analysis and dynamic security analysis as part of a development life cycle. These security tests hope to root out as many security vulnerabilities as possible before a new piece of software is released, especially one exposed to the Internet—chapter 2 related various forms of NAT and demonstrated that a virtual private network had similarities. A vulnerability was discovered in VPN software in a recent security case, causing users of the VPN software major issues ("Cisco AnyConnect Secure Mobility Client Arbitrary Code Execution Vulnerability," 2020). This vulnerability demonstrates the need for security tests during the development of any NAT model.

Recommendation for Future Research

Research should continue to develop systems for how an IGD will identify itself as a system supporting this type of NAT and other forms of NAT. This system development is challenging as the NAT traversal could become less transparent.

Further development should continue to better the security mechanisms found in this work. For example, further work to change the NAT traversal's nature to follow a more policy-based mechanism could improve the current state. This policy-based mechanism would allow an internal client to specify multiple connection conditions before an external host could connect through a firewall to the internal device. The first of these policy restrictions is where the connection originated. The client could require a connection from an external source from a specified IP address or a hostname with a valid certificate. Using a hostname and certificate method could be appropriately verified using the already implemented certificate authority (CA) system and DNS.

Although this study focuses on NAT traversal using IPv4, IPv6 adoption is growing (Beeharry & Nowbutsing, 2016). The mechanisms described here allow an external host to initiate connections to an internal host, and the proof of concept developed for this study could be improved to support IPv6 further. As the adoption of IPv6 grows on the Internet, internal networks may still implement IPv4 addresses and

Summary

Chapter 5 wrapped up this study by presenting the conclusions and the recommendations made by the researcher. This study's primary objective was to inquire about the change in overhead due to additional security mechanisms to NAT traversal with dynamic configuration and bi-directional traversal. This study completed the objective, and all of the sub-objectives were left answered. The most significant sub-objectives were the third and final sub-objectives measuring the network's change after the researcher's intervention. These measures saw significant increases in overhead in two of the three categories, making the new model computationally more expensive to run.

The newly implement NAT could provide additional features that introduced bi-directional traffic and security features. Overall, networking tools, protocols, and systems are needed to continue sustaining and securing new technologies that are being created and connected to networks every day.

REFERENCES

Amir, K. C., Goulart, A., & Kantola, R. (2016). Keyword-driven security test automation of Customer Edge Switching (CES) architecture. *Proceedings of 2016 8th International Workshop on Resilient Networks Design and Modeling, RNDM 2016*, 216–223. https://doi.org/10.1109/RNDM.2016.7608290

Anderson, D. (2020). How NAT traversal works. Retrieved November 2, 2021, from https://tailscale.com/blog/how-nat-traversal-works/

Beeharry, J., & Nowbutsing, B. (2016). Forecasting IPv4 exhaustion and IPv6 migration. *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies, EmergiTech 2016*, 336–340. https://doi.org/10.1109/EmergiTech.2016.7737362

Boslaugh, S. (2013). *Statistics in a Nutshell* (2nd ed.). O'Reilly.

Boucadair, M., Penno, R., & Wing, D. (2013). *Universal Plug and Play (UPnP) Internet Gateway Device - Port Control Protocol Interworking Function (IGD-PCP IWF)*. https://doi.org/10.17487/rfc6970

Braden, R. (1989). *Requirements for Internet Hosts - Communication Layers* (R. Braden, Ed.). https://doi.org/10.17487/rfc1122

Canonical Ltd. (2016). The leading operating system for PCs, tablets, phones, IoT devices, servers and the cloud | Ubuntu. Retrieved December 2, 2018, from Website website: https://www.ubuntu.com/

Cheshire, S., Boucadair, M., Penno, R., & Selkirk, P. (2013). *Port Control Protocol (PCP)* (D. Wing, Ed.). https://doi.org/10.17487/rfc6887

Cheshire, S., & Krochmal, M. (2013). *NAT Port Mapping Protocol (NAT-PMP)*.

https://doi.org/10.17487/rfc6886

Cisco AnyConnect Secure Mobility Client Arbitrary Code Execution Vulnerability. (2020). Retrieved from https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-anyconnect-ipc-KfQO9QhK

Creswell, J. W., & Creswell, J. D. (2018). *Research design: qualitative, quantitative, and mixed methods approaches* (Fifth). SAGE.

