Spring 3-2021

Block the Root Takeover: Validating Devices Using Blockchain Protocol

Sharmila Paul
Dakota State University

Block the Root Takeover: Validating Devices using Blockchain Protocol

A Dissertation Defense Presented to Dakota State University in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in

Cyber Operations

March, 2021

By
Sharmila Paul

Dissertation Committee:

Dr. Michael Ham
Dr. Yong Wang
Dr. Katie Anderson
Dr. Dallas Wright
Dr. Sunny Wear

DISSERTATION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Science in Cyber Operations 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: Sharmila Paul

Dissertation Title: Block the Root Takeover: Validating Devices using Blockchain Protocol

Dissertation Chair:
Dr. Michael Ham _____ Date: _____

Committee member:
Dr. Yong Wang _____ Date: _____

Committee member:
Dr. Katie Anderson _____ Date: _____

Committee member:
Dr. Dallas Wright _____ Date: _____

Committee member:
Dr. Sunny Wear _____ Date: _____

Abstract

This study addresses a vulnerability in the trust-based STP protocol that allows malicious users to target an Ethernet LAN with an STP Root-Takeover Attack. This subject is relevant because an STP Root-Takeover attack is a gateway to unauthorized control over the entire network stack of a personal or enterprise network. This study aims to address this problem with a potentially trustless research solution called the STP DApp. The STP DApp is the combination of a kernel */net* modification called *stpverify* and a Hyperledger Fabric blockchain framework in a NodeJS runtime environment in userland. The STP DApp works as an Intrusion Detection System (IPS) by intercepting Ethernet traffic and blocking forged Ethernet frames sent by STP Root-Takeover attackers. This study's research methodology is a quantitative pre-experimental design that provides conclusive results through empirical data and analysis using experimental control groups. In this study, data collection was based on active RAM utilization and CPU Usage during a performance evaluation of the STP DApp. It blocks an STP Root-Takeover Attack launched by the Yersinia attack tool installed on a virtual machine with the Kali operating system. The research solution is a test blockchain framework using Hyperledger Fabric. It is made up of an experimental test network made up of nodes on a host virtual machine and is used to validate Ethernet frames extracted from *stpverify*.

Declaration

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

Signed,

_____

Sharmila Paul

This paper is written in loving memory of my father who imparted in me a thirst for knowledge and a curiosity to learn how things work and my grandmother who taught me strength and bravery.

"Why, sometimes I've believed as many as six impossible things before breakfast."

— Lewis Carroll, *Alice in Wonderland*

Acknowledgment

Es ist wahr: Wir lieben das Leben, nicht, weil wir ans Leben, sondern ans Lieben gewöhnt sind
We love life, not because we are used to living but because we are used to loving
— Friedrich Nietzsche, *Also sprach Zarathustra*

I used to think of achievement as individualistic and that only we are the sole sculptors of our successes. However, in my case, such paths always came to a roadblock. In this journey, my biggest lesson was to learn that the word achievement not only has an 'i', but many other letters to support it. I thought academic success was a personal goal, but it was more than that. I realized that without the support and encouragement of my family, friends, and colleagues to help me find those stepping stones in the rocky paths, no such goal could manifest.

My mother has guided me through this journey along the path of life as if it were her own. Her unwavering loyalty to help me through life's changing seasons has forever cemented her place in my heart. My brother managed a lot of things on my behalf when I struggled to focus. I can't thank him enough for his grounding advice in the moments I needed him most. My lifelong friends from my work overseas, Dana and Trisha, took a real conscientious effort in helping me repatriate back home to the US while on this journey. Although we live at a physical distance from each other, we still kick each other's rears continuously like we are next to each other. These are my friends that helped me realize that family is not always blood. Thanks for being patient with me when I could not be there for you when I wanted to and just dealing with me in general. Our friends are families WE choose....so I guess this one's our fault, and we are stuck with each other.

In an unexpected and very last-minute turn of events, I was required to use the university's network on location to complete my research. Words cannot describe the crucial role of Dr. Wayne Pauli, Dr. Tom Halverson, and Dr. Michael Ham to ease my transition to Dakota State University's campus and its family. Because of their great attention to detail, what would have been a difficult move turned into home very quickly.

It is never difficult to find research topics but manifesting the research was a feat I had never imagined. I was lucky to have the undivided guidance of the chair of my committee,

Dr. Michael Ham, and committee experts Dr. Yong Wang, Dr. Katie Anderson, Dr. Dallas Wright, and Dr. Sunny Wear. My committee inspired me and took so much time out to help me understand what this journey is all. Dr. Michael Ham never faltered in genuine patience and generosity in sharing a high bandwidth of knowledge and experience to ensure strong research content and make this the best possible work I could do. Dr. Yong Wang provided authentic advice and played the *advocatus diaboli* to help strengthen the research. Dr. Katie Anderson took so much time to scour my writing for technical articulation, readability, and grammar based on her invaluable expertise from her field. She allowed me a practice run on the intimidating presentation part of the degree requirement. Dr. Dallas Wright is a former classmate and super-cheerleader with a high dependability factor. I would call him when I was panicking and he usually had an answer Finally, and not least, I am so blessed to have one of my colleagues, Dr. Sunny Wear from Tampa's Cybersecurity community, take an interest in my work. Her expertise, friendship, and enthusiasm proved so valuable in getting through this. I was very lucky to have a strong team that interested in learning about networks, the blockchain, and its ideology. It was the true collaboration among all committee members that helped me shape a strange idea into something tangible.

It is difficult not to expound extensively on what everyone has done for me through this journey. This year has forced me to manage and appreciate both the good and the bad and I've learned equally from both. I look forward to what lies ahead and hope that this research inspires security awareness and a better network infrastructure.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

Cyber Operations is an academic field of study designated by the National Security Agency (NSA) that utilizes computer science and engineering to research security problems and enhance national security (NSA, 2021). One way to research a security problem would be to analyze the root cause or its point of origin. The identification of the root cause can initiate the search for a solution. The root cause of a security problem is typically in an area where a weakness begins. In the idiom, ”a chain is only as strong as its weakest link,” a ”weak link” is the root cause that can make an otherwise foolhardy chain useless for its purpose. The first question for a security researcher will be to ask if the security problem exists. Upon validating the security problem, the second question would be to find the underlying root cause of the security research problem. What can designate one link weaker than another in a security research problem? Rahalkar (2018), in his book, *Network Vulnerability Assessment*, describes a vulnerability as a weakness in a digital information system (Rahalkar, 2018). More specifically, the vulnerabilities that he referenced in his book are in handling data in transport. A data transport system provides the path taken by data from its origin to its destination. A vulnerability in this path would be a weakness that could affect the Confidentiality, Integrity, and Availability (CIA) of the data. This triad is a common benchmark in measuring the breadth of digital security (Fenrich, 2008). To find the origin of these defining characteristics in is a subject of debate. However, it provides a high-level understanding of secure information transport. If Alice were to send Bob an email, the message would leave an email client on her computing device and reach Bob’s email client in another location. In this path, a security professional would analyze whether there was any point where this email could have been accessed by an unauthorized party, thus, weakening the Confidentiality of the message transport. Integrity is equally as crucial as it defines the trustworthiness of the data. If an unauthorized user accessed the email and they changed the contents, a weak link exists in the path. Lastly, Availability ensures that Alice can trust that the email she sent will reach Bob due to the information systems’ stability used to send the email.

A weakness in one of these primary security tenets can create a significant security

problem. A solution to such a situation would be to find a stable reinforcement that would prevent further vulnerability in the information system. When the email leaves Alice's computer to its destination, it follows a network path. A network path is like a road that data takes to reach its destination. Metaphorically speaking, congestion or construction can disrupt an otherwise uneventful drive. Equally, a vulnerability in a network path can also disrupt data transport.

Securing a vulnerability to strengthen the network path adds security measures so the unauthorized intruder cannot access, rewrite or block Alice's email from reaching Bob. Securing information is not a new concept. Secure communication started with military communications in warfare (Dooley, 2013). It was the messenger's ultimate job to ensure the security of information from one location to another. Julius Caesar (1919) would describe one example in his book, *The Gallic Wars* (Caesar, 1919). He describes protecting his messages by using a set pattern to manipulate the message to make it unrecognizable to an unauthorized reader (Caesar, 1919). This pattern, otherwise known as a cipher, is one of the earliest known forms of cryptology to secure the transport of information from source to destination (Cohen & Kahn, 1997; Dooley, 2013).

Open Systems Intercommunication (OSI) Reference Model

The Open Systems Intercommunication (OSI) can break down the path that Alice's email takes from her computer to Bob's computer into seven abstract network infrastructure today. These layers are called abstract because there is no defining physical feature to each layer. The OSI model defines each layer by its duty in handling the transport of the data on the network path. The network path works like an assembly line where each layer has a specific task to complete a working product. Hubert Zimmerman (1983) and the International Standards Organization (ISO) formalized the OSI reference model in 1977. The OSI model's creation was to standardize network communication (Day & Zimmermann, 1983). The OSI network architecture includes universal standards and protocols organized in abstract layers. Unfortunately, the current network infrastructure under the OSI model has significant limitations and design issues that date back to its inception, including vulnerabilities that provide malicious actors an opportunity to gain unauthorized access to network systems (Huang, 1995; Rahalkar, 2018).

The OSI reference model's seven abstract layers are the Physical, Data Link, Network, Transport, Session, Presentation, and Application layers. (Howser, 2020). Each layer in the OSI reference model provides a service by implementing layer-specific protocols. All the layers are interconnected and use the protocols to provide services to the layer above it, beginning at the Physical Layer and ending with the Application Layer (Day & Zimmermann, 1983). Originally

defined as abstract layers by John Day and Hubert Zimmerman (1983), the Physical Layer consists of standards to maintain the network systems' devices. For example, a security protocol stating that the data center must have a human guard would prevent an unauthorized intrusion to the physical devices at the Physical Layer. The Physical Layer, or Layer 1, sends data to Layer 2, the Data Link Layer, as a bit. A bit, or a binary digit, is the atomic form of digital data. Bits are the lowest form of data broken down into the smallest unit. To connect Layer 1 to Layer 2, a form of media is required to connect one device to another for data transport. Media can be physical or wireless. For example, physical media would be a copper wire cable connecting two computers (Howser, 2020). Media connects a device to the network using a Network Interface Card (NIC). Devices connect and communicate with media connections in between to form Local Area Networks (LANs). Wide Area Networks (WANs) connect separate LANs to each other. A unique identifier is assigned to each NIC to control access to the data transmitted and received to specific devices. The unique identifier is like an address for each network device and it uses a format defined by the Media Access Control (MAC) protocol (IEEE, 2018).

The Data Link Layer

The MAC address, also known as a physical or hardware address, allows a device to send and receive data using source and destination addresses. The data can be sent from a device in three different ways based on its intended destination. If a device sends data meant for one recipient in the LAN, it is called *unicast* messaging (Howser, 2020). A *broadcast* message is destined for receipt by all LAN devices, and a *multicast* message is addressed to specific devices with unique MAC addresses in the LAN (Howser, 2020). When a message is broadcast to all devices in the LAN, the destination address is FF:FF:FF:FF:FF:FF (Cisco Networking Academy, 2020). The Data Link Layer transports data in different ways based on the LANs topology. Two examples of Data Link Layer topologies are Token Ring and Ethernet LAN (Thompson, 2019).

A legacy topology using a Token Ring prevents collisions of transmitted and received data by passing a token to each device in the LAN, to each device one at at time (Cisco Networking Academy, 2020). Collisions can cause data loss because when two devices transmit data simultaneously, it must be re-transmitted by both devices causing network lag (Lammle, 2011). Like the game of passing the baton, when a computer gets its turn with the token, it transmits and receives data. Once the computer with the token has completed data transmission, it passes the token to the ring's next computer. The Institute of Electrical Electronics Engineers (IEEE), an organization of professionals for technological advancement, defined a set of stan-

dards for Token Ring LANs called IEEE 802.4. Support for IEEE 802.4 dwindled after 2008 when the Ethernet LAN topology took de-facto prevalence in the market (Thompson, 2019).

In standard Ethernet-based LANs, the Data Link Layer provides three services (IEEE, 2016). The first service consists of layer-specific standardized protocols to prevent errors in the Physical Layer data. The second service is with a protocol that manages connections among Ethernet devices. These Data Link Layer devices are called bridges or switches, and they move data across a LAN using MAC addresses as their source and destination points (Mehra & Krishnan, 2018). Two or more connected switches form a LAN, and a WAN connects two or more adjacent LANs (Sharma et al., 2018). The third service is ensures error free data transported to and from the Network Layer (Howser, 2020).

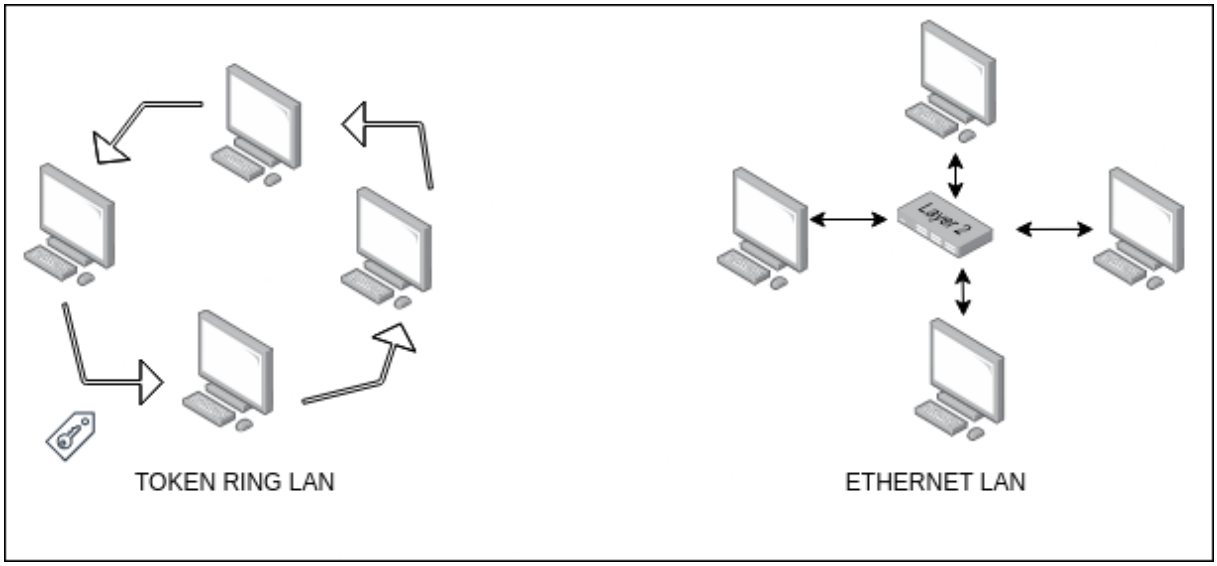


Figure 1. Token Ring vs. Ethernet LAN

The IEEE has defined a set of standards for network services using Ethernet. Wired Ethernet is a technology used to connect devices in a LAN and WAN using physical media based on IEEE's Project 802.3 (IEEE, 2018). Wireless Ethernet technology is defined under IEEE 802.11x and works with a different protocol set. For this study's purpose, the focus will be on the protocols used for wired Ethernet technology in the IEEE 802.3 standard (IEEE, 2016). IEEE 802.3 defines the low-level services in the Physical and Data Link Layer for wired Ethernet. The protocols provide these wired Ethernet services and conform to IEEE 802.3 (IEEE, 2018). Together, the Data Link Layer protocols deliver data among physical devices in a LAN, beginning with protocols that provide services like error checking the OSI layers below and above.

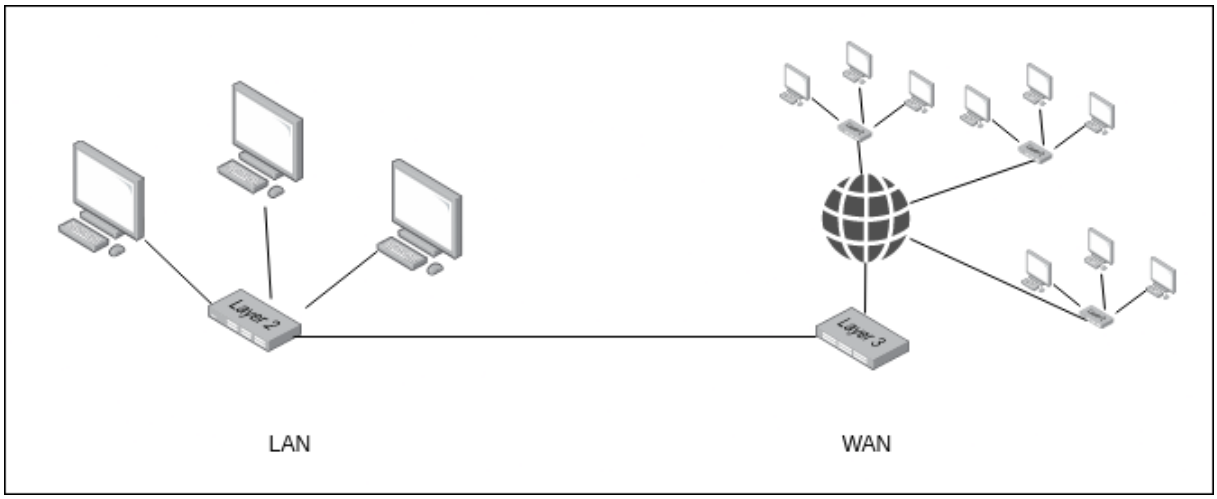


Figure 2. Visual Representation of a LAN vs. WAN

The Data Link Layer has two abstract subsections called the MAC sub-layer and the Local Link Control (LLC) sub-layer (Cisco Networking Academy, 2020). The MAC sub-layer provides protocol services, including Data Encapsulation and the Media Access Control (MAC) protocol (Cisco Networking Academy, 2020). At a high level, Data Encapsulation is a service that has three functions in the MAC sub-layer. It puts the Physical Layer binary data into a format called an Ethernet frame that allows its transport within the Data Link Layer (Cisco Networking Academy, 2020). It also uses Ethernet addressing to add a source and destination MAC address to the frame to be sent to the correct MAC address in the LAN. Lastly, it adds a Cyclic Redundancy Check (CRC) code at the end of the Ethernet frame to allow a hash check for errors in transmission (Chu, 2019). The MAC protocol manages Flow Control which defines and controls how frames are transmitted and received (Zimmerman & Spurgeon, 2014; Cisco Networking Academy, 2020).Further, the MAC protocol also handles the NIC hardware-based unique identifier used as addressing for data transport in the Data Link Layer (Howser, 2020). The LLC sub-layer provides a service to the Network Layer by identifying the Layer 3 protocol used for the Ethernet frame (Cisco Networking Academy, 2020). The LLC and MAC sub-layer services play a key role in ensuring stable and efficient data transport from the OSI network stack's lower and upper layers.