Cullen, M., Hartman, S., Zhang, D., & Reddy, T. (2015). *Port Control Protocol (PCP) Authentication Mechanism*. https://doi.org/10.17487/RFC7652

Deering, S., & Hinden, R. (2017). *Internet Protocol, Version 6 (IPv6) Specification (No. RFC 8200).*

Deshmukh, D., & Iyer, B. (2017). Design of IPSec virtual private network for remote access. *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, *2017-Janua*, 716–719. https://doi.org/10.1109/CCAA.2017.8229894

Etherington, D., & Conger, K. (2016). Large DDoS attacks cause outages at Twitter, Spotify, and other sites. Retrieved January 12, 2019, from TechCrunch website: https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/?guccounter=2%0Ahttps://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/

Flores, M. A., & Santisteban, J. (2017). Complete cone symmetric temporary NAT. *Proceedings of the 2016 IEEE 23rd International Congress on Electronics,* (Server X), 1–5. https://doi.org/10.1109/INTERCON.2016.7815574

Ford, B., Kegel, D., & Srisuresh, P. (2009). *Peer-to-Peer Communication Across Network Address Translators*. 1–18.

Frankel, S., Hoffmann, P., Orebaugh, A., & Park, R. (2008). Guide to SSL VPNs - Recommendations of the National Institute of Standards and Technology. *NIST Special Publication*. https://doi.org/150393

FreeBSD. (2021). Retrieved from https://www.freebsd.org/

Fuller, V., & Li, T. (2006). *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan (No. RFC 4632)*. 1–27. Retrieved from https://tools.ietf.org/html/rfc4632

Ham, M. (2017). *BGP Route Attestation: Design and Observation Using IPV6 Headers*.

Ilker, E., Musa, S. A., & Alkassim, R. S. (2016). Comparison of Convenience Sampling and Purposive Sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1. https://doi.org/10.11648/j.ajtas.20160501.11

Iqbal, A., Pattinson, C., & Kor, A. L. (2015). Performance monitoring of Virtual Machines (VMs) of type I and II hypervisors with SNMPv3. *2015 World Congress on Sustainable Technologies, WCST 2015*, 98–99. https://doi.org/10.1109/WCST.2015.7415127

Kamkar, S. (2020). NAT Slipstreaming v2.0. Retrieved from https://samy.pl/slipstream/

Kantola, R. A. (2010). Implementing trust-to-trust with customer edge switching. *24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010*, 1092–1099. https://doi.org/10.1109/WAINA.2010.27

Kaup, F., Michelinakis, F., Bui, N., Widmer, J., Wac, K., & Hausheer, D. (2015). Behind the NAT- A measurement based evaluation of cellular service quality. *2015 11th International Conference on Network and Service Management (CNSM)*, 228–236. https://doi.org/10.1109/CNSM.2015.7367363

Keranen, A., Holmberg, C., & Rosenberg, J. (2018). *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal*. https://doi.org/10.17487/RFC8445

Kumar, R. (2019). *Research methodology: A step-by-step guide for beginners (5th. ed.)* (5th ed.). SAGE.

Lazar, J., Feng, J., & Hochheiser, H. (2017). *Research Methods in Human-Computer Interaction* (2nd ed.). Morgan Kaufmann.

Leppaaho, P., Beijar, N., Kantola, R., & Santos, J. L. (2013). Traversal of the customer edge with NAT-unfriendly protocols. *IEEE International Conference on Communications*, (1), 2933–2938. https://doi.org/10.1109/ICC.2013.6654988

Mahy, R., Matthews, P., & Rosenberg, J. (2010). *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN) (No. RFC 5766)*. 1–67. Retrieved from https://tools.ietf.org/search/rfc5766

Müller, A., Evans, N., Grothoff, C., & Kamkar, S. (2010). Autonomous NAT traversal. *2010 IEEE 10th International Conference on Peer-to-Peer Computing, P2P 2010 - Proceedings*, (Figure 1), 0–3. https://doi.org/10.1109/P2P.2010.5569996

Netgate. (2018). pfSense® - World's Most Trusted Open Source Firewall. Retrieved December 2, 2018, from https://www.pfsense.org/

Netgate. (2019). References — Netgate Documentation.