The Data Link protocols have enabled LAN networks to connect to other LAN networks worldwide but attacks on the Data Link Layer remain prevalent (Mahmood et al., 2020). This section provided a high-level overview of how the abstract Data Link Layer works within the OSI network infrastructure. The next section will provide insight into the security vulnerabilities that leave the Data Link Layer open to common attacks.

Common Attacks on the Data Link Layer

The last section described the Data Link Layer and its role in the network infrastructure. This section will introduce its vulnerabilities and current attacks targeting the Data Link Layer. A common gateway attack that begins at the Data Link Layer is the Man in the Middle (MITM) attack (Rahalkar & Jaswal, 2019). Publicly available security tools make common attacks like MITM attacks accessible to the average internet user (Chantzis et al., 2021). The MITM allows a malicious user to cause a Denial of Service (DoS) attack or stay dormant in the Ethernet LAN for information gathering (Singh, 2019). Layer 2 is arguably the most vulnerable to MITM attacks (Sak& Ram, 2016). Security precautions and software are more prevalent for Layers 3 and up (Mahmood et al., 2020). A breach in Layer 2 will inevitably affect the data's security in the layers above it (Mahmood et al., 2020). In a MITM attack, an unauthorized user gains access to a LAN network.

LAN networks can be internal networks connected together in an office or span several buildings. Typically, the Data Link Layer is most available to insider attacks and social engineering (UcedaVelez & Morana, 2015). A MITM attack works by inserting a rogue device into a LAN and routing traffic through it (Pingle et al., 2018). MITM attacks are initiated in the Data Link Layer because it is the weakest entry point (Gregg & Watkins, 2006). Some attackers that are information gathering with sniffers collect data coming from the upper layers, such as the Network and Application Layers (Rietz et al., 2018). Even if the rest of the layers have set up standard security settings, the unauthorized surveillance at the Data Link Layer compromises the Confidentiality of the data being passed through the other layers in the network stack (Lai et al., 2014; Rai et al., 2011).

Once a rogue device has a network connection to the LAN, the malicious user can use a sniffer to look at data going through the network. Sniffers access emails, email attachments, chat messages, network device configurations, web browser data and traffic, HTTP requests containing transaction information, and login information (Anu & Vimala, 2018). Although a MITM initiated the sniffing attack at Layer 2 from Ethernet II frames, the sniffed frames are the Network and Application Layers' data. The Data Link Layer protocol encapsulated the data from those layers in that frame. Using the information from a sniffer, an attacker can also forward traffic using various publicly available software tools for these attacks. They can also perform a DoS attack to disrupt the LAN system resources' Availability by forwarding enough traffic to overload a resource. Regardless of the attacks' methods, protocols such as STP that govern the wired 802.3 Ethernet LANs are trust-based. The network does not validate new devices

and payloads sent by a MITM attacker for authenticity. The STP-enabled network devices in an Ethernet LAN trust each other; therefore, it is trust-based (Mahmood et al., 2020; Perlman, 1985; Yeung, Yan & Leung, 2006). Several trust-based protocols have been a part of the network infrastructure since the 1980s and persist today. For example, the default server setup in a network allows for similar accessibility requirements; therefore, server systems' typical internal LAN environment will have open communications and channels to perform efficiently (Keeriyattil, 2018). An intruder that uses an internal server to perform malicious acts using these open channels takes advantage of the internal LAN's trust-based infrastructure (Keeriyattil, 2018).

Attacks that are initiated and executed in the Data Link Layer include passive and active attacks. Passive attacks do not involve any direct participation in the network, and active attacks involve direct manipulation of the attacker's network. Common attacks that have persisted over time have been passive sniffing attacks and active attacks, including MITM attacks, MAC attacks, ARP attacks, STP attacks, and dynamic host configuration protocol (DHCP) attacks. Noted but not in this study are two attacks on proprietary network switches called Cisco Discovery Protocol (CDP attacks) and Virtual LAN (VLAN) attacks (Mahmood et al., 2020; Yeung et al., 2006). They all take advantage of the trust-based nature of the protocols utilized to compromise the LAN systems. The initial step for the attacker to perform these attacks is to gain access to the Ethernet LAN of interest with a network packet analyzer device. A network sniffer or analyzer is software that captures traffic in the connected LAN network for monitoring and analysis (Gregg & Watkins, 2006). A few examples of commonly used network sniffers are *Wireshark*, *dsniff*, and *Cain and Abel*. This software performs a passive attack when used for malicious reasons because network traffic monitoring can allow the attacker to gather unauthorized information for malicious purposes. It is a passive attack because the attacker is only receiving information and not sending any traffic with a network analyzer (Siswanto et al., 2019). The attacker can choose to use a laptop, tablet, single-board computer, mobile phone, or networked device to host software and has a NIC. On Ethernet-based devices, a NIC has a MAC address and is used to send and receive unicast, multicast, and broadcast data (Cisco Networking Academy, 2020). As was mentioned earlier, unicast and broadcast messages are sent to specific MAC addresses, so a NIC under an Ethernet protocol will only receive messages addressed to their unique MAC address (Howser, 2020). However, when using a network analyzer, to pick up all the traffic passing through the network, even a unicast message meant for another device, the NIC set to monitor mode intercepts all traffic (Sak & Ram, 2016).

The network sniffer has a capture filter that will allow an attacker to choose the type of traffic for protocol analysis (Siswanto et al., 2019). For example, since the network sniffer works

in the Data Link Layer of an Ethernet LAN, the filter can be configured to save all Ethernet II frames. More protocol analysis filters can be used to choose a specific protocol in the frame's Ethertype like ARP requests or HTTP requests (Gregg & Watkins, 2006).

Through information gathered with a sniffer, an attacker may pick up data to launch an active attack. Data like IP addresses and MAC addresses are used to perform a spoofing attack. An attacker will take on another device's identity and perform malicious tasks like that in a spoofing attack. Identity theft is like spoofing, where a thief can gather personally identifiable information about a victim and start opening credit cards with this new identity, unbeknownst to the victim. MITM attacks are considered a significant threat to network security. They can be performed in three successive or mutually exclusive steps: entry and attack limited to just the LAN, access from the LAN to remote attack, and exclusively remote entry and attack (Ornaghi A. & Valleri, 2003). Although this concept is dated, the attacks that happen in this manner are relevant today. This study will focus on vulnerabilities and attacks within the LAN; however, it is crucial to note that pertinent attacks that compromise the Network Layer are provided a gateway from the initial LAN attack (Marconi Foundation, 2018; Rietz et al., 2018).

An attacker targeted by the LAN may compromise a device connected to the router or act as the router itself. The gateway router is the device used to enforce Network Layer protocols, including the IP protocol (Cisco Networking Academy, 2020). The infiltration into the Ethernet LAN can provide an attacker insider knowledge using the ARP protocol. Although it is a protocol associated with the Network Layer, the attacks use the MAC sub-layer protocol's vulnerability. The ARP protocol matches IP addresses with MAC addresses of a device and continually updates lists with no validation, making it trust-based. With the knowledge of IP and MAC addresses from devices in the network, an attacker can use MAC spoofing to enter the network. With MAC spoofing, the attacker passes the rogue device as a legitimate device by assigning it a MAC address and IP address already verified in the network. The attacker can then intercept communication between two users in the LAN by changing ARP cache entries. An ARP cache is a list of stored IP address / MAC address combinations of devices within the LAN. If a LAN device does not know the MAC address associated with an IP address, it will send out an Ethernet II frame with an ARP request, and if a device in the LAN has the pairing, it will send back an ARP reply with the MAC address. An ARP cache entry is updated when two devices in a LAN communicate, and an Ethernet II frame for an ARP reply updates the ARP cache on a device (Younes, 2017). An attacker already in the network can broadcast an ARP reply without an associated ARP request to each LAN device using the attacker's spoofed MAC address. The attacker will update the ARP cache for those two computers with their own MAC.

Now all packets meant for the other computer will go directly to the attacker's machine. Since the attacker changed the ARP cache, this type of attack is called a poisoning attack. Hence, it is called an ARP cache poisoning attack (Conti et al., 2016).

A Content Addressable Memory (CAM) table poisoning attack is similar, but it is directed at a switch instead of a computer. The switch has a table of MAC addresses associated with port numbers on the switch. Each port number and MAC address combination will connect directly to a networked device. The attacker has already associated themselves with a port in the switch and can generate Ethernet II frames replacing the source MAC address in the MAC header with all the other MAC addresses on the Ethernet LAN's switch. Now, the attacker will direct all the traffic directed to any device connected to the switch to the attacker's device. Since the other devices are no longer receiving traffic, compromising the network resources' Availability, this doubles as a DoS attack (Trabelsi, 2012). An intruder can perform a DoS attack called MAC address flooding by generating many Ethernet frames with spoofed MAC addresses to the switch to overload the CAM table with more entries than it can hold due to its limited buffer size. The switch is designed to work as a hub due to a fail-safe that allows it to continue communicating with the LAN devices. However, a hub only broadcasts traffic, so all computers, including the rogue machine, will receive all traffic (Anu & Vimala, 2018). The CSMA/CD protocol is used in half-duplex communications on the Data Link Layer. Half-duplex communications work by sending and receiving on separate media or control sending and receiving at separate times when working on the same media (Cisco Networking Academy, 2020). Measures taken by the CSMA/CD protocol to avoid collisions can result in other DoS attacks within a wired Ethernet LAN. In a topology like the Token ring, one device is transmitting data, and other devices wait for random times to send data (Cisco Networking Academy, 2020). An attacker automates a continuous stream of Ethernet frames to hinder other device transmissions in a Single Adversary Attack (SAA). Two rogue devices send data to block other devices from communicating in a Colluding Adversary Attack (CAA), there will be (Dasari, 2017).

An Application Layer protocol compromised in the Data Link Layer using a standard MITM attack is the Dynamic Host Configuration Protocol (DHCP). DHCP is a trust-based network server that assigns IP addresses to new hosts, and it takes advantage of a ticket-based solution that requires a key to secure the MAC / IP address combinations. It is called Ticket-based Address Resolution Protocol (TARP). MAC spoofing or other MITM mechanisms placing a rogue DHCP server in a LAN can allow an attacker to assign a new host on the network an IP address, a subnet address, and a gateway address. It can also be housed anywhere in the LAN

and can be a part of router software (Younes, 2017). The IP address, like a MAC address, is a unique address for a networked device. The subnet is an IP address grouping system based on the number of IP addresses available to the LAN. The gateway address is the IP address associated with the router that sends traffic into and out of the LAN. This IP assignment is temporary so that the attacker can send a DHCP server a request message for an IP lease renewal using the spoofed MAC and IP address. A Local Ticket Agent (LTA) authenticates the MAC and IP address combination sent by the attacker and sends a ticket back to the rogue DHCP server (Younes, 2017). This ticket acts as validation for the DHCP server with an ARP reply frame generated with the same spoofed addressing to update all the ARP cache entries in the LAN (Younes, 2017). Now the attacker can control which devices use the DHCP server as a router to maintain positive control of all the traffic passing in and out of the LAN.

Lastly, a notable protocol that made it possible to expand small, isolated LANs to the internet we have today, resides in the Data Link Layer. Radia Perlman originally designed the Spanning Tree Protocol (STP) in 1985 to prevent loops and broadcast storms created in smaller LANs (Perlman, 1985). By resolving this issue, STP enabled multiple LANs to connect to the Layer 3 routers forming WANs, significantly expanding the number of interconnected devices from small LANs to the internet's worldwide connectivity today (Newnham, 2016). Malicious users commonly use the STP Root-Takeover attack to take advantage of a vulnerability caused by the STP protocol on modern networks. There has been an update to the STP protocol since its inception in 1985 to enhance resource usage called rSTP (Wojdak, 2003). IEEE later standardized it because of its low convergence times that decrease network lag in larger networks (IEEE, 2016). The change in the Spanning Tree Algorithm (STA) in the STP protocol due to rSTP made it more vulnerable because this protocol's focus was to increase resource usage rather than secure data transmission (Marro, 2003). The vulnerability is due to rSTP allowing all Ethernet LAN switches to send BPDUs (Marro, 2003).

This chapter's introduction is meant to provide baseline knowledge to understand the research problem in this study. This section provides a high-level description of the OSI model's network infrastructure, the Data Link Layer, and its vulnerabilities. It introduces the STP protocol and its vulnerability, leading to the STP Root-Takeover attack, which is the subject of this study. The following sections take a deeper dive into the the subject of this study: STP protocol and the STP-Root Takeover attack.

Background of the Study

The last section provided an overview of the Data Link Layer, and common attacks to it. It also introduces STP, a Data Link Layer protocol. The beginning of this section is a high-level overview of the STP protocol and its vulnerability that is the subject of this study.

Introduction to STP

To understand how the Spanning Tree Protocol (STP) works would be to understand the steps this protocol takes to complete its function on the switch. STP works on the Data Link Layer, also known as Layer 2 in the OSI Model. The Data Link Layer transports data in the form of Ethernet. Before the inception of the STP protocol, the Advanced Research Projects Agency Network (ARPANET) had created a LAN to support an unclassified network for US Military bases. The network later came to be called NIPR (Non-classified Internet Protocol Router), the standard internet used by the Department of State (DOD) (Evans, 2018). ARPANET had an Ethernet LAN topology that allowed for a small group of computers. Two issues that needed to be resolved were its insufficient scalability and redundant links that form loops. How does a loop manifest itself on redundant links? Two switches are set up with two connections. The second connection is a redundant link that was created as a fail-safe for the first link. However, when both links are available on the network, data meant to move from switch A to a computer connected to switch B could not leave switch B. Instead, the data ended up in a loop from switch A to switch B due to amplification (Perlman, 1985). Multiple broadcast frames echoed into the Ethernet LAN can amplify this cyclic redundancy, forming what is known as a broadcast storm. The broadcast storm is a much larger loop that also prevents the data from reaching other devices in the LAN. Due to frequent loops and broadcast storms, the Ethernet LAN network could not expand further than a manageable and small LAN (Perlman, 1985).

In 1985, Digital Equipment Corporation assigned Radia Perlman to create an algorithm to expand the LAN without the disruption of a broadcast storm. She created STP to prevent the redundant loops and change the shape of the network path, solving the redundant link and scalability issue (Perlman, 1985). Goals achieved by STA are standardized LANs for automated interconnections of LANs, allow redundancy of bridges and LANs as a fail-safe, make the LAN self-configuring, minimize memory and bandwidth usage of the STA, fast convergence, LAN bridges hold forwarding tables with topology information, eliminate loops, and automated topology adjustment for removed or failed bridges (Perlman, 1985).

Spanning Tree Algorithm

The Spanning Tree Algorithm (STA) creates a self-organizing topology that allows data to travel efficiently. It automatically creates broad network paths and expanding networks to connect to other LANs. The Spanning Tree topology is in the form of a tree with one Root (Perlman, 1985). As described in the introduction, all network devices in the Ethernet LAN have a unique identifier called a MAC address (Howser, 2020). The MAC address is also used as the unique identifier, or ID, for each STP-enabled bridge in the LAN. The STA self-configures to use the bridge with the lowest priority and lowest bridge ID in the LAN as Root. The bridge with the shortest path to the Root is called the Designated Bridge. In the case of multiple LAN segments, each LAN will have a Designated Bridge (Rai et al., 2011). In this case, each Designated Bridge is considered the Root of its LAN (Perlman, 1985). Their path distance organizes all the bridges that fall under the Designated Bridge in a LAN segment to the Root; therefore, the rest of the tree structure bridges have a longer path than their predecessors to the Root (Perlman, 1985). As part of the shortest path requirement, the Designated Bridge interface with the shortest path to the Root is called the Root Port. Among the Designated Bridges, the interface with the shortest path to the Root is considered the Designated Port (Rai et al., 2011). Every new STP-enabled bridge added to the LAN considers itself the Root and sends out messages alerting other bridges in the LAN of its ID and path cost. Once it receives a BPDU message showing a Root with a lower ID, it adjusts its forwarding table. It attempts to be the Designated Bridge until it alerted another bridge with a lower path cost (Perlman, 1985). Along with the ID and path cost, the BPDUs share the waiting time for a new message. If a bridge exceeds that waiting time, it is discarded from the topology (Rai et al., 2011).