Novo, O. (2018). Making Constrained Things Reachable. *ACM Transactions on Internet*

Technology, *19*(1), 1–21. https://doi.org/10.1145/3230640

Postel, J. (1981). *Internet Protocol*. https://doi.org/10.17487/rfc0791

Rahman, H., Wang, G., Chen, J., & Jiang, H. (2018). Performance Evaluation of Hypervisors and the Effect of Virtual CPU on Performance. *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 772–779. https://doi.org/10.1109/SmartWorld.2018.00146

Redzovic, H., Smiljanic, A., & Savic, B. (2017). Performance evaluation of software routers with VPN features. *Telfor Journal*, *9*(2), 74–79. https://doi.org/10.5937/telfor1702074R

Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., & Lear, E. (1996). *Address Allocation for Private Internets (No. RFC 1918)*. 1–9. Retrieved from https://tools.ietf.org/html/rfc1918

Rosenberg, J., Mahy, R., Matthews, P., & Wing, D. (2008). *Session Traversal Utilities for NAT (STUN)*. https://doi.org/10.17487/rfc5389

Rosenberg, J., Weinberger, J., Huitema, C., & Mahy, R. (2003). *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs) (No. RFC 3489)*. 1–47. Retrieved from https://tools.ietf.org/html/rfc3489

Santos, J. L., Kantola, R., Beijar, N., & Leppaaho, P. (2013). Implementing NAT traversal with Private Realm Gateway. *2013 IEEE International Conference on Communications (ICC)*, 3581–3586. https://doi.org/10.1109/ICC.2013.6655107

Srisuresh, P., & Egevang, K. (2001). *Traditional IP Network Address Translator (Traditional NAT)*. https://doi.org/10.17487/rfc3022

Srisuresh, P., & Holdrege, M. (1999). *IP Network Address Translator (NAT) Terminology and Considerations*. 1–30. https://doi.org/10.17487/rfc2663

Suzuki, H., Goto, Y., & Watanabe, A. (2007). External dynamic mapping method for NAT traversal. *ISCIT 2007 - 2007 International Symposium on Communications and Information Technologies Proceedings*, 723–728. https://doi.org/10.1109/ISCIT.2007.4392111

Tailscale. (2021). Retrieved from https://tailscale.com/

Timcenko, V., Djordjevic, B., Rakas, S. B., & Davidovic, N. (2014). Performance examination of type-2 hypervisors: case of particular database application in a virtual environment. *Proceedings of the International Conference on Information Systems and Design of Communication*, 122–126. https://doi.org/10.1145/2618168.2618187

Vojnak, D. T., Eordevic, B. S., Timcenko, V. V., & Strbac, S. M. (2019). Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation. *2019 27th Telecommunications Forum (TELFOR)*, 1–4. https://doi.org/10.1109/TELFOR48224.2019.8971213

Ware, H., & Frédérick, F. (n.d.). Ubuntu Manpage: vmstat - Report Virtual Memory Statistics. Retrieved from http://manpages.ubuntu.com/manpages/trusty/man8/vmstat.8.html

Yang, L., & Lei, K. (2016). *Combining ICE and SIP Protocol for NAT Traversal in New Classification Standard*. 576–580.

Zhang, R., Zhu, L., Han, Y., Zhang, L., & Feng, D. (2016). Design and implementation of double-NAT-traversal data transmission system. *Proceedings - 5th International Conference on Instrumentation and Measurement, Computer, Communication, and Control, IMCCC 2015*, (61370063), 1161–1165. https://doi.org/10.1109/IMCCC.2015.249

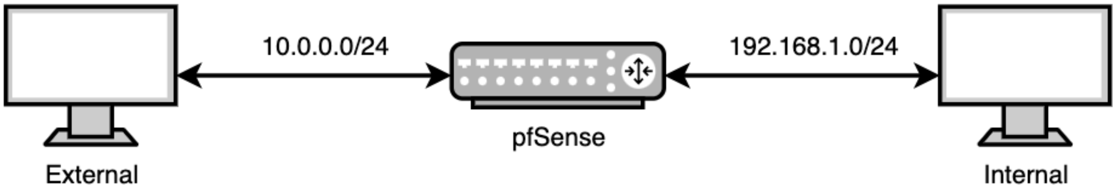
Zhao, G., & Gao, Y. (2015). Study on the Ultralin NAT-PT performance. *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 649–653. https://doi.org/10.1109/CYBER.2015.7288017

APPENDIXES

APPENDIX A: SYSTEM DESIGN AND NETWORK CONFIGURATION

The network used was created in an isolated, virtualized environment. There was an IGD or router, and in this case, pfSense was the IGD in use. pfSense connected to two separated networks and did NAT translations between them. The “External” network had a subnet of 10.0.0.0/24, and the “Internal” network used a subnet of 192.168.1.0/24. DHCP provided both networks with addresses, although both could use static addresses.

The operating systems in use were Ubuntu 20.04 for both the external and internal machines.



Included in future appendices is the code running on each of the devices for each type of test.

APPENDIX B: SUMMARIZED INSTRUMENT OUTPUT

The table below shows the averages of the data collected for the static port forwarding tests. The external device sends 1,000 requests through the IGD.

Trial	**CPU**	**RAM (KB)**	**RTT (Seconds)**
1	1.429%	734.84	0.267
2	1.482%	734.84	0.270
3	1.405%	734.84	0.267
All	1.44%	734.84	0.268

Trial	**CPU**	**RAM (KB)**	**RTT (Seconds)**
1	75.665%	831.42	1.397
2	75.713%	835.98	1.398
3	75.653%	850.42	1.399
All	75.68%	839.27	1.398

APPENDIX C: MODIFICATIONS MADE TO PFSENSE

Below is a listing of the modifications made to pfSense for ease of repeatability. Developing some changes on a FreeBSD system and then transferring it to pfSense may aid in development.

Adding pfSense_FauxAPI (The Github packages repository contains the latest package available): https://github.com/ndejong/pfsense_fauxapi_packages). The version used in this study was 1.4.1

1. Download the latest package
2. Upload package to pfSense
3. Static install via `pkg-static install pfSense-pkg-FauxAPI-1.4_1.txz`
4. Modify the credential file `/etc/fauxapi/credentials.ini`
    a. The Github Repository has information on how to correctly set up this file for securely using the package: https://github.com/ndejong/pfsense_fauxapi
5. The credential file used for testing

```
[PFFATesting011]
secret = Password11Password11Password11Password11
permit = *
comment = Testing
```

Install Python pip for installing other packages:

1. `curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py`
2. `python3.7 get-pip.py`

Installing pfSense-Fauxapi Python package

1. python3.7 -m pip install pfSense-fauxapi

Installing requirements for Python webserver

1. `python3.7 -m pip install flask`
2. `python3.7 -m pip install flask_sqlalchemy`
3. `python3.7 -m pip install gunicorn`
4. `pkg install py37-sqlite3-3.7.9_7`
5. `python3.7 -m pip install flask_migrate`

APPENDIX D: TEST/RETEST SCORES FOR MODIFIED TOOL

The test/retest method verified the instrument's reliability to measure the round-trip times. The test/retest method is a method to determine an instrument's reliability by comparing two testing rounds (Kumar, 2019). The number of tests chosen for each of the tests was 10,000. Once the testing concluded, the average time calculation led to the average round-trip time present in the table below. This number provides a significant enough sample size to compare times.

Trial	**Amount**	**Average Round-Trip Time**
1	10,000	1.8437 seconds
2	10,000	1.8504 seconds