The bridges would indiscriminately share the most efficient path to reach other network switches and avoid broadcast storms. The STP algorithm creates an automated and fluid network topology allowing for scalability with minor administrative calculation. This automation allowed the network to be self-organizing and adaptable (Perlman, 1985). Further, it allowed the network's expansion to connect multiple LANs to fulfill the ARPANET's original goal: to enable LAN to connect to any computer on the net (Roberts & Wessler, 1970). The function of STP is to compute the topology that the data will follow to reach a destination by hopping through various switches. The algorithm estimates efficient transmission by calculating the port path cost (PPC) to build the topology, hence avoiding redundancy or looping (Lai et al., 2014). The IEEE standardized a revision of this protocol and called it 802.1d. (Lammle, 2013).

Bridge Protocol Data Unit Messaging

The STP protocol creates its topology by the free exchange of Bridge Protocol Data Unit (BPDU) messages between switches. The BPDU messages are sent to every switch in the Ethernet LAN using a broadcast destination address. The BPDU frames contain a BPDU message, *conf_BPDU*, containing the Root ID, Root Path Cost (RPC) and Bridge ID (BID), and the message timing attributes. The Root ID is the BID of the Root. The RPC calculates the shortest path to the Root, and the BID identifies the ID of the bridge that is sending the BPDU. Message timing attributes are based on STP timers specified by the Root (Rai et al., 2011). There are three STP timers: *hello_time, forward_delay*, and *max_age* (Rai et al., 2011. The *hello_time* is the interval of time between *conf_BPDUs* sent, the *forward_delay* is the maximum listening and learning time allowed between *conf_BPDUs*, and *max_age* is the maximum amount of time the bridge is allowed to send a *conf_BPDU* before being removed from the topology (Rai et al., 2011). The *hello_time* calculated between the bridge and the Root is the RPC. The RPC determines the election of the Designated Bridge. The STA follows an iteration of the free exchange in the BPDU messages to determine which switch has the shortest *hello_time*, and that is how the Designated Bridge (Root for each LAN segment) is elected. An Ethernet LAN topology where more than one switch has the same *hello_time* (Marro, 2003; Perlman, 1985; Rai et al., 2011).

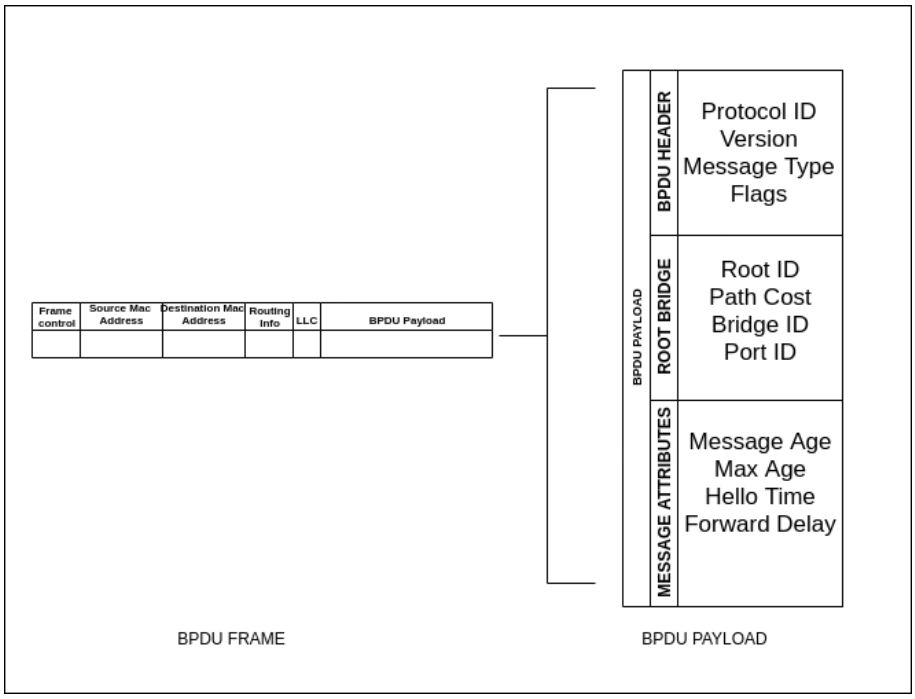


Figure 3. The Anatomy of a BPDU frame

If this topology is of an extensive network connecting several LAN segments, the Root

would be at the top connected to Designated Bridges, also considered the Root of the LANs segments (Perlman, 1985). Multiple LANs connect to the Root bridge to creating a connection between the LAN segments. Metaphorically speaking, the Ethernet bridge can be compared to a bridge that connects two pieces of land. In a topology with a Root and a Designated Bridge, the Designated Bridge has the lowest RPC out of all the LAN switches. Further, the lowest RPC determines which ports on the Root and Designated Bridge transmit data. After the Designated Bridge is chosen, the Root chooses a Designated Port which is the interface with the least RPC to the Designated Bridge. It is used to forward data to the Designated Bridge's Root Port (Marro, 2003). When a new Root is elected by the LAN bridges, they send Topology Change Notification (TCN) BPDU (*tcn_BPDU*) messages to the nearest Designated Bridge. The Designated Bridge sends this TCN to its Root port which forwards the TCN to the Root. The Root uses a Topology Change (TC) flag on all BPDU messages to the Designated Bridges to initiate convergence (Rai et al., 2011). Convergence is the STA's process to complete the original Ethernet LAN topology conversion to the revised topology with a new Root. The Root acknowledges each switch in the LAN and updates the topology on all switches to complete the convergence. During convergence, Ethernet frames are not forwarded, and the convergence time on an average Ethernet LAN is between 30-50 seconds (Marro, 2003; Whalen, M. Bishop, 2009). Perlman's (1985) BPDU messaging system, as described in this section, provides a set standard to separate Ethernet frames for data in transport from BPDUs. The BPDU is specifically linked to the functionality of STP. The next section describes a modification made to STP, creating the standard used for Ethernet LAN switching today (IEEE, 2016).

STP vs. rSTP

As the network grew exponentially, a revision of 802.1d, the rapid Spanning Tree Protocol (rSTP), began its adoption as the de-facto standard in 2001 to increase topology convergences' time efficiency (Wojdak, 2003; Marro, 2003; IEEE, 2016). Due to a demand for efficiency to fit an expanding network, IEEE standardized rSTP as 802.1w (IEEE, 2016).With overwhelming support by the industry, 802.1w became a standard to replace 802.1d, which became a legacy. After this STP vs. rSTP section, all references to STP in this study will be to the current 802.1w standard (IEEE, 2016). The main functional difference between the IEEE 802.1d and 802.1w standard is the latter has lower convergence times in larger networks to decrease lag time. In 802.1d, only the Designated Bridge could send TCN messages to each switch individually.

Efficiency in lag time, due to a shorter convergence, was made with modification to

the STA. In the original STP (802.1d), only a Designated Bridge Root Port can send a BPDU message for an official topology change (Perlman, 1985). In IEEE 802.1w, all switches send the TCN messages through the tree to the Root to decrease the time required for convergence and increase efficiency in lag time (Marro, 2003). rSTP (802.1w) introduces ”inferior” BPDUs that any switch can send with a port in a forwarding state (Wojdak, 2003). An interface configured in the forwarding state can send a BPDU and update filtering tables containing topology information (Wojdak, 2003). In STP, the Root sends TC and receives TCN BPDU messages. The rSTP modification has one type of BPDU with flags containing state information of the BPDU's source (Wojdak, 2003). In rSTP, Ethernet LAN switches can self-configure by switching neighboring switch interfaces to a forwarding state, allowing them to adjust their own filtering tables that hold information on all topology BIDs and port states (Wojdak, 2003). The original STA had the Designated Bridge send out periodic $hello\_time$ messages to check the availability of other LAN devices. The rSTP modification allows all LAN devices to remove a device and adjust the filtering table accordingly to speed up convergence times for larger networks. This section describes the difference between STP and rSTP and how the change in the standard also allowed for a substantial modification to the STA described earlier. STP was created when LANs were smaller and localized. A need for fast convergence times came at the time of the Internet revolution which connected more devices to metropolitan area networks. The larger network infrastructure required faster convergence times to work efficiently (Wojdak, 2003). Unfortunately, such resource efficiency came at a cost to security as discussed in the following section which outlines the security flaws in STP to include those manifested by the 802.1w incarnation of STP.

Security Flaws in STP

The previous section describes the difference between the original STP and STP. This section reveals the security flaws in the current STP standard. Moving forward, all references to the current STP standard are based off of the IEEE 802.1w de-facto standard (IEEE, 2016). Marro (2003) identified three points of vulnerability that originate from the functionality of STP (Lai et al., 2014; Marro, 2003). It is important to note that according to the Review of Literature; Lai et al. (2014) last acknowledged these vulnerabilities in 2014. Because the IEEE 802.1w standard for STP remains, a correct assumption was made in the proposal defense that such vulnerability points still exist in the current network infrastructure. For this reason, the initial test verified that these vulnerabilities still exist (Lai et al., 2014; Marro, 2003). A high-level overview of these points of vulnerability begins with the trust-based model of the STP protocol.

802.1d and 802.1w are derived from a trust-based STP protocol which does not require the validation of the BPDU messages (Lai et al., 2014; Marro, 2003). Second, due to the increase in switches broadcasting BPDU messages in 802.1w, STP allows for more network messaging during a state change, resulting in a Denial of Service (DoS) due to BPDU message flooding (Lai et al., 2014; Marro, 2003). Lastly, the Root is not monitored if there is a change in topology, so the rest of the network may not detect the wrong Root has taken over after a convergence regardless of how quickly it occurs (Lai et al., 2014; Marro, 2003). Since STP allows all switches to send BPDU messages, any bridge can elect, the Root role if the network's switches detect the Root Path Cost (RPC) lower in a different port or switch. This allows an attacker to eliminate the old Root and gain full access to data transported through multiple LAN segments previously connected to the old Root. In his introduction to rSTP, Wojdak (2003) describes a *point-to-point link* that allows switches to control forwarding states of other switches. Topology changes in STP occur when another switch is elected as the Root because it has a lower priority and ID in the BPDU. When a different Root takes over, the BPDU messages sent to notify the switches of the topology change do not undergo any validation. The adoption of 802.1w increased the STP protocol's vulnerability by allowing these BPDU messages to be sent among LAN switches allowing topology control outside of the Root. It also allows any LAN switch to access and change filtering tables which provides a larger surface for attackers to access and manipulate topology information. Lastly, the removal of any LAN device when it does not respond for a specified amount of time was controlled by $hello\_time$ messages sent from the Root. In 802.1w, any Ethernet LAN switch has full topology information to include routes to the Root (Wojdak, 2003). Access to this filtering table can allow an attacker the privilege to remove switches and stateful interfaces in an Ethernet LAN. STP's trusting nature assumes that BPDU messages sent by any Ethernet LAN switch are valid and come from a good source. These vulnerabilities set the stage for seven kinds of Layer 2 attacks: MAC attacks, STP attacks, CPD attacks, VLAN attacks, DHCP attacks, ARP attacks, and WLAN attacks (Mahmood et al., 2020; Rahalkar, 2018; Yeung, Yan & Leung, 2006; Younes, 2017).

STP Root-Takeover Attack

Due to the relevance of STP attacks in common MITM and DoS related incidents, this study focuses on a specific STP attack called the STP Root-Takeover attack. To stage the STP Root-Takeover attack, the malicious user performs a MITM to gain entry into the network. Then the attacker can send a BPDU using a new Root identifier (BID) that spoofs their malicious device as the Root (Rai et al., 2011). This allows the attacker to claim the Root role using the STP

Root-Takeover attack . After an STP Root-Takeover attack, an attacker can use this malicious device for flooding or topology manipulation (Whalen, M. Bishop, 2009). Yeung et al. (2008) identified four attack tools that malicious attackers use to perform the Root-Takeover attack on Linux Ethernet bridges (Yeung et al., 2008; Whalen, M. Bishop, 2009). The vectors for these attacks are a file called *stp.c*, which generates pre-configured BPDU frames, *Ettercap* that launches the STP attack, and *SToP*, which modifies existing BPDU messages (K. Yeung, Fung & Wong, 2008). *Yersinia* is the fourth exploitation tool available to perform STP Root-Takeover attacks by decoding the current topology and launching different STP attacks by forging BPDU messages (Lai et al., 2014). It claims the Root role by sending a BPDU with a lower ID, keeping all of the rest of the BPDU information the same. All these tools are readily available to the public using the Metasploit Framework and the Kali Linux Operating System. Both house a comprehensive collection of exploitation tools for several attacks. Unfortunately, the Literature Review shows this 2014 study as the last experiment with these tools, so this study required an initial trial experiment on the latest LTS versions of the Kali Operating System and Ethernet LAN switches.

This chapter has introduced a subsection of the network infrastructure called the Data Link Layer. It identifies the research problem and correlates the trust-based STP protocol with the root cause of the STP Root-Takeover attack. A further introduction is required for a set of encryption algorithms that have emerged to provide security to trust-based frameworks. This set of algorithms combines to form the blockchain protocol. The next section introduces the blockchain protocol and why it can provide a solution to security problems caused by a trust-based framework.

Introduction to Blockchain

The first part of the chapter provides an introduction to the STP Root-Takeover attack as central to the study. This section introduces blockchain as a solution to the research problem. Blockchain technology provides a security solution in several industries with trust-based frameworks in the current infrastructure. However, it has not fully emerged as a viable use case at the enterprise level. A viable use case is possible when the technology is adopted due to its part in economic growth from creating an efficient increase in productivity (Davidson et al., 2018). Use cases have been introduced in the supply chain, healthcare, contract management, financial services to eliminate the need for third-party oversight and manual administrative work (Lawrence, 2017). The lack of established use cases makes blockchain a grassroots technology. It remains at a research and development phase in many industries due to the economic impact of a com-

plete conversion to the new system (R. Shores, personal communication, May 15, 2019). The opportunity for a conversion would allow for secure and automated peer-to-peer systems using blockchain technology. However, due to how blockchain works and how fully integrating it will provide a radical change to any infrastructure. Cryptographic algorithms drive the technology that gives it unique capabilities that may have some relevance in securing modern-day network security issues (Gilder, 2018).

Blockchain was introduced in a white paper by Satoshi Nakamoto as an automated system that tracks transactional records with an electronic cash system called Bitcoin (Nakamoto, 2008). As Bitcoin performs as a virtual monetary unit, the blockchain records all the transactions produced with it using a combination of cryptographic algorithms. The combination of these algorithms forms the protocols that automate the blockchain processes. Together, Bitcoin and the blockchain form the Bitcoin Core. Since 2009, various privatized blockchains have been introduced in multiple industries and markets (Strawn, 2019). This study will focus on Satoshi Nakamoto's original blockchain from the Bitcoin Core and how the mechanics of these protocols can solve the security loopholes in the current network infrastructure. The original blockchain is the Bitcoin Core, driven by its three main characteristics; immutability, trustlessness, and decentralization. Due to a combination of cryptographic algorithms and hashing algorithms, the blockchain creates secure protocols that enable these three characteristics. The blockchain is immutable because attackers cannot easily change the transaction records. Any alteration made is preserved in the blockchain in the order with which the change was created (Haughwout, 2017). This form of version control provides a record of all transactions. To say blockchain is trustless means that all parties providing services to the blockchain are not trusted. Instead, they work in conjunction with a shared trust. Without the reliance of a trusted third party, validation of its functionality is maintained by a protocol called consensus. Nodes provide the resources to execute the transactions. They are anonymous and only provide computing and electrical resources to administer the automated transactions and have no hand in interfering with its automated workings. Trust between nodes is not required because the distributed nodes are worldwide, and transparency provides less opportunity due to all blockchain records' public nature. Lastly, blockchain has nodes distributed around the world with no central authority. A central coordinator or institution typically handles decision-making in a trusted process. However, blockchain is a distributed and decentralized database with no reigning authority (Davidson et al., 2018). Together, these three elements drive a peer-to-peer technology that can provide a reliable security infrastructure in an environment that has not been designed with security in mind (Gilder, 2018).

As mentioned in this study, the current network stack is not fundamentally designed with a reliable security solution due to the fragmented nature of Data Link Layer security and its gateway to compromising the entire network's security. The grassroots pioneers did not build peer-to-peer (P2P) internet technology on security (Gilder, 2018). Nodes connected to the internet would provide services in a bidirectional pattern. In the late 1980s, services would include P2P file sharing and communication services like Internet Relay Chat (IRC) (Gilder, 2018). Centralized services that replaced P2P services initialized a client-server model that relied on services from a server (Gilder, 2018). Soon, these became services provided by large institutions. The nodes receiving the services came to be called clients (Strawn, 2019). This framework's evolution offers an idea of how the current internet came to be, notably, its effects on having been built on already insecure network architecture. The introduction of cryptography to add security to network communications came about with the cypherpunks in the early 1990s. The cypherpunks consisted of cryptographers and programmers alike (Bartlett, 2016). The basis of cypherpunk ideology was privacy (Hughes, 1997). Further, privacy is brought about by autonomous transaction systems. These systems would allow the users to reveal their identities on their terms, including anonymous mail, cryptography, digital signatures, and electronic money (Hughes, 1997). Centralized services like email clients compromised privacy in the communications, including an email sent between Alice and Bob (Hughes, 1997). In *Cypherpunk Manifesto*, Hughes stated that a centralized email service has access to Alice and Bob's identities and their communications contents. Further, the network is such that the email is sent in plaintext and does not protect it from sniffing attacks. The invention of Pretty Good Privacy (PGP) led to the encryption of email services that are used today (Bartlett, 2016). The introduction of Bitcoin to society was an introduction to an autonomous electronic cash transaction system that allowed privacy in monetary transactions and is an achievement attributed to the cypherpunks under Satoshi Nakamoto's guise (G. Higgins, personal communication, May 1, 2019).