APPENDIX E: CODE USED FOR PFSENSE

Flask web application run on pfSense:

```
     #!/usr/local/bin/python3.7
     # Middleware (pfSense)
     from flask import Flask, request
     from flask_sqlalchemy import SQLAlchemy
     from flask_migrate import Migrate
     from werkzeug.middleware.proxy_fix import ProxyFix
     import uuid, time
     import json
     import requests
     from PfsenseFauxapi.PfsenseFauxapi import
PfsenseFauxapi
     import random
     import sys
     import os

     app = Flask(__name__)
     app.config['SECRET_KEY']='Password1!'
     app.config['SQLALCHEMY_DATABASE_URI']='sqlite:////root
//test.db'
     app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
     db = SQLAlchemy(app)
     app.wsgi_app=ProxyFix(app.wsgi_app)
     migrate = Migrate(app, db)

     class Devices(db.Model):
         id = db.Column(db.Integer, primary_key=True)
         port = db.Column(db.Integer)
         internal_device = db.Column(db.String(20))
         key = db.Column(db.String(100))
         device = db.Column(db.String(100))

     class Translations(db.Model):
         id = db.Column(db.Integer, primary_key=True)
         pfsense_port = db.Column(db.Integer)
         external_ip = db.Column(db.String(20))
         internal_ip = db.Column(db.String(20))
         device = db.Column(db.String(20))
         device_port = db.Column(db.Integer)
         key = db.Column(db.String(100))
         time = db.Column(db.String(30))

     @app.before_request
     def before_request():
         T1 = Translations.query.all()
         for i in T1:
             if int(time.time()) > int(i.time)+150:
                 try:
                     apiobj = PfsenseFauxapi('127.0.0.1', 
'PFFATesting011', 
'Password11Password11Password11Password11')
                     config = apiobj.config_get()
                     for j in config['filter']['rule']:
                         if j['descr'] == i.device:

config['filter']['rule'].remove(j)
                     for j in config['nat']['rule']:
                         if j['descr'] == i.device:

config['nat']['rule'].remove(j)
                     apiobj.config_set(config)


Translations.query.filter_by(device=i.device).delete()
                     db.session.commit()
                 except Exception as e:
                     print(e)

     @app.route("/")
     def index():
         return {"Test":"Data"}

     @app.route('/create/<device>', methods=['GET','POST'])
     def createDevice(device):
         data = request.get_json(force=True)
         ip = request.remote_addr
         d1 = Devices(port=data['port'], 
internal_device=ip, key=data['key'], device=device)
         db.session.add(d1)
         db.session.commit()
         return {"Message":"Success"}

     @app.route('/open/<device>', methods=['POST'])
     def openPort(device):
         try:
             data = request.get_json(force=True)
             t1 = int(time.time())
             src_ip = request.remote_addr
             pfsense_port = 
str(random.randint(40000,50000))
```

```
            ports = [t.pfsense_port for t in
Translations.query.all()]
            while (pfsense_port in ports):
                pfsense_port =
str(random.randint(40000,50000))

            d1 =
Devices.query.filter_by(device=device).first_or_404()
            internal_port = d1.port
            src_interface = 'opt1'
            uid = uuid.uuid1()
            apiobj = PfsenseFauxapi('127.0.0.1',
'PFFATesting011',
'Password11Password11Password11Password11')
            config = apiobj.config_get()
            firewall_rule = {'associated-rule-id': '',
                               'created': {'time': str(t1),
                                           'username': 'NAT
Port Forward'},
                               'descr': str(device),
                               'destination': {'address':
str(d1.internal_device), 'port': str(internal_port)},
                               'interface': src_interface,
                               'ipprotocol': 'inet',
                               'protocol': 'tcp',
                               'source': {'address':
src_ip},
                               'tracker': str(t1)}

            #nat_rule = {'associated-rule-id': '',
            nat_rule = {
                                'created': {'time':
str(t1),
                                            'username':
'admin@192.168.1.10 (Local Database)'},
                                'descr': str(device),
                                'destination': {'network':
'opt1ip', 'port': pfsense_port},
                                'interface': src_interface,
                                'ipprotocol': 'inet',
                                'local-port':
str(internal_port),
                                'protocol': 'tcp',
                                'source': {'any': ''},
                                'target':
str(d1.internal_device),
                                'updated': {'time':
str(t1),
                                            'username':
'admin@192.168.1.10 (Local Database)'}}

             config['filter']['rule'].append(firewall_rule)
             if 'rule' not in config['nat']:
                 config['nat']['rule'] = []
             config['nat']['rule'].append(nat_rule)
             apiobj.config_set(config)
             ret1 =
os.system('/etc/rc.filter_configure_sync')
             dport =
Devices.query.filter_by(device=device).first_or_404().port
             t2 =
Translations(pfsense_port=int(pfsense_port),
external_ip=src_ip,
internal_ip=d1.internal_device,device=device,
device_port=dport, key=data['key'], time=t1)
             db.session.add(t2)
             db.session.commit()
         except Exception as e:
             return json.dumps({"Exit":str(e)})
         return json.dumps({"Port":int(pfsense_port),
"Time":str(t1)})

     @app.route('/delete/<device>/<time>', methods=['GET'])
     def deleteTranslation(device, time):
         Translations.query.filter_by(device=device,
time=time).delete()
         db.session.commit()
         apiobj = PfsenseFauxapi('127.0.0.1',
'PFFATesting011',
'Password11Password11Password11Password11', debug=True)
         config = apiobj.config_get()
         frules = config['filter']['rule']
         for j in frules:
             if j['descr'] == device and j['tracker'] ==
time:
                 frules.remove(j)
         nrules = config['nat']['rule']
         for j in nrules:
             if j['descr'] == device and
j['created']['time'] == time:
                 nrules.remove(j)
         config['nat']['rule'] = nrules
         config['filter']['rule'] = frules
```

```
    apiobj.config_set(config)
    os.system('/etc/rc.filter_configure_sync')

    return {"Delete":"Success"}

if __name__ == '__main__':
    app.run()
```