PGP, electronic cash systems, and the myriad of corporate services provided in the client-server model today now reinforce and comprise the internet and its porous network stack. Gilder (2018) stated in *Life after Google* that today's internet infrastructure is the weakest link, and its solution is a peer-to-peer-based architecture for blockchain technology (Gilder, 2018). This study is based on the trustless, immutable, and decentralized aspects of the blockchain's cryptographic algorithms to create an autonomous transaction system. The fundamental strategy to build new infrastructure with blockchain technology requires changes from the ground up. According to Gilder (2018) in his book *Life after Google*, an architecture built with security in mind needs not be focused on security. In other words, the technology upon which new network

infrastructure is built will be inherently secure and will not require a security strategy (Gilder, 2018). For example, a smartphone shopper will assume it will have capabilities, including end-to-end communication using telephone services, SMS, and the internet (Gilder, 2018). They will not say, "I want a smartphone that will allow me to make phone calls and allow me to browse the internet." Inherently secure network architecture will not require additional services to provide security (Gilder, 2018). Provided that the Physical Layer will require security at the human element, how can a secure infrastructure use the blockchain protocol at the Data Link Layer? Blockchain protocol can establish a P2P architecture from its technology to look at network attacks that take advantage of the STP protocol's vulnerability, a well-established protocol used on Ethernet-based networks worldwide, according to Gilder (2018) in *Life after Google* (Gilder, 2018). This section introduced blockchain and the ideology behind the technology. The next section introduces blockchain as a network security solution.

Blockchain as a Security Solution for the STP Root-Takeover Attack

The last section introduced blockchain and this section explores how it can secure a trust-based network protocol like STP. The trustless characteristic of blockchain is in direct opposition to the trusted nature of the STP protocol. As was mentioned before, it is assumed trust that causes the vulnerabilities with the STP protocol. To say that blockchain is trustless is to say that all the components in the blockchain's automation have no reliance on a third party. There is no hierarchical reliance of trust of one node over another to maintain a secure architecture. Due to the inherent nature of its functionality, there is no implicit trust (Wessling et al., 2018). All nodes are equally required to validate transactions using the consensus protocol. The blockchain's encrypted contents are transparent in the public record, reducing required trust in the human element and physical systems (Wessling et al., 2018). Insufficient trust in a central management system and the need for a publicly verifiable reputation system lead us to leverage public ledger techniques, including blockchain, to design routing protocols (Ramezan & Leung, 2018). Another point of contention with many solutions to the STP problem is with central authorities. When there is centralization, there is an implicit trust built into the system. A bank, for example, is a central authority that has a database that records money going into and out of a bank account. Using this bank as a central authority requires trust in the accuracy of bank account records. Bitcoin core was designed to eliminate implicit trust in the banking industry and was created to replace it with an automated and self-sustaining economic system (Nakamoto, 2009).

A centralized blockchain requires trust in a central authority to handle validation accu-

rately. A permissioned and centralized blockchain, including some privatized blockchain solutions created for institutional use, requires authentication to a central authority. These features could create a weak link, so the concept of the permissionless blockchain was borne. Carrying Bitcoin Core's blockchain's original ideology, the built-in consensus protocol in a permissionless blockchain is the basis for the efficiency and security in the blockchain system (Zhang et al., 2019). This idea sets up the possibility of doing away with authenticating new devices on the network rather than using the consensus protocol to add new nodes to the topology. In the STP protocol, the switch elected as the Root for a given Ethernet LAN maintains the switch topology for speed and efficiency. It sends BPDU messages to the rest of the LAN switches to determine any changes in the topology. Suppose BPDU messages are coming from a Root of a different identity, like a rogue switch. In that case, the other switches will automatically accept the change and will adapt the new Root using the trust-based STP protocol. A validation mechanism from the consensus protocol will ensure that all switches, including the Root, are made aware and approve of the election of a new Root in the LAN's network topology. The validation mechanism will ensure that a rogue device is not accepted as a new Root, preventing an STP Root-Takeover attack directed at the STP vulnerability in an Ethernet LAN.

STP uses BPDU messaging to determine the strength of the links between the switches. As mentioned earlier, the RPC is shared freely among trusted switches to determine the most efficient topology according to the amount of time it takes for frames to be sent from one switch to another. The efficiency check used in STP can also be emulated using blockchain technology. A blockchain has a distributed node structure; nodes broadcast data in the distributed network to add new transactions and perform continual validation similar to how switches communicate via the BPDU messages. Bazari et al. (2019) have proposed finding nodes in a blockchain with the highest performance degradation. The nodes that are considered weaker are removed from the network to increase the efficiency of blockchain broadcasts (Bazari et al., 2019). This solution reflects STP's primary function by measuring the efficiency among the switches in the network. This section describes two ways in which the STP protocol can use blockchain algorithms to circumvent trust-based validation in an STP-enabled network. One is to use use consensus for Root validation and second is the use of efficiency checks to remove dead or weak nodes in the network. The next section declares the problem statement and describes the purpose of the study.

Problem Statement and Purpose of Study

The whole chapter up until this section has been an introduction to concepts that are related to the research problem in this study. This section will narrow down the research topic with a declaration of the problem statement and the purpose of this study.

The protocols that govern the services in the Data Link Layer of the network stack are not designed with security in mind (Gilder, 2018). Consequently, the vulnerabilities incur a risk to every subsequent layer in the OSI referenced network stack (Rietz et al., 2018). To maintain the network's integrity, a security solution is recommended for this fundamental weak link. Ideally, end-to-end secure communications starting from the Ethernet frames in the Data Link Layer is recommended (Gilder, 2018; Strawn, 2019). As it stands, the STP protocol, the IEEE standard for an Ethernet network, is used to automate continual Ethernet network LAN changes to increase efficient data transmission and connect to the other Ethernet LANs and the internet. There are publicly available attacks that target the STP protocol's vulnerability (IEEE, 2018). This leads to the main problem statement: The STP service uses trust-based BPDU messaging from the Root to determine the LAN topology among switches. An STP Root-Takeover attack is possible if a malicious actor injects forged BPDU messages fashioned to look like they have originated from a legitimate Root switch. Due to insufficient validation of the BPDU messages, the rogue device is integrated into the LAN's topology, which may cause a Denial of Service due to BPDU message flooding or a topology misconfiguration by electing a malicious device as a Root (Nijim et al., 2017).

The research solution and the purpose of this study is the implementation of a hybrid system that replaces the trusted STP protocol with a proof of concept model emulating a de-centralized and trustless solution based on the Ethereum Platform's broadcast and consensus algorithms. A hybrid system integrates with the current client-server model because a complete transformation to incorporate blockchain technology would replace the existing infrastructure (Gilder, 2018). This study's research solution uses a hybrid blockchain using a small-scale, permissioned, and peer-to-peer blockchain framework as a potential solution to block attacks in the Data Link Layer and subsequent layers of the network stack. The Ethereum platform's blockchain is similar to Bitcoin Core's blockchain, described as trustless, immutable, and de-centralized. It has an Ethereum Virtual Machine (EVM) that performs tasks one at a time within the blockchain. These tasks are called Smart Contracts (Buterin, 2014). This research proposes Smart Contracts to validate the Root in the BPDUs sent for the STP-enabled LAN. The solution explores this system to secure the Data Link Layer from malicious STP Root-Takeover

attacks that inject forged BPDU frames to claim the Root and cause a Denial of Service due to malicious topology misconfiguration and BPDU flooding attacks that overwhelm the network devices with BPDU messages. The purpose of this study is designed to test whether the use of blockchain protocol-based hybrid systems can use validation to block a forged BPDU frame from an STP Root-Takeover attack. The goal of this section was to clarify the problem statement for the research in the study. It also introduces the specific purpose of this study as a solution for the declared research problem. The next section provides validation for the importance of a research solution for this problem.

Relevance of the Study

The last section defined the problem statement and purpose of study. This section discusses the relevance of this problem. As defined by the OSI reference model, Ethernet's current network infrastructure is based on the early P2P and client-server model built to connect computers worldwide. The trust-based client-server model raises concerns that can only be resolved by rebuilding a new network infrastructure from the ground up (Gilder, 2018; Strawn, 2019). This issue is of insufficient security in the design, beginning with virtualized data in the Data Link Layer to the Application Layer where the user and the computer function. Starting with the Data Link Layer, the underlying protocols that perform its services are built on a trust-based algorithm. Trust means the protocols do not perform validation on the services rendered and are open to common attacks performed by publicly available attacker tools. Further, there is no insufficient information and resources available to perform these attacks successfully (Lin et al., 2017; Strawn, 2019).

A significant number of attacks that are performed on network systems are based out of the Data Link Layer (Marconi Foundation, 2018). An attacker can gain access by passively gathering information required for an active attack using a publicly available network sniffing tool (Pingle et al., 2018). After collecting the required information about devices in the LAN, the attacker can ”spoof” the unique identifiers of a legitimate device onto a rogue device to enter the network. Without proper validation, this rogue device can now function within the LAN and perform further attacks to increase its foothold and deprive legitimate LAN users of required resources (Pingle et al., 2018). This infiltration is the gateway to compromise the Network Layer and Application layer, which extends the attacker's ability to control devices and gather information beyond the local Ethernet LAN (Gilder, 2018). Sniffing and MITM attacks are performed frequently at the Data Link Layer, which requires a solution that prevents

attacks by invalidated rogue devices that infiltrate the Ethernet LAN. Layer 2 used in all end-to-end network transmissions to include Metropolitan Network systems and Industrial Control Systems (ICS), a security solution that can prevent intrusions before they happen is vital. These systems manage data, and it is not easy to base detection and prevention on signatures and anomalies (IEEE, 2016; Lin et al., 2017). The DoS attacks and Distributed Denial of Service attacks (DDOS) attacks are high at the Data Link Layer (Nijim et al., 2017).

Using cryptographic blockchain protocols as a switch validation solution could replace the current trust-based model of the STP protocol with a hybrid trustless-based blockchain protocol. Ultimately, if encryption can prevent the STP Root-Takeover attack, it may contain other common attacks. A secure network infrastructure may be built by integrating security-based protocols in every network stack layer (Gilder, 2018; Marconi Foundation, 2018).

This chapter provides a high-level overview of the network infrastructure and a detailed account of the STP protocol and its vulnerability to the STP Root-Takeover attack. The Data Link Layer is a subsection of the network infrastructure that is prone to common attacks. In it, the STP protocol performs a critical function to maintain the Ethernet LAN. It has a weak spot that is advantageous to the cyber attacker, and the blockchain protocol provides a unique solution to mitigate to the weakness introduced in this study. The next chapter highlights the research that has been done to rectify the vulnerability that leads to the STP Root-Takeover attack. It also sheds light on various models for a secure network infrastructure with blockchain.

Chapter 2

Review of Literature

The previous chapter was a deep dive of the constructs, concepts and their applications which are relevant to the study. The introduction provided a high level overview of the current network infrastructure and where the security problem begins. The STP Root-Takeover attack is but one example of the opportunities that cybercriminals have to gain access to an Ethernet LAN. Most cyberattacks target the Ethernet LAN because it is a way to gain access to information and systems that form internal network. Access to this internal network can begin with infected email attachments, browser attacks or a direct connection to the network. Both of these avenues provide an opportunity for successful attacks because of vulnerabilities in the Data Link Layer, the subsection of the network that forms the Ethernet LAN. Typically, these attacks go unnoticed because they begin with passive information gathering to provide what is required for more aggressive attacks (Rietz et.al, 2018). The Data Link Layer represents a small piece of much larger problem because its vulnerability provides a gateway for unauthorized users to gather information and take control of a home or enterprise network and all of the computer systems connected to it (Younes, 2017). The subsection of the network infrastructure defined in the research problem is the Data Link Layer and it's specific trust-based service is the STP protocol. Several attacks on the STP protocol are introduced in the previous chapter, however the STP Root-Takeover attack is the center of this study as it fine tunes the perimeters of the research solution and its defense (Lai et al., 2015). Having introduced a research problem in Chapter 1, this chapter will document the literature review conducted for this study. The literature review starts with an overview of the conceptual framework used for searching literature as a part of the literature review process. After this, a description of the research problem as it relates to the research solutions is introduced. It is followed by an analysis and critique research solutions based on positions taken by the researchers. The research solutions can be divided into three main categories: validation based on cryptography, Alternatives to the STP protocol and blockchain-based solutions. The literature review is guided by the theoretical framework introduced in the next section.

Theoretical Framework

The previous section outlines the Review of Literature chapter and introduces the philosophy behind a literature review. The theoretical framework is a guide for reviewing current research in a study. It is a conceptual model that provides a structured framework to create an understanding of the research problem as it stands currently (Connelly, 2014). The theoretical framework for the research study is structured by a six-step literature review process which begins with the selection of a study topic (Machi & McEvoy, 2016). The first chapter provides an overview of the STP protocol, the STP Root-Takeover attack and blockchain as a possible solution. To fine tune the topic of this study, a research topic question must be developed. The research question in this study is the following: Can a blockchain be used to mitigate an STP Root-Takeover attack? To fine tune the topic of the research study is the first step to a complex literature review based on the chosen topic (Machi & McEvoy, 2016). The second step in the literature review process begins with the argument of discovery and its goal is to present the current state of knowledge on the subject. This goal is achieved by gathering information related to the topic and analyzing it. The *argument of advocacy* is a critique of the knowledge gained and applied to the research question (Machi & McEvoy, 2016). The third step is to select literature to review which is based on the chosen topic and the type of material that will best present the arguments required as evidence for the claims made in the study (Machi & McEvoy, 2016). The choice of resources is a decisive action that provides an opportunity to defend claims made in the study. For example, academic debates, professional experience and expert opinion can be used collect current knowledge (Machi & McEvoy, 2016). The research topic for the study is based on global knowledge that has been collected over years. A literature review that is conducted based on the search of peer-reviewed academic literature is a known source of expert opinion. The search was fine tuned to prefer literature within the last five years due to the fast paced nature of information technology. Further, a preference was made to books, guides, conference proceedings and academic journals based on information technology for the literature review in the study. Both of these selections will ensure current knowledge in the topic. The databases that comprise the literature used in this study provide validated sources of research. The integrity of a research study is based on validated research because uses verified research methodology. This study is based on research methodology so it is natural to base it off of the research and recommendations of scholarly research.

A significant part of scholarly research is the peer review. Peer review is a form of validation used in scholarly research through the verification of research methodology. It is a

critique of the research made anonymously and/or by experts in the same field of study. Peer review can be seen as similar to the validation methods used in peer-to-peer technology of the blockchain. The trustless validation provided by peer review provides integrity to scholarly research. The search for literature was divided into two subjects: security research in the STP protocol and network security research in blockchain. The goal is twofold: to assess the research for underlying commonalities and critique the research solutions provided in both subjects against the research solution in this study. These two goals are defined by Machi and McEvoy (2016) as a *discovery argument* and critique of the literature; steps four and five respectively. The final step of the process is this chapter which is a written documentation of the literature review. This section described the theoretical framework as a guide for the literature review. The next section describes the research question and introduces current solutions that pertain to it.

The STP Root-Takeover Research Problem

The literature review in this chapter explores current solutions, designs and models that have been introduced to mitigate the STP Root-Takeover attack. Various ideas to this problem have been introduced and a few are implemented on a small scale. The STP Root-Takeover attack is an example of how existing vulnerabilities in the Data Link Layer can provide an unauthorized user control of an Ethernet LAN. The specific vulnerability in the Data Link Layer that allows for the success of the STP Root-Takeover attack is the STP protocol. The STP protocol manages the topology of an Ethernet LAN using trust based messaging among network devices (Lammle, 2020). The lack of validation in a trust based system has no mechanism to prevent a rogue device from masquerading as a valid Ethernet device (Singh et. al., 2018). The subsection of the network infrastructure defined in the research problem is the Data Link Layer and it's specific trust-based service is the STP protocol. Several attacks on the STP protocol are introduced in the previous chapter, however the STP Root-Takeover attack is the center of this study as it fine tunes the perimeters of the research solution and its defense. Although serious vulnerabilities in the Data Link Layer cause MITM attacks, MAC spoofing and sniffing attacks, the study will be centered upon an aggressive attack that provides a solid example of why trust based systems such as the STP algorithm require revision. The research problem is validated by solutions produced to prevent STP Root-Takeover attack.

The STP Root-Takeover attack is based on the STP protocol which is designed to automate the expansion of LANs. Perlman (1985) defines the goal of the STP protocol to create a self configuring topology that can interconnect LANs and avoid loops (Perlman, 1985). When

the algorithm, Perlman (1985) described of the characteristics of this protocol:

- The topology forms a spanning tree with a unique Root and the predecessors of all of the bridges in the tree are closer to that Root
- The Root is elected based on having the lowest ID which comes from its unique hardware address. The topology is initially created by each bridge electing itself as Root.
- The Designated Bridge has the lowest path cost to the Root. The lowest path cost can be calculated by the amount of bandwidth time it takes to send a message to the Root.
- The Designated Bridge is responsible for sending out *hello_time* messages that track path cost to the Root and check for non-functioning bridges in the network
- Bridges in the topology that do not respond are removed from the topology.

The goal of the STP protocol was to expand the network , connect LAN segments, detect and mitigate redundant links and prevent loops (Perlman, 1985). At the time of its inception in 1985, the requirement for security measures was not evident and the trust-based STP protocol became a de-facto standard for network expansion (IEEE, 2018). The STP Root-Takeover attack in this study is performed by disrupting the election of the Root by the STP algorithm in an Ethernet LAN. The network devices in the Ethernet LAN will typically pass messages called bridge protocol data units (BPDU) that name the current Root and provide other identifying characteristics such as the *hello_time* and RPC. The Root is elected by having the lowest ID, therefore, the attacker infiltrates the Ethernet LAN and sends a forged frame with the lowest ID to claim the Root role. The new Root is accepted and acknowledged by all of the Ethernet LAN devices to create a new topology. In claiming the Root role, the attacker has the ability to manipulate data in the Ethernet LAN, manage connected LAN segments, perform DoS attacks and reroute all traffic to their own device. This attack is possible because the network devices in the Ethernet LAN do not have a mechanism to validate the source of a forged BPDU frame. The lack of validation makes the STP protocol trust-based which is the root cause of its vulnerability to this attack. The following three sections divide the solutions into three main solution categories: cryptographic, alternatives to STP and blockchain-based. The next section introduces the use of cryptography as a means for network device validation.

Cryptographic Validation of Ethernet Devices

The last section describes the research topic central to the Review of Literature chapter. This section introduces the first category of solutions reviewed and analyzed. The literature

review produced two interesting cryptographic solutions that are used to add validation to the trust-based nature of the STP protocol. As it stands, the trust based STP protocol does not validate new devices that are added to an Ethernet LAN. Although the STP protocol was adopted to solve the issue of expansion, the issue of security in the Data Link Layer is a problem (Lai et al., 2014). The use of cryptography provides proven and secure techniques used to handle issues of validation (Aumasson 2017). These techniques can be integrated into a network protocol like STP to handle the validation of new network devices and solve the problem of trust. These solutions use a different forms of cryptography that provide unique verification techniques for STP switch validation. The solutions described in this section are public key cryptography and hashed message authentication codes.

Public Key Infrastructure for Validation

Shortly after creating the STP algorithm, Perlman (1988) introduces public key infrastructure (PKI) to a network layer protocol design that validates network layer devices by verifying the source device's digital signature. PKI provides validation by verifying the source of digital information through public and private keys (Aumasson, 2017). The public key is shared among all network devices and verification of a specific device's identity is made using it's unique private key. The public key is required to verify the private key, therefore, all valid devices in a network using PKI validation would have a public and private key pair (Aumasson, 2017). The use of two different keys, one to encrypt and the other to decrypt, makes it a form of asymmetric cryptography (Aumasson, 2017). Although the design that Perlman (1988) suggested does not provide a solution to the trust-based nature of the STP protocol in the Data Link Layer, it recommends the use of validation for the trust-based nature of a network protocol in the network layer (Perlman, 1988). The introduction of PKI for trust-based validation a few years after the STP protocol was introduced, provides some insight into how security was viewed at the time of the network infrastructure's inception. Perlman's (1988) thesis validates that the trust based nature of network protocols in the infrastructure was a problem, however, the integration of PKI cryptography for validation was not fully realized in the STP protocol.

A PKI solution was produced by Lai et al. (2014) to validate devices in an Ethernet LAN using the STP protocol. The solution is trust based in that the Ethernet LAN is made up of switches called *trusted switches*. Each switch, upon joining the LAN, must obtain a certificate to distribute BPDU messages. This certificate is obtained by a LAN-based certificate authority (CA) with the public key. The CA is a trust-based issuer of certificates. It works by validating a switch and providing it an encrypted certificate as proof of attestation (Boonkrong, 2020).

The certificate is only distributed to switches that have an Attestation Identity Key (AIK) which is the private key that all valid switches have before joining the LAN (Lai et al., 2014). This solution performs validation of switches by providing each of them with a unique identity. The attestation of this unique identity is made by the LAN's trusted CA that issues it a certificate. This solution is able to provide validation of the network device and stop a switch that has not been provided with a valid AIK (Lai et al., 2014). The research shows that the solution blocks a rogue switch in the Ethernet LAN from sending BPDU frames because it does not have a certificate. The researchers launched an attack using the Yersinia attack tool to provide evidence that this solution blocks frames from a switch that has not been provided a certificate to validate it on the Ethernet LAN (Lai et al., 2014). While this solution effectively blocks a forged BPDU frame from allowing a rogue device to claim the Root role, there are two factors that make the solution difficult to implement. First, it requires that all switches have a chip that provides its private key. Second, the certificate is granted by a trusted CA that verifies the chip. Although these two components provide effective security, it is a proprietary system that would require agreement on a common CA and the ability to work with the additional hardware chip on every network device. This issue can burden its expansion to a large scale network infrastructure.

Hashed Message Authentication for Validation

An alternative to the asymmetric PKI solution is the hashed message authentication code (HMAC). Message authentication code (MAC) is a symmetric form of cryptography that uses a single shared key. It is important to note that this type of a MAC is different from the MAC protocol described in Chapter 1's introduction and the unique hardware address used through the study. The type of MAC described in this section is related to a type of symmetric cryptography that requires more trust than the asymmetric PKI. The MAC does not use unique and unshared private keys, it is an alternative algorithm that provides message integrity and authentication (Fall & Stevens, 2011). The HMAC uses an additional hash algorithm to add to its strength (Krawczyk, 1997). A study by Whalen & Bishop (2009) produced a solution that uses a "password-keyed HMAC". Each network device in the Ethernet LAN will be programmed to create a hashed password based on the HMAC algorithm. It will add this password to the BPDU payload in the unused digest field of the BPDU frame (Whalen & Bishop, 2009). The receiving device will recompute the password using HMAC and make a comparison check for validation of the BPDU frame. This solution is tested on devices that are kernel compiled to use the HMAC algorithm verification mechanism. The solution in this study introduces the use of HMAC as a form of verification to remove external factors such as certificate authorities and key

exchange (Whalen & Bishop, 2009). Rather, it is a precompiled and integrated authentication mechanism to validate BPDU frames. This solution is lightweight and provides a proof-of-concept for resolving issues with attestation in the STP protocol, however, the use of the HMAC algorithm as a standard is difficult to attain. A problem with the use of the shared password is that it would be difficult to keep secret and requires trust. Such a password would have to be configurable to change by organization and may cause overhead for local administration. Although the solution is more efficient than the CA, the amount of trust required in the expansion of this protocol addendum would create another vulnerability. Moreover, the amount of time required to perform HMAC encryption calculations and add them to each Ethernet frame may cause a lag in the STP protocol's performance (Rai et al,. 2011). The HMAC solution concludes the review of cryptographic solutions. The next section introduces four unique alternatives to prevent STP Root-Takeover attacks.

Alternate Solutions for STP Device Validation

The previous section provides an overview of cryptographic solutions that have been proposed to alleviate the threat of an STP Root-Takeover attack. The use of PKI and HMAC are common mechanisms for authentication and access control that are useful in alleviating the risk of layer two attacks (Whalen & Bishop, 2009; Boonkrong, 2020). This section dives into solutions that find results in infrastructure changes that manipulate how the STP algorithm is rendered on the Ethernet LAN. The literature search through academic journals have provided alternatives such as a vendor-specific Cisco solution, LAN partitioning, an intrusion detection system (IDS), and a software defined network (SDN) solution.

Cisco stops the STP Leak

The Cisco solutions include three widely used configurations in the Cisco IOS to block the STP protocol's automated functionality. These configurations are called *Root Guard, STP Portfast, BPDUguard, STP loop guard* and *BPDU filter* (Cisco, 2007; Gooley et al., 2019). *Root Guard* blocks switches from Root election, requiring the manual configuration of the Root instead of using automated topology changes for Root election . *STP Portfast* can be configured on switch interfaces to prevent the topology change notification (TCN) messages that alert all devices of a new Root. This prevents the switches from automatically changing the topology's Root based on the BPDU messages (Gooley et al., 2019). *BPDUguard* blocks BPDU messages on individual switch interfaces which prevents them from connecting to rogue switches (Singh,

2020). It is an additional safety mechanism that is administered globally on the switch should a BPDU frame slide past the *STP Portfast* configuration (Gooley et al., 2019). *STP loop guard* prevents loops when a link fails a transmission because loop prevention is the original job of the STP protocol. Without the STP protocol, an Ethernet LAN is vulnerable to loops and broadcast storms that may prevent the traversal of data to other devices in the network (Perlman, 1985). Although a fix may have been added in recent years, the *STP loop guard* and *Root Guard* cannot function synchronously on a single interface allowing the switch to be elected as Root and start a loop (Yeung et al., 2008). It is also important to note the *BPDUGuard* and *Root Guard* may be helpful in preventing an STP Root-Takeover attack, but leave the Ethernet LAN vulnerable to other attacks (Lai et al., 2014). Also, this solution is vendor-specific, so it can only be used with switches configured with Cisco IOS. Lastly, this Cisco solution does not resolve security issues in the STP protocol. Instead, it blocks it from working by using so many prevention mechanisms its like trying to prevent a water balloon from leaking by patching each hole. Finding an efficient topology with BDPU messages and blocking loops in redundant networks is left to the administrators in charge of network maintenance because the STP protocol is deterred from its function (Rai et al., 2011). Manual topology configurations and monitoring efficient path cost is heavy work due to insufficient automation from the STP protocol (Lai et al., 2014; Rai et al., 2011; Yeung et al., 2006).

Spoof the Spoofer with a Partitioned Infrastructure

While a common enterprise solution to the STP vulnerability has been maintained with messy Cisco configurations, another solution to prevent Layer 2 attacks at the infrastructure-level had emerged. In a typical STP Root-Takeover attack, a forged BPDU frame is created by gathering topology information from sniffing attacks . Sniffing attacks, although passive, are deadly because it weaponizes the unauthorized user with data required to launch aggressive attacks (Forshaw, 2017). Yeung et al., (2006) had proposed a system that will hide topology information in BPDU messaging. This solution is implemented by partitioning an Ethernet LAN into two: internal and external. The internal non-infrastructure switches and the external infrastructure would be organized with a gateway switch in between. The non-infrastructure LAN switches would use pseudo bridge ID and pseudo Root IDs to hide basic topology information to prevent malicious attacks from the internal LAN (Lai et al., 2014). The infrastructure partition would contain STP enabled switches in the network outside of the LAN. This solution provides a secure network infrastructure by using a common strategy in stronger control of the network: the demilitarized zone (DMZ). In the field of network security, the DMZ is a perimeter

created for added security with an inside that is protected from the outside (Davies, 2019). The solution is creative because the STP algorithm's functionality is not altered or blocked as it is with the Cisco solution. A possible problem with the modified and nonstandard STP protocol implemented for the internal LAN, is that it will require administrative adaptation making it feasible for a smaller network rather than a larger one (Rai et al., 2011). This solution can prevent information gathering from an internal attack, however, the outer infrastructure is vulnerable to the STP Root-Takeover attack. This vulnerability may provide the gateway to a targeted effort for spoofing the pseudo IDs to match the internal switches. Further, it is unclear whether a spoofing attack using the pseudo IDs can be prevented in the event of a successful man in the middle attack on the external LAN and the gateway switch.

The IDS/DIDS Solution

One of the biggest problems with Layer 2 exploits such as the MITM or the STP Root-Takeover attack is that they often go unnoticed unless the attackers make themselves known by launching a DoS attack or until it has affected the layers above it (Lai et al., 2014). The role of an intrusion detection system (IDS) is to use common attack patterns to detect a possible attack (Shrivastava, 2020). The IDS will inspect all network traffic in search of an anomaly (suspicious network traffic). Upon detection of an anomaly, the IDS will typically send a warning in the form of an email or alert that is received by an administrator (Cox & Gerg, 2004). With respect to the STP Protocol vulnerability at hand, an IDS solution has been proposed to prevent an STP Root-Takeover attack by inspecting traffic in the Ethernet LAN for anomalies. The IDS in this solution targets the network traffic passing through the switch interfaces (Rai et al., 2011). Any switch can receive a BPDU message with a lower ID and update its database electing the the switch with the lower ID as the new Root. This newly elected Root is broadcast as a BPDU message to all of the switches in the Ethernet LAN. In turn, the rest of the switches in the LAN will accept this new Root and update their databases (Rai et al., 2011). The solution proposes that an IDS can be added to all or a set amount of switches in the Ethernet LAN. This IDS will have an algorithm called a *Root_Change_Handler* that detects BPDUs that indicate a new Root. It uses other verification algorithms to detect malformed BPDUs and anomalies in the topology's STP timing calculations from the BPDU payload (Rai et al., 2011). The IDS sends a probe to the new Root for an acknowledgment with the topology's state information to verify it. Rai et al., (2011) maintain that if the STP Root-Takeover attack is administered from a rogue switch, it would not have the stateful information required to acknowledge the probe, thus triggering the detection alarm (Rai et al., 2011).

This IDS solution has algorithms that get triggered from the common STP Root-Takeover attack allowing for a successful detection. Moreover, Rai et al., (2011) have taken measures to ensure less false positives by considering the stateful information collected about the topology. However, if the attacker collects the stateful information by listening for a longer period of time, it may be able to mimic the thresholds calculated by the algorithm, allowing it to spoof its way into the Root role of the Ethernet LAN. Also, an IDS is intended to alert administrators of a possible issue. The problem with this type of a system is that alerts are often ignored due to the frequency of false positives (Cox & Gerg, 2004). Since the detection system does not stop the activation of the new Root role, the attacker may perform more aggressive attacks before administration is able to identify and prevent them from occurring (Ljubuncic & Litterer, 2019).

A Software Designed Network Approach

The typical Ethernet LAN, as described in Chapter 1, historically started with hardware switches that were connected by hardware media. The hardware network infrastructure is slowly being replaced with virtualized and software based systems that provide centralization of network device controls. In a hardware system, each switch in the Ethernet LAN is described as autonomous with varying configurations based on network requirements. A software designed network (SDN), centralizes the administration of network devices (Pujole, 2020). A hypervisor, or virtualized environment, allows for an Ethernet LAN to made completely of software and network traffic can be centrally processed (Dotson, 2019). For example, the previous IDS solution required that a certain number of switches to be configured as an IDS (Rai et al., 2011). The IDS solution in a virtualized environment, would allow centralized network traffic inspection through one IDS instead of having to configure it on every switch. The evolution of software based systems in virtualized environments has also brought software based networks to the forefront (Pujole, 2020). SDNs have begun to pave the way for rapidly introducing and changing technology used without the hassle of expensive hardware upgrades (Santos, 2020). To keep with the current STP Root-Takeover attack issue which still plagues modern day virtualized technology, a solution for the trust-based STP protocol to validate devices still exists (Lai et al., 2014; Rietz et al., 2018). Rietz et al., (2018) assert that due to STP vulnerabilities, attackers can still gain access to the Layer 2 despite Layer 3 detection systems on the modern software based network infrastructure (Rietz et al., 2018). They claim that an SDN solution can protect a network from STP-based Layer 2 & 3 attacks without changing the configurations of the Ethernet LAN devices (Rietz et al., 2018). An SDN can provide a separate controller that controls network traffic without the administrative overhead of configuring each device. The

controller can provide administrative control of Ethernet frames with rules to provide secure switching (Rietz et al., 2018). The SDN solution limits broadcasting to control network flow. Broadcasting, as was defined in the introduction, is Communication using a broadcast destination address to send a message to all the Ethernet LAN devices. Since such a broadcast address echos the broadcasted frames to all of the LAN devices, DoS attacks after claiming the Root role in the STP Root-Takeover attack can limit system availability to the victim (Easttom, 2018).

Although this solution does not address the STP Root-Takeover attack specifically, the solution provides for firewall rules that can isolate BPDU frames that show a change in the Root role and send them to a centralized SDN controller for administrative control. This will prevent the quick adoption of a new rogue Root bridge and allow a process for inspection and the ability to manipulate the network traffic. The solution is a model that provides network inspection that can prevent STP based attacks as well as other Layer 2 & 3 attacks on other protocols (Rietz et al., 2018). Although broad, this type of a solution can be fine tuned to prevent many attacks. This type of a design does not show experimentation in production, however, introduces a paradigm shift in how network security is maintained. If network security solutions based on SDNs provide a Layer 2 frame inspection and control, the system will not be vulnerable to Layer 2 attacks which lead to attacks in higher layers (Rietz et al., 2018). An improvement to this solution would be an artifact that can demonstrate frame and packet inspection through the centralized SDN controller. Currently, this solution does not exhibit real evidence of preventing an STP based attack and would be stronger with a software based solution that provides empirical data.

So far, the literature search and brought forth cryptographic solutions that demonstrate modifications to the STP protocol to provide validation of newly elected Roots using PKI and HMAC. Alternative solutions have provided other formats to manipulate the STP protocol from automatically validating Roots. Three out of these four alternative solutions are dated and do not necessarily follow the evolution of the modern network infrastructure. They are solutions that work on hardware based networks and solve issues that encompass hardware and software based networks creating more overhead and expensive adoption (Pujole, 2020). These three alternatives are:

- The Cisco solution blocks and manipulates the implementation of the STP protocol at a granular level.

- A partitioned infrastructure that contains an internal and external LAN by hiding the topology information.

- An IDS that is installed on each STP enabled switch in the LAN that detects anomalies.

The fourth solution follows the adoption of software based system infrastructures that have created a new paradigm in network and system architecture (Pujole, 2020). The SDN controller in the fourth alternative solution provides a software based approach to a centralized network traffic flow. As virtualized environments with an infrastructure built on hypervisors start to hit more enterprise systems, the advancement leads to approaches that work in cloud based architectures. These emerging technologies lead the average consumer towards approaches like the SDN (Abuelenain, 2021). To carry the research solution to new technology and software-based solutions, the introduction of blockchain technology as a medium for security based solutions provides a stronger shift. The second part of the literature search points to blockchain solutions to network security problems such as the STP Root-Takeover attack in the research question.