APPENDIX F: CODE USED FOR INTERNAL HOST

The code running on the internal host is a web server designed to act as a fake internal device. This device could simulate an Internet of Things (IoT) device living inside a home network.

```
     #!/usr/bin/python3
     # Internal Server
     from flask import Flask, request
     import requests
     import netifaces
     import json
     requests.packages.urllib3.disable_warnings()
     app = Flask(__name__)

     @app.before_first_request
     def initiate():
         # Get key from server (this would be a known
domain name)
         homing_url = "https://10.0.0.10/generate"
         r1 = requests.get(homing_url, verify=False)
         # Sends name to middleware to keep in the table
         key = json.loads(r1.json())['key']
         print(key)
         device = 'UbuntuIoT'
         port = 443
         gateway = "192.168.1.1"
         url = "https://" + str(gateway) + ":8080/create/"
+ device
         print(url)
         r = requests.post(url,
data=json.dumps({"port":port, "key":key}), verify=False)
         print(r.text)


     @app.route('/', methods=['GET', 'POST'])
     def index():
         return {"message":"Success Index"}

     @app.route('/receive', methods=['GET', 'POST'])
     def receiveTraffic():
         print(request.get_json(force=True))
         return {"message":"Success Recv"}

     if __name__ == '__main__':
```

```
app.run()
```

APPENDIX G: CODE USED FOR EXTERNAL HOST

The external host runs both a web server and a simple python script. The web server is for the initial client to reach out and generate a key.

```
     #!/bin/python
     # External key generation
     from flask import Flask, request
     import requests
     import json, os, time, uuid, sys

     app = Flask(__name__)


     @app.route('/generate', methods=['GET'])
     def generateKey():
         print("Generate")
         # Hardcoding key for now, generate this
dynamically for more security
         return json.dumps('{"key":"Password1!"}')


     if __name__ == '__main__':
         app.run()
```

APPENDIX H: CODE USED TO TEST UNMODIFIED PORT FORWARDING TIMES

Internal:

```
#!/usr/bin/python3
# Internal Server (Port Forward)
from flask import Flask, request
import requests
requests.packages.urllib3.disable_warnings()
app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    return {"message":"Success Index"}

@app.route('/receive', methods=['GET', 'POST'])
def receiveTraffic():
    print(request.get_json(force=True))
    return {"message":"Success Recv"}

if __name__ == '__main__':
    app.run()
```

External:

```
#!/usr/bin/python3
# External Server
import requests
import json
requests.packages.urllib3.disable_warnings()

def sendInformation(device, m, port):
    r =
requests.post('https://10.0.0.1:'+str(port)+'/receive',
data=json.dumps({'message':m}), verify=False)
    #print(r.text)

def main():
    d1 = "UbuntuIoT"
    port = "51000"
    sendInformation(d1, "UpdateInfo", port)
```

```
if __name__ == '__main__':
    main()
```

APPENDIX I: RUN SCRIPT

Script used to run tests for the secure model of NAT (`run1.sh`)

```
num=1
ssh admin@10.0.0.1 -f -x "vmstat -c 100000 -w .5 -H >>
ftime$num.txt"
for i in `seq 0 1000`; do (time python3 external_runner.py)
&>> ftime$num.txt; done
pid=`ssh admin@10.0.0.1 -x "pgrep vmstat"`
ssh admin@10.0.0.1 -x "kill $pid"
```

Script used to run tests for port forwarding version of code (`run.sh`):

```
num=1
ssh admin@10.0.0.1 -f -x "vmstat -c 100000 -w .5 -H >>
ptime$num.txt"
for i in `seq 0 1000`; do (time python3
external_portfwd.py) &>> ptime$num.txt; done
pid=`ssh admin@10.0.0.1 -x "pgrep vmstat"`
ssh admin@10.0.0.1 -x "kill $pid"
```

APPENDIX J: GITHUB LINK FOR CODE

`https://github.com/tjflaagan/Dissertation`