Blockchain solutions for the STP Root-Takeover Attack

The last section was an overview of trends in security solutions to resolve vulnerabilities in the standardized STP protocol. It poses as a survey of solutions that follow the trend in enterprise network infrastructure. The most dated solutions provide cryptographic validation for network devices in the Ethernet LAN. Other solutions that were proposed various changes to the STP protocol that can work in the traditional hardware based Ethernet LAN (Cisco, 2007; Rai et al., 2011; Yeung et al., 2006) . An SDN solution introduced the power of software based network traffic inspection to solve a host of Layer 2 & Layer 3 vulnerabilities (Rietz et al., 2018). To push forward with the SDN solution, such a strategy can be experimented and tested using blockchain technology. Blockchain technology is software that provides unique security elements such as consensus (validation), distribution (cloud share), encryption and hashing to its use cases (Lantz & Cowrey, 2020). Blockchain is at a grassroots level so it is difficult to produce specific use cases that deliver empirically proven conclusions. All use cases are considered startup development because businesses that propose using it are in a research and development phase (R. Shores, personal communication, May 15, 2019). Network security solutions have been introduced in startup development using blockchain as an SDN for encrypted traffic flow in a distributed cloud. An SDN that is implemented in a distributed network would not have a single SDN controller as was described by Rietz et al.(2018) but is maintained by a distributed set of SDN controllers that manage the network nodes (Amin, 2017). This means that all of the network traffic will be managed by a distributed network of nodes. Nodes are individual sys-

tems connected to a distributed cloud network that provide resources such as managing the path of network traffic (Raj, 2019). The basis for a blockchain system is that it it provides validation and consensus through a distributed peer to peer system (Bashir, 2020). Each node is a peer and provides resources to keep the blockchain network running. This construct allows for an automated system that is not governed by a central authority (Bashir, 2020). To maintain Byzantine fault tolerance (BFT), each node is incentivized (Bashir,2020). BFT is a concept that ensures that all nodes in the network maintain a single version of the blockchain network (Bashir, 2020). The name is derived from the Byzantine General's Problem which is the idea that all general's in the army must maintain a majority vote in a single plan to attack a city. The problem lies in how to keep each general from going astray (Raj, 2019). BFT is the solution that ensures that all of the generals in the army stick to the same plan. Consensus is the BFT solution to maintain a distributed blockchain network where each node agrees on the the contents of the blockchain (Raj, 2019). Typically, the agreement would be made through Proof of Work (PoW) or a general majority agreement on a hash value. PoW is used in the Bitcoin Core network which requires intensive hash calculations to be rewarded for adding a block to the network (Bashir, 2020).

Blockchain-based Contractual Routing Protocol

Ramezan and Leung (2018) stated a standard blockchain solution that uses its entire technology in its architecture uses incentives and rewards for users that provide resources to maintain a decentralized infrastructure. The peer-to-peer network of distributed nodes add no cost to a central authority so it is independent of central management. The proposed solution uses a blockchain network with nodes that accept payment for routing data packets through consensus. Routing of data packets involves a path selection for the data to follow from source to destination in the network (Zinin, 2001). The proposed model is called the Blockchain-Based Contractual Routing (BCR) protocol. The BCR protocol maintains a blockchain network that connects IoT devices together allowing a peer to peer distributed system using smart contract routing. The lack of trust in the vendors of IoT devices triggered the creation of this model (Ramezan & Leung, 2018). Ramezan and Leung (2018) proposed a multi-hop network with a gateway that provides the authority on a network by enabling communication, approving new devices and uploading data and firmware required for the IoT devices to function (Ramezan & Leung, 2018). Transactions are added to the blockchain to incentivize miners with tokens that later increase in value (Ramezan & Leung, 2018). Ramezan and Leung (2018) acknowledge that attackers in the blockchain network can maliciously disrupt the routing of data packets by providing the wrong route or dropping data packets. They run a simulation of the smart contracts

on a network and test whether their consensus algorithms can lower the amount of attacks.

The BCR protocol provides a different perspective to network traffic control. It allows for validation using smart contracts and consensus algorithms. Although this solution does not specifically gear itself towards the Layer 2 STP protocol vulnerability, it does present a solution that provides an alternative software based network traffic manager. Although this solution manages the routes of data, it requires consensus verification that manages the source, destination and path of network traffic. This construct can be used to manage Ethernet traffic and use consensus algorithms to deter malicious nodes. The model creates a typical blockchain solution that uses a decentralized peer to peer network. The problem with creating a standing solution for the STP Root-Takeover attack is that it must adapt to the current network infrastructure. The BCR protocol calls for a redefinition of the network infrastructure whether it is used to connect IoT devices or manage an Ethernet LAN. Secure transmission and verification adds the blockchain security aspect to the solution, however, its implementation is not realistic on the network infrastructure as it stands today. The next option is based on the underlying security issues with the current network infrastructure and has specific solutions for securing the Data Link Layer.

Marconi Protocol

The premise of the Marconi protocol is based on three problems addressed about the current network infrastructure: (1) Ethernet frames are not designed with encryption exposing data transported in Layer 2 (2) The inflexibility of the current network infrastructure causes an additional expense due to the requirement of security add-on features (3) The centralized model of the current network infrastructure leaves availability up to the internet service providers (Marconi Foundation, 2018). The following features are provided in the whitepaper by Marconi Foundation:

- The Marconi protocol proposes a communication channel called a Marconi pipe for the transport of network traffic among peers (Marconi Foundation, 2018).
- The encryption of network traffic begins at layer two with Ethernet frames to ensure end to end encryption for all data in the network infrastructure using Diffie-Hellman between peers in the blockchain network.
- The Marconi peer to peer distributed network is made up of nodes that are awarded with tokens for providing bandwidth and resources.
- The model is said to be self-sustaining and can work in conjunction with the current

network infrastructure.

- Packet inspection can locate anomalies despite the encryption.
- Packet relays will provide untraceable routing paths.
- Peer ranking strategies are used to ensure the quality of packet management and speed.

The Marconi Foundation set up a model for a network infrastructure that follows the ideology from which blockchain is based. It ranks privacy and decentralization as a priority for a sustainable network infrastructure starting from the ground up. As an SDN, its a perfect blockchain solution has been created to provide security starting at the Data Link Layer. It is a platform through the Ethereum blockchain that has redesigned encrypted packets and provides a new infrastructure to build new Decentralized Applications (DApps). The solution is an example of how blockchain can provide a new infrastructure using protocols that provide encryption built into the Data Link layer providing a way to transport data and ensure privacy (Marconi Foundation, 2018).

Unfortunately, the current network infrastructure runs on a centralized client-server model that is so integral to the underlying system, that such a network would not easily run side by side with the current system as the Marconi Foundation claims (Gilder, 2018). Such a system can only be built from the ground up and maintained if a need for a new network infrastructure arose. This circumstance seems inconceivable which is why its feasibility for success as a a replacement for the current network infrastructure is low. A significant issue in transforming current problems to blockchain solutions is the rate at which the existing infrastructure can handle the change. The worldwide infrastructure and acceptable IEEE standards are built upon standards that will take years to change. A viable use case to fix the current infrastructure is to integrate a hybrid system that will work with current acceptable technology but sacrifice some blockchain functionality. The next solution is built on a software based network infrastructure. The technology for this infrastructure is currently in use and may provide more insight into a sustainable option for the near future.

Blockchain Virtual Extensible LAN

Amin (2017) introduces a Layer 2 infrastructure based on the blockchain. It is a solution that does not require the STP protocol but introduces a new one based on the management of Layer 2 virtual LANs (VLANs) . Although this solution is not a specific amendment of the traditional STP function, it provides an alternate management of Ethernet frames that is free of the STP topology requirement and may eliminate the vulnerability from BPDU messaging. The se-

curity of separate Ethernet LANs that are connected to the same Layer 3 network device (router) can be managed by partitioning them into VLANs. Where an Ethernet LAN can be formed by a physical separation, a VLAN forms a logical separation of a network (Mathew & Prabhu, 2017). The STP protocol was created to expand the traditional Ethernet LAN by creating an algorithm that stops loops created by redundant links (Perlman, 1985). The SDN paradigm shift allows for load sharing network traffic through redundant links that do not form loops. Distributed services such as cloud technology manage the distribution of traffic through algorithms that permit load sharing (DeJonghe, 2018). In the current SDN construct, the blockchain network will find redundant links advantageous for a multipath data flow (Amin, 2017). It's goal of is to expand the scalability of these segments by using redundant links for efficient data transmission in a distributed network (Amin, 2017). The solution is called a Blockchain Virtual Extensible Local Area Network (BLAN) (Amin, 2017). It works with cloud-based Layer 2 bridges called a virtual extensible LAN (VXLAN) network and it encapsulates Ethernet frames from layer two with an four additonal headers: VXLAN, UDP, IP and MAC (C, 2016). It uses a cloud based protocol called equal cost multi path (ECMP) to load balance traffic flow. Current VXLANs work in virtualized hypervisor environments such as Vmware NSX (C, 2016). The BLAN follows this format but in with blockchain technology. As a blockchain, the VXLANs would be distributed among nodes. All MAC addresses of the network devices in the VXLANs will be distributed by a controller that maintains a list (Amin, 2017). Amin (2017) states that using this construct in a private blockchain would be "faster, cheaper and respects the organization's privacy"(Amin, 2017).

The BLAN solution introduces a blockchain to a current SDN infrastructure that includes Layer 2 & 3 technology. It is inferred that the use of STP is not required due to VXLAN bridging and the maintenance of MAC address lists using a controller in the BLAN. Typical layer 2 bridging using VXLANs in the typical Vmware NSX architecture does not use the STP standard (Vmware, 2019). If this is the case for the BLAN, the vulnerability caused by BPDU messaging will not exist and the STP Root-Takeover attack and other attacks based on the STP protocol are eliminated. The blockchain solution is a feasible solution based off of current cloud technology. The increased usage of SDN network technology according to emerging trends in data center technology (DeJonghe, 2018). This solution is based on a purely software based network using Cisco software features for blocking BPDU messaging and a Vmware environment for Layer 2 & 3 network traffic management. This environment is in accordance with emerging trends, however, it does not have a place with hardware-based Ethernet LANs. The use of the BLAN would require a transition to an SDN environment prior to adoption.

Summary

The literature search, analysis and critique has revealed a broad survey of suggested security solutions to prevent the STP Root-takeover attack. The solutions provided created three sets of solutions to provide a perspective of the current standing of the layer two vulnerability. The first subset was cryptographic solutions to provide validation of devices in the STP protocol. This is a solution that directly gets to the heart of the vulnerability in the STP protocol; the lack of switch validation in BPDU messaging allows an attacker to claim the Root role in an Ethernet LAN. This validation can be performed by PKI or HMAC techniques to verify new switches (Lai et al,. 2014; Whalen & Bishop, 2009). Unfortunately, the cryptographic solutions have not been accepted as a standard even though there is awareness of the vulnerability. The second subset of solutions reveals ideas on how to offset the functionality of STP by creating variations of it. A Cisco solution uses a granular means for blocking its BPDU messaging functionality and using another means for avoiding loops. It does not solve anything but provides a way to control the topology (Cisco, 2007). A partitioned network solution provides a means of masking topology information in an internal LAN to make information gathering more difficult in a sniffing attack (Yeung et al., 2008). An IDS solution to detect anomalies using algorithms that save topology information provides alerts, however, a lot of damage can be done by the time such an attack is mitigated (Rai et al., 2011). An IDS or Distributed Intrusion Detection System (DIDS) uses signatures and self-learning software to maintain equilibrium but many attacks bypass these mechanisms (Lin et al., 2017). Lastly, the most recent technology to match current trends in network administration is the SDN solution that allows for an inspection of all network traffic and provide control over layer two traffic inspection and firewalling (Rietz et al., 2018).

The third subset in the literature search was blockchain solutions that could provide a base for network security solutions that would block the STP Root-Takeover attack. The first option was a model for a network of IoT devices which was focused on the route management of data packets. A simulation was made to demonstrate that it could use consensus algorithms to limit attacks from malicious nodes that would provide false routes or drop packets (Ramezan & Leung, 2018). The second option was by the Marconi Foundation which provided a secure network infrastructure that provided symmetric key encryption on all data transported starting at Layer 2. It would get bandwidth from peer to peer nodes and incentives for resource sharing (Marconi Foundation, 2018). These two blockchain options do not seem feasible because they require a sustainable blockchain network to run efficiently. Such a feat could take a long time. They are also difficult to adopt because of the current network infrastructure. The last blockchain

solution called BLAN has the option for a private blockchain network that is adaptable to the current environment. (Amin, 2017). Moreover, it is built on emerging SDN technology already in existence such as Vmware NSX and Cisco STP Portfast (C, 2016). This option is the most feasible of the blockchain options, however, most networks have not advanced to an SDN type of environment. Also, the peer to peer distribution of this network can take some time to establish. This chapter was an exploration of the current standing in research for solutions to the STP Root-Takeover attack. There are several different approaches and one thing is in common: the root cause of this vulnerability is that it is based on a trust-based model of the STP protocol. Hence, all the proposed solutions looked into a technique to eliminate the election of a Root without validation.

One problem throughout the literature review is the feasibility of integrating the new solution into a well-established network infrastructure. The issue of integration leads to a hybrid solution that would not require a complete upgrade of the current network infrastructure. Instead, the solution should be a permissioned blockchain that does not require a peer to peer network in a distributed environment. The problem with the permissionless blockchain that runs on a peer to peer and decentralized network is that it takes a long time to establish. Further, a lot of marketing, research and development is required to establish a fully functioning blockchain network with distributed and anonymous peer nodes. Such a feat is difficult to obtain for the means of an experimental study for this research problem. In lieu of the decentralized, trustless blockchain, a hybrid approach would use the security characteristics of the blockchain that provide immutability. This approach can ensure the integrity of the data in the blockchain and use the key elements of validation using hashing and encryption. These characteristics would provide more security to STP device validation than the earlier cryptographic solutions provided. It would also be more feasible than the options that required hardware changes, modifications to the STP protocol and a fully functioning decentralized blockchain network. The key to adoption and acceptance is to provide a solution that is adaptable to current systems to provide a stepping stone for change. The hybrid solution provides a testable environment to demonstrate the success of blockchain's security characteristics on an Ethernet LAN enabled with the STP topology.

Chapter 3

Research Framework

The STP protocol is a Data Link Layer algorithm for automated network device management in an Ethernet LAN. The trust-based nature allows the election of a topology management device without validating the named root bridge in the frame. Rogue devices enter Ethernet LANs through MITM attacks and launch false STP traffic to the switches electing themselves with the root role to gain control of the Ethernet LAN. This determination led to finding a solution using one of the most robust multilayered trustless solutions available: blockchain. This research problem analysis provided a deductive approach to choosing an appropriate research framework for the study. After resolving the initial decision to research and test a viable solution to solving the STP protocol algorithm's vulnerability by finding a means of validation, the second decision was to resolve how the research study would be conducted. There is a multitude of perspectives from which to deduce a specific choice in the field of research. A careful study of these available choices would reveal a specific research path that provides the best fit for the study topic. Creswell defines the logical deduction of narrowing a broad perspective down to procedural specifics as a sum of three successive steps: research approaches, research design, and research methods (Creswell & Creswell, 2018). An inquiry into the available paths can bring a specific topic more focused thought as a decision is made at each fork in the road to a successful study. The following three sections provide the analysis of these three successfive steps with the research topic. The next section begins with the research approach that is optimal for the study.

Research Approach

The first step and the broadest perspective to choose the ultimate research path to determine the most decisive direction to take from three high-level choices (Creswell & Creswell, 2018). The three choices from which to determine one approach are quantitative, qualitative, and mixed methods. The first two approaches, quantitative and qualitative, provide the broadest difference in perspective because the third approach, mixed methods, combines the two meth-

ods (Creswell & Creswell, 2018). The qualitative research approach allows the researcher to ask open-ended questions to determine the direction of progress. Its flexibility allows for a more fluid approach that helps in a situation where a narrowly determined solution would not fit (Creswell & Creswell, 2018). There are several research problems in information technology that provide a good fit for a qualitative research approach. One example would be the IEEE's feasibility of taking on a significant change in complex traditional network infrastructure (Creswell & Creswell, 2018). This research study could be conducted with interviews with open-ended answers to apply the most integrity in the research study's direction (Creswell & Creswell, 2018). Social issues, an appropriate research category for this type of research approach, would be the economic impact to the dissolution of several critical areas of vulnerabilities that have allowed them to prosper. For example, if a solution to block MITM attacks existed, it would impact network security companies as they would no longer have a problem to resolve.

The quantitative research approach allows a researcher to determine answers to close-ended questions (Creswell & Creswell, 2018). If the subject of these questions is not a person, it can be a variable in an experiment. Finding the answers to questions that involve a binary response, such as "it either works or it doesn't," would be more fitting in a quantitative research approach (Creswell & Creswell, 2018). Such experiments can be set up in a controlled lab environment where the variables and relationships between them can be studied for a multiple-choice or true/false answer. The carefully determined narrowing of responses works well with research questions answered by testing a specific hypothesis a researcher wants to explore (Creswell &Creswell, 2018). A research topic that may work towards a solution for a vulnerability that either works or does not would be best resolved using a quantitative approach (Creswell & Creswell, 2018). An approach using non-human variables, such as computers in a controlled setting, could be if encryption can deter a vulnerability. The two variables used in this study would be computers that use encryption technology and a set of computers that do not. The same set of factors to manufacture the vulnerability and observe the results would be the relationship between the variables.

If the researcher wanted to work with a narrow set of results but wanted additional insight with open-ended ideas, a mixed-methods approach would be appropriate (Creswell & Creswell, 2018). This research study would be time-consuming because it would require two studies combined into one to complete (Creswell & Creswell, 2018). The study results would add a human approach to an otherwise "black and white" answer. An example of a mixed-methods approach with a security vulnerability-lab experiment would be to determine the feasibility of encryption software in deterring vulnerabilities with the two-variable approach, and

slower network speeds inconvenience the computer operators' open-ended reactions. This type of study may be of interest to a security company that would be interested in exploring the impact of this new security technology on human emotions (Creswell & Creswell, 2018). Concerning the narrow approach to how the proposed solution will be tested in a lab environment, it can be logically concluded that the appropriate approach to this research study would be quantitative. The research question is narrowly ended in that a common vulnerability was tested in a controlled lab environment. A research solution was tested in the same lab environment to analyze its causal relation to the attack vector. The next step to narrow down the research approach is to determine a research design most applicable to the research study.

Research Design

The previous section's research approach analyzed why a quantitative research approach is appropriate for this study's security research question. The quantitative approach was chosen because the topic of study requires specific and conclusive results from the data collected. Moving forward with this research approach, the next set of choices pertains to a logical conclusion on what type of research design to take. A research design explores the way inquiry is made to collect data for the research question.

The quantitative research approach offers two designs that determine how variables are chosen and how to observe their relationships. The critical difference is in how these variables are chosen and how their relationships are evaluated. The philosophy behind this difference comes from post-positivist theory (Creswell & Creswell, 2018). Postpositivism is a progressive response made by the scientific community to the positivist theory that states purism and absolute truth is "true knowledge" (Diaz, 2014). Where positivism provided a purist a format for the realization of conclusions through deductive reasoning and logic, the movement towards finding a solid basis for these conclusions provided the element of causality. In other words, a post-positivist would ask, 'What are the elements that cause this conclusion to be true?'. Since positivism is based on assumptions, post-positivist thought's mere orientation lies in the scientific method (Johnson, 2009). Therefore, the collection of empirical data became an idea to strengthen a conclusion. The main difference between positivism and postpositivism is the causality that determines the outcome and its resulting reductionism (Creswell & Creswell, 2018). In causality, a determining factor is how one element may cause a change in another. In reductionism, a logical conclusion to a post-positivist problem is deduced by measurements and tangible evidence. Therefore, post-positivist thought determines the cause of change in an

element by making conclusions about collected data. Typically, this theory would result in measuring a specific causal effect on a reduced set of variables. The post-positivist theory behind the scientific method typically entails empirical data collection that agrees or disagrees with the scientific method's hypothesis (Creswell & Creswell, 2018).

The research in the effectiveness of a technical solution on non-human variables is not reduced to validating ideas that can be stated as true, false, or meaningless. Therefore, this study's approach would have no value without measurements and data to prove its validity. A positivist debate could provide a conclusion that determines the effectiveness of information security. The subject derivation of a conclusion would be based on logic and reasoning alone, and this provides no value for the analysis of a software tool. The post-positivist approach, which follows the scientific method, has the means to derive a logical conclusion with measurable data. The research problem allows for a technical approach to conclude the analysis of measurable data. Concerning the preference for postpositivism in this study, the quantitative approach's resulting design options are observation and data collection through a survey or experimental design. Survey design takes an approach to causality in that observation of specific variables, and their relationships require more time to reach a conclusive result (Blalock, 1991). Hence, this type of causality takes a correlative approach (Creswell & Creswell, 2018). With many research topics, live subjects provide a more complicated approach to a reduced set of results. The survey design would be a good approach.

Alternatively, the second option to observing causality is with experimental design. The research problem requires a conclusion that is based on a causal approach with non-human variables. This study's variables provide measurable data that can provide value in experimentation and collecting measurements and empirical data. In an experiment, variables and tests can be provided and manipulated with conclusive results (Creswell & Creswell, 2018). The critical difference is that variables and their relationships are defined in a controlled environment. Although causality in experimental design can be performed on live subjects in a controlled environment, such as medical observation of subjects taking medication and a placebo, it is appropriate when observing inanimate objects. The manipulation of variables and their relationships for tests comes with slight complications when studying inanimate objects (Creswell & Creswell, 2018). Further, setting up a controlled environment with inanimate resources provides an ideal setting for experimental design. For example, if an experiment were to be performed on the effects of specific resources on a computer were to be measured, there is no measure of ethics for manipulation of the variables, and the selection of controlled variables is straightforward. The research study is the examination of variables and the collection of data and measurements

through tests in a controlled lab environment. The test subjects are software, virtual computers, and networking switches, a specific subset of network infrastructure variables without random sampling. Experiments will measure direct causality making the experimental design most appropriate for the choice of variables, data observation, and collection (Creswell & Creswell, 2018).

The components of this research study for research design include the variables, procedure, and measures. The variables are chosen based on available resources that will represent the overall network architecture. The experimental design choice provides various types of variable assignments. There are the pre-experiment, true experiment, quasi-experiment, and single-subject design (Creswell & Creswell, 2018). The critical factor in how the true experiment and quasi-experimental designs differ is sampling a control group and experimental group for comparison. One group is observed and measured for control data where another is provided the experimental condition. The condition is observed on two groups, so it is essential to have both groups have equal representation (Creswell & Creswell, 2018). In the true experiment, a random sampling ensures that one group's observation will not vary from the other outside of the experimental condition. The quasi-experimental approach to observing both groups has a manipulated variable of interest that is required due to infeasible random sampling due to the research topic and lack of true variance (Creswell & Creswell, 2018). The pre-experimental design uses one group and observes the intervention results with the same group's experimental condition. A single-subject design is used to observe a single subject over time (Creswell & Creswell, 2018).

This research study uses a control group that is observed under the attack in the research study. The same control group will be used when applying the experimental condition to observe results and data. This type of experimental design is described as the pre-experimental research design.

Research Methods

The research approach is a quantitative experiment with a pre-experimental research design. The last step in fine-tuning the research plan is to describe the research methods used with the control group's variables, procedures, and measurements (Creswell & Creswell, 2018). This section introduces the pre-experimental design variables and analyzes how they were used in the research plan. Next, a detailed procedure is provided based on the pre-experimental design plan. A plan for data collection is added to prove the research study's validity and its solution.

Lastly, an illustration of responses will be provided as part of the research framework to provide a disclaimer to questions that could arise to the research topic's legitimacy, choice of variables, design, and possible conclusions from the resulting data.

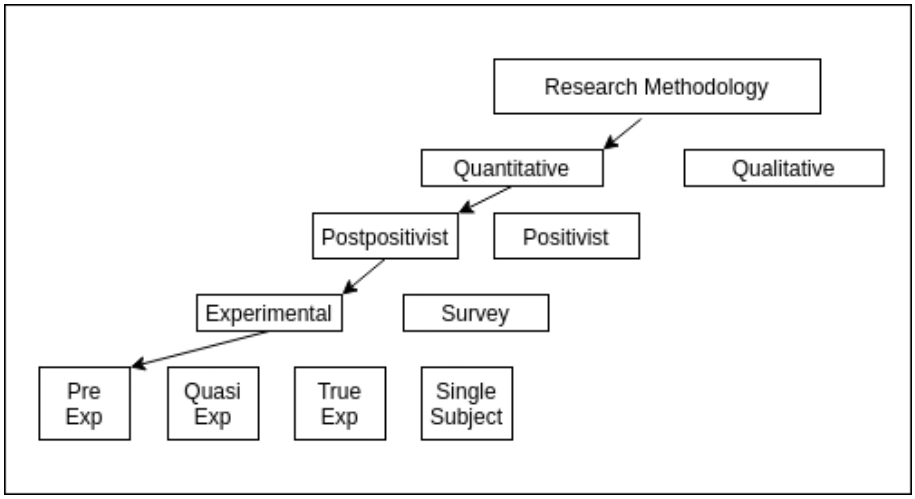


Figure 4. The Research Methodology Decision Tree

The pre-experimental research design has a control group of independent variables. This control group was measured and observed before and after the dependent variable was added to it. The independent variables of the control group represent how the vulnerability exists in the current network infrastructure. The control group was set up and observed for effectiveness in a lab experiment. The control group was made up of three vital elements for the STP Root-Takeover attack. This control group was configured to prove that the attack still exists using the current standards for STP in the Data Link Layer. The experiment required a virtualized setup using available resources to represent the real-world network infrastructure affected by the STP Root-Takeover attack. This virtualized lab environment housed the first independent variable, the Ethernet LAN. It comprises STP-enabled software switches connected with the IEEE 802.3 standardized Ethernet LAN topology in a hypervisor environment. This lab setup provides a software-based representation of the abstract data link layer running standard protocol services in the Ethernet LAN per the IEEE standard (IEEE , 2018). A manipulation check measure was utilized on an independent variable to ensure observable data and test the effectiveness of the ability to manipulate the independent variable (Creswell & Creswell, 2018). The manipulation check measure used in this experiment is the second independent variable applied to the control group, the attack vector. The second independent variable is the STP Root-Takeover attack vector used to manipulate the first independent variable, the Ethernet LAN. This manipulation was tested for validity in observation one. The validation test on the Ethernet LAN attack ensured that the vulnerability still existed and was evidence that the trust-based

STP protocol allows a rogue device into the Ethernet LAN. The outcome of the manipulation of the second independent variable provided the justification requirement of a dependent variable (Creswell & Creswell).

Variables used in Pre-Experimental Research Method	
Independent Variables	Dependent Variables
• Ethernet LAN • Attack Vector	Blockchain DApp

Figure 5. Pre-Experimental Control Group Variable Assignments

The dependent variable in the pre-experimental design is the research solution that provides a second set of observable data on the control group. The same control group was observed before and after applying the dependent variable because the solution was tested for its ability to detect the control group's condition after manipulation by the second independent variable. The pre-experimental quantitative guide provides the requirement for proven elements in the experiment. The choice of independent and dependent variables provides a methodological approach to experimenting with scientific validity (Creswell & Creswell, 2018). The Ethernet LAN from the Observation One was presented with the the dependent variable for Observation Two. The dependent variable is the software research solution created for the test environment to prevent the STP Root-takeover attack. The attack vector from the experimentation in step one is reintroduced, and the results observed tested the validity of the research question. Concerning the research study, the STP Root-Takeover attack was administered on the Ethernet LAN to be observed and observed for its response to the attack. After the research solution was applied to the Ethernet LAN, the same STP Root-Takeover attack vector was administered on the Ethernet LAN for observation. The research solution is a software-based application that uses the blockchain protocol-based encryption algorithms to validate devices in an Ethernet LAN.

Research Procedure

A research procedure is a step-by-step plan for the study based on its approach, design, and selection of variables (Creswell & Creswell, 2018). The plan creates replicable results in a controlled lab environment to answer the research question. The study began with creating a lab environment to house the experiment, and the pre-experimental methods to the research

design required one control group. The control group consists of entities affected by the research problem were tested to validate the relevance of the research question in Observation One. The same control group was then used to add the research solution used for testing and data collection in Observation Two. The control group was made up of independent and dependent variables. The independent variables are those that remain consistent, namely, the test environment and the control group. The dependent variable was introduced in Observation Two for data collection and observation. The research procedure is described in three sections: *Control Group setup and Observation One, Implementation of the STP DApp* and *Control Group STP DApp setup, and Observation Two*. Each section provides the implementation procedure.

Pre-Experimental Research Procedure	
Control Group	Ethernet LAN, Attack Vector
Observation One	Validate success of STP Root Takeover attack on the Control Group
X	Add the Blockchain DApp to the Control Group
Observation Two	Collect data of action taken by the Blockchain DApp during the STP Root Takeover attack on the Control Group

Figure 6. High-Level Pre-Experimental Research Procedure

The *Control Group* with the Ethernet LAN and the attack vector was set up for both observations. *Observation One* validates that the STP Root-Takeover attack is performed successfully on the control group. The implementation of STP DApp, represented by *X* in the diagram, follows the validation of the attack. After the implementation is complete, *Observation Two* is conducted on the *Control Group* with the attack in Observation One. In *Observation Two*, the STP DApp is launched before the attack is performed. The results are based on data gathered on the effectiveness of the STP DApp to detect the attack.

Control Group Setup and Observation One

This study's most decisive step was to test whether the STP Root-Takeover attack is still a security problem. This circumstance is due to the nature of information technology. Information technology is a field that is rapidly changing, which results in rapid responses to publicized security problems. The control group environment must reflect the IEEE standard of updated technology in widespread use to form a valid research study. Further, this control group, albeit housed in an experimental test environment, must provide a sample that can prove that STP Root-Takeover attacks still exist on modern networks. The question of the research problem's validity was still in question at the time of the research proposal. Any such experiment con-

ducted to prove the current status of the STP Root-Takeover attack as a security problem was beyond the scope of the archives of academic research. At the time of the literature review, the newest experiment proving the STP Root-Takeover attack's validity was six years old. The first step answered the first question in the experiment. When the attack vector is introduced, is the STP Root-Takeover attack successfully executed on the control group? In a valid test, the control group is exposed to the attack vector and undergoes the STP Root-Takeover attack. After this crucial first step has been conducted to prove the problem, a solution is introduced to the experiment to test the research question. The second step in the experiment is to test the control group's response to the attack vector using the research solution introduced in the study. The research solution is implemented on the control group after the research problem is validated in the first step.

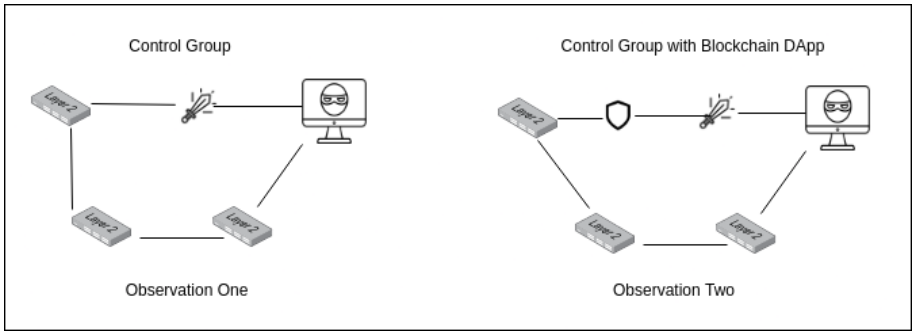


Figure 7. Diagram of the Control Group in Observation One and Observation Two

Can the research solution for this study prevent the STP Root-takeover from happening? The research solution is implemented on the previously affected control group, and the attack vector is reintroduced. Observational data is then conducted for analysis to prove the scientific validity of the solution. The resulting control group will be made up of the independent variables, the Ethernet LAN and the attack vector. These independent variables are the same in both steps of the experiment. Hence, the procedural setup of these variables is crucial to maintain proven continuity.

The Ethernet LAN

The Ethernet LAN required to conduct both steps of the experiment must be made up of switches with interconnecting media that use the STP protocol. Due to varying costs of lab environments, the pragmatic solution resulted in an entirely virtual Ethernet LAN. An Ethernet LAN made up of virtual switches and virtual media interconnecting them can be housed in a virtual environment called a hypervisor. A hypervisor is a software environment that can run large-scale enterprise networks and one host computer with no limitations to typical day-to-

day functionality that is vulnerable to the research problem (Blenk et. al., 2016). The ease of use, low cost, and accessibility in cloud networks provided an opportunity to create an optimal Ethernet LAN with a stable implementation and the required resources to test the problem and its solution (Browne et.al., 2018). The Ethernet LAN test with the solution provides observable data relevant to Ethernet LANs of all formats and sizes.

Yersinia: The Attack Vector

A separate virtual machine is also introduced to the Ethernet LAN to run the Ethernet LAN's STP Root-Takeover attack. This virtual machine is also a Debian Linux distribution called Kali Linux. The 2004.3 LTS version was the most current at testing time. The Kali distribution was chosen to perform this task because it is a widely used operating system used for penetration testing and performing attacks such as the STP Root-Takeover attack. The latest available version of the Yersinia application is installed on the Kali virtual machine. Lai et al. (2014) used Yersinia is an attack tool used on an earlier Kali distribution in the test environment for the STP Root-Takeover attack.

Observation One

Yersinia was initialized as a root user to provide its functionality through the graphical mode to perform the task. Due to its predictable functionality in claiming the root role, the graphical interface was used to send *BPDU_conf* frames to the Ethernet LAN. At initialization, Yersinia detects an interface for listening to all network traffic, configures it to accept all traffic. The tool provides the option of making a MITM attack with MAC spoofing to gain access to the traffic for information gathering. After selecting the option *"Claiming the Root Role"* is launched in the application, Yersinia sends a fake BPDU frame into the Ethernet LAN. Proof of a validated attack is verified by checking the current Root for the Ethernet LAN bridges.

Implementation of the STP DApp

The STP DApp is an implementation that is built into two parts. One part is the kernel-based interceptor written in C. The other part is a Hyperledger Fabric blockchain framework in userland. It is a web application written in the NodeJS runtime environment with an interface to communicate with the interceptor for Root validation, front-end application, and nodes written in JavaScript and Chaincode written in Go. The nodes are virtual machines that make up the blockchain framework. The virtual machine used to house the STP DApp tool has the latest Ubuntu 20.04 LTS Desktop installation running eight cores of CPU, 29GB of RAM, and 256GB of hard disk space in the VCloud datacenter. This virtual machine resource configuration is

relevant to the data collection used for the study.

The Network Interceptor

The interceptor's setup directly connects to the LEB's network traffic from the Data Link Layer to the application layer kernel */net* module. The kernel space provides the main operating system functionality. It is written in the C language, and a modification to the kernel space is most directly handled by adding and initializing a definition of the modification. The C language uses kernel libraries that provide functions that control network traffic's ingress, egress, and manipulation. It is the part of the operating system where Ethernet frames can be parsed and accepted or rejected according to set rules.

The Hyperledger Fabric Blockchain Framework

The Hyperledger Fabric framework is an open-source project by IBM for research and development in enterprise systems. It provides a platform for ease of use and testing. The study's test framework is encased in a NodeJS runtime environment and deployed with two Peer nodes that handle the search and addition of Root IDs to the blockchain. A JavaScript interface pulls the Root IDs from the interceptor's memory file called *root_out* in the proc folder. The Root IDs are pulled from the *root_out* folder to run the search and return validation results.

Control Group STP DApp setup, and Observation Two

The control group used for Observation One is used for Observation Two. In Observation One, only the independent variables were used to verify the attack. Observation Two includes the two independent variables in observation one and the dependent variable. The dependent variable is the STP DApp research solution. The control group already exists, so there is no need to set up the three bridges and the rogue attack vector in an Ethernet LAN. First is the addition of the implemented STP DApp to the control group. The STP validation tool is implemented in an LEB similar to the LEBs in the control group. The LEB is added to the Ethernet LAN. In Observation Two, the Ethernet LAN includes the three bridges, *leb1,leb2*, and *leb3*, the attack vector, rogue, and the STP DApp VM called *bcvm*. The STP DApp is first activated by initiating collection and querying the Root IDs in the network traffic. Since any bridge could be elected as the Designated Bridge, the blockchain ledger holds a list of the Ethernet LAN bridges as valid devices. The attack will then be initiated on Yersinia as the STP DApp collects data in *proc* files from the interceptor. The virtual machine will be running the Wireshark application and the STP DApp to collect data for the study.

Data Collection

CPU usage and RAM utilization were measured while using the STD DApp. The implementation of the STP DApp is part kernel-based mod called *stpverify* and part Hyperledger Fabric blockchain framework. It measures the stability of the resource load during the STP DApp validation process. The measurements were taken in one-second intervals using the *vmstat* command tool.

- Six trials of a hundred measurements were taken as a baseline on a separate Ethernet LAN bridge
- Six trials of a hundred measurements were taken of STP DApp validation during an attack.
- Six trials of time efficiency were collected based on Wireshark packet analysis

The research plan included a procedure that is appropriate for the quantitative approach with a pre-experimental design. This section outlines the research methods that were employed for the research plan. The following section is a description of the research solution that was implemented for this study.

Research Solution

The previous sections in this chapter describe the study's research methodology and procedure. It includes the implementation of the STP DApp research solution. This section will detail the STP DApp and how it functioned in Observation Two. The solution to the STP Root-Takeover attack integrates the SHA256 Hash algorithm used for authentication, transmission, and storage of data in the blockchain ledger. It also uses the Merkle Tree algorithm to validate the integrity of the data in storage in the blockchain ledger. Implementing a Decentralized Application (DApp) that performs Root validation is proof of concept and stepping stone towards a trustless peer-to-peer solution in a large-scale distributed peer-to-peer network as blockchain ideology would have it.

Testing Environment

The STP Root-Takeover attack research problem was validated in Observation One of the experiment. Observation Two began after the implementation of the STP DApp solution. The tool is integrated with the control group as an additional LEB in the Ethernet LAN. Observation is continued in a controlled and virtualized lab environment on a hypervisor. The LAN is

made up of network devices installed on virtual machines. The LAN connections are made up of virtualized media using the TCP/IP protocol for NAT assignment on device interfaces from a Pfsense router. Observation One was intended to prove the validity of the virtualized test environment using software-based switches in a virtualized Ethernet LAN. The version of all devices used for these experiments is significant due to the rapid pace of updates and security patches made to them. Knowledge of the operating system allows the researcher to understand if it had a vulnerability during testing. The Ethernet LAN is configured in the VMware VCloud 10.2 hypervisor environment. The control group was test Ethernet LAN housed in a VMware hypervisor environment consists of three bridges, a rogue machine to run the attack, and a router used as a firewall to protect the Ethernet LAN from common attacks. The network switches that make up the LAN bridges are four virtual machines installed with the Ubuntu 20.04 LTS desktop version that was current during the study. All four of these virtual machines were configured as Linux Ethernet Bridges (LEBs) using integrated bridging application programming interfaces (APIs) in the */net/bridge* module. A second network interface was added to the LEBs for bridging. Three of the bridges are used as Ethernet LAN network devices, and the fourth has the STP DApp solution installed. In Linux nomenclature, the configuration of a bridge makes the device akin to a switch. Therefore, the switches in this Ethernet LAN will be referred to as bridges moving forward. It is important to note that the words ”bridge” and ”switch” are synonymous in this study. These bridges are configured to use the STP protocol for STP-enabled Root validation in the Ethernet LAN control group.

The STP Root-Takeover attack was launched from a virtual machine installed with the Debian Linux-based Kali Operating System 2020.4 LTS Desktop, the latest stable Kali OS version. This attack was aimed at an Ethernet LAN made up of three switches and a router using version 2.4.5_p1 of pfSense, providing a firewalled LAN interface. The rogue virtual machine uses the Yersinia attack tool on the Kali operating system. The key to proceed with a research solution is based on the validity of this attack on a similar system and network environment seven years later. In the study by Lai et al., Yersinia was installed and run on a Kali virtual machine to launch an attack with falsified STP topology information to allow it a Root role and subsequent control of the Ethernet LAN (Lai et al., 2014). During the test experiment for this study, forged BPDU frames claiming Root role by the rogue system were launched and validated by the Ethernet LAN bridges.

Observation One showed a successful STP Root-Takeover attack launched by the Yersinia attack vector on the control group. The LAN LEBs were running the latest version of the Ubuntu Desktop OS. It’s success also validates that current attacks available on the latest Kali Operat-

ing System prove a vulnerability in the STP protocol of the Data Link layer in the LEB's STP implementation. A solution using a hybrid blockchain framework was implemented to provide a Proof of Concept (POC) . Although the STP DApp was permissioned due to test environment constraints, the blockchain system's potential allows the researcher to provide a theoretical example of a trustless STP protocol through validation. Although the Merkle tree hashing algorithm provides an element of trustlessness in that validation is made through automation and immutability, a small-scale corporate solution that allows manual entries for blockchain transactions requires a trusted component for testing. A large-scale move towards an inherently secure network infrastructure would provide the testing ground for integrating a peer-to-peer distributed network. This network is required to implement the full capacity of trustless validation using consensus algorithms and the Merkle tree. The STP DApp solution is a hybrid permissioned blockchain system with CAs, SHA256 based PKI authentication, encrypted transmission, and Merkle tree validation integrated into the framework to eliminate the trust-based nature of STP that lacks BPDU validation.

Construct and Function

This STP DApp research solution's functionality is to validate Root IDs in the Ethernet LAN. This functionality adds a validation element to the trust-based STP algorithm. The blockchain framework is used to perform validation of BPDU Root roles intercepted by the *stpverify* kernel *net* module. The solution is trustless because it adds validation. However, a genuinely trustless solution based on blockchain protocol would not require a trusted user to add valid Root IDs to the blockchain manually. This quasi-trustless solution solves the research problem in small-scale networks that would require a trusted network administrator to validate and manually add the MAC address of all new switches added to the Ethernet LAN. The use of a trusted administrator makes the blockchain framework centralized and permissioned by definition (Nakamoto, 2009). The STP DApp works by dropping frames from bridges invalidated after a Chaincode query. When Yersinia's forged frames are dropped, the attack in Observation Two can no longer take the Root role. This solution provides a reasonable modification to the current trust-based STP protocol used in integrating new switches to an Ethernet LAN by requiring validation. The practical solution is a stepping stone towards solutions in large-scale enterprise environments with trusted network administrators. The modification to trustless validation will eliminate the vulnerability that causes the STP Root-Takeover attack by validating BPDU frames before the official selection of a new Root. The hybrid solution introduces an additional LEB that acts as a Layer 2 switching device with an encrypted and immutable ap-

plication layer based MAC table created by the blockchain protocol. A search of this emulated MAC table implemented in the blockchain ledger, allows BPDU frame validation by rejecting traffic containing a Root ID that does not match the list of known devices in the Ethernet LAN. Therefore, the blockchain framework search determines if the new Root selection is a validated switch in the Ethernet LAN. DApp, or Decentralized Application, is the standard nomenclature for an application connected to a blockchain. The decentralization aspect alludes to a distributed and anonymous peer-to-peer network emulated by two localhost Peer servers in secure containers making up a part of the blockchain framework. A standard switch typically has a MAC table that consists of validated devices in the Ethernet LAN but it is not traditionally used for STP Root validation. For validation, the research solution emulates two encrypted ”MAC tables” with a protected current state hashed database for quicker searches and a blockchain for the immutable storage of validated bridge IDs using an emulated consensus protocol. These two ”MAC tables”, accessible only through encrypted and authenticated transmission, provide quicker searches with the implementation of encrypted current state databases on Apache CouchDB 3.1 servers and immutable storage for validated bridge IDs using an emulated consensus protocol (Xu et al., 2021).

The experiment to validate the STP Root-Takeover attacks on modern Ethernet LANs answered questions in the research proposal about the validity of Linux Ethernet Bridges (LEBs) for a test experiment emulating the average Ethernet LAN. According to the experiments in this study, the Linux-based software bridge replicated the functionality of a traditional hardware switch by popular vendors in its vulnerability to the STP Root-Takeover attack when using the STP topology. More specifically, the research study shows that the STP protocol on Linux-based switches is implemented by IEEE 802.3 standards, as evidenced by the Ethernet frames' format. The STP-based Ethernet frames used as Layer 2 traffic are identical in byte by byte format to those transmitted among popular hardware switches by vendors such as Cisco and Juniper. The difference between these hardware switches and the LEB is that the latter uses the Application Layer and has a kernel module called *net* that holds various network-related modules with application interfaces to network components. Some of these modules have applications that allow a direct connection to Layer 2 traffic from the bridge interface (Kankipati, 2020). This construct allows the LEB to perform as a Layer 2 device that passes BPDU frames. In a description of LEBs, Nuutti Varis maintains that the Linux Ethernet Bridge operates in the Application Layer (Varis, 2012). The exploration of how the Linux-based software switch works in the network stack and its integration with the Data Link Layer to create an Ethernet LAN led to the kernel module called */net/bridge* that contains programming for bridge functionality and access

(Kankipati, 2021).

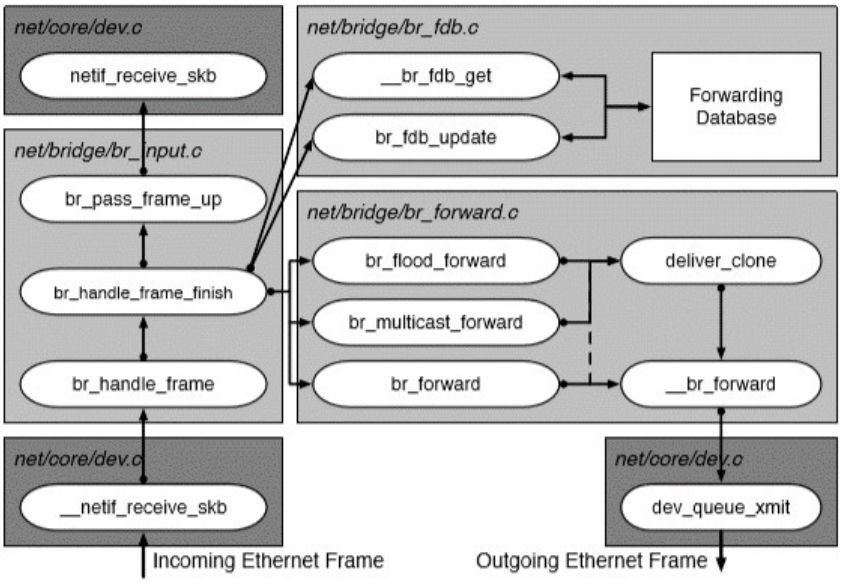


Figure 8. The Anatomy of the Linux Ethernet Bridge (Vaaris, 2012)

The *stpverify* kernel modification collects network traffic through the LEB's network interface. It allows the BPDU frames' interception by manipulating their route from the Data Link Layer to the Application Layer through the network interface. Access to BPDU traffic in the kernel allows the use of the STP DApp for blocking attacks. The benefit of implementing the DApp is the additional functionality and control over the BDPU frame validation on the LEB by using a blockchain protocol. Due to its ability to forward or drop Ethernet frames, the LEB provides adaptability to the use of a DApp, an application layer-based web application for validation. A typical DApp has its front-end functionality at the Application layer with a cloud-based blockchain for anonymous crowd-sourced validation. The research solution uses a hybrid blockchain framework with a front-end run in a NodeJS runtime environment connected to containerized Peer servers that form an emulated cloud system with a localhost network of authenticated artifacts. A DApp model was implemented to intercept STP frames (BPDU messages) from data link layer traffic and perform a validation check by comparing the extracted Root ID with a database of valid Bridge IDs that have been hashed and integrated into a blockchain. It will perform the validation and respond to whether the Root selection is a valid switch in the blockchain before a new Root selection is approved. The DApp was built and executed on an LEB similar to the network devices. It has the latest NodeJS and Docker software to support implementing the hybrid blockchain framework called Hyperledger Fabric. Based on the Hyperledger framework created by the Linux foundation, Hyperledger Fabric by

IBM provides a turn-key approach to building a functional blockchain application for research and testing (Parisi, 2020). Its robust system allows ease of use to program business logic outside of a blockchain to increase speed and efficiency. It allows the easy addition of a database of the blockchain's current state. The database allows resource-efficient searches because the traversal of the blockchain is energy consumptive and slow (Xu et al., 2021) Lastly, it provides a user interface (UI) in the front end that allows user accessibility for decoded searches and manually adds transactions to the blockchain (Adhav, 2020). The lack of integration with the distributed peer-to-peer cloud for the blockchain network eliminates the consensus protocol that adds a blockchain network's actual trustless functionality. However, the use of distributed network validation through the consensus protocol does not add value to this grassroots-level study on a blockchain network's feasibility for localized validation of network devices.

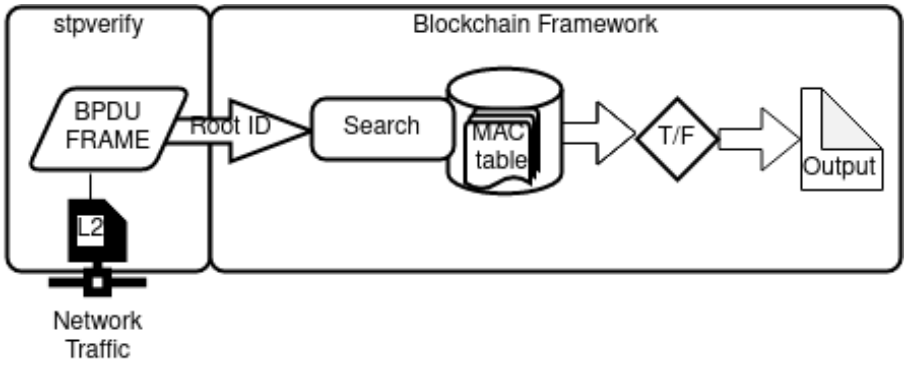


Figure 9. The Anatomy of the STP DApp

Why a Hybrid Framework instead of the Average Blockchain?

The experiment is built on a pre-experimental approach that requires a working solution with empirical data to measure and base conclusions. This experiment will require a working product to deliver a usable solution in a current testing environment. The average blockchain, which is trustless, immutable, and permissionless, provides security ideology. A true consensus algorithm uses Proof of Work (PoW) or Proof of Stake (PoS) to build trustless validation (Lepore et al., 2020). Such an environment requires a distributed cloud environment with participants that lend resources. This environment can take years to build, especially in validation for network devices. Using consensus in a peer-to-peer environment is not feasible for a test network. IBM's Hyperledger Fabric has provided an environment for testing that builds on many of blockchain's encryption algorithms. A partial or hybrid blockchain environment allows for a test environment where testing is feasible for research. This study provides value when it is possible to reproduce and build on for future study. Although not ideal, the permissioned blockchain or

hybrid approach allows a test-bed for future ideas to come to fruition. The Hyperledger Fabric framework has many differences from Bitcoin Core's technology and its permissioned option is one of them (Parisi, 2020).

The key elements are based on the trustless nature of a peer-to-peer environment. It took years to build; however, Bitcoin Core's ideal allows for a trustless environment where the distributed nodes are run in a self-automated fashion (Nakomoto, 2009). The consensus algorithm to reward those that provide resources uses PoW algorithms rather than a salary from a trusted authority. Lastly, the permissionless environment in which a trusted authority is not required; an ideology on which Bitcoin Core is based (Nakamoto, 2009). The underlying feature of the Hyperledger Fabric is that it is a permissioned environment to emulate the corporate atmosphere in which it is being integrated. The reality of integrating the blockchain into the corporate atmosphere allows for a smoother transition to new technology. The permissioned approach to Hyperledger Fabric allows for research and development. Funding for research and development by the commercial industries builds practical use cases for blockchain technology (R. Shores, personal communication, May 15, 2019).

STP Validation Built on Hyperledger Fabric

For the sake of this experiment and its environment, a hybrid blockchain solution has been created to test its ability to use Smart Contract validation. Smart Contract validation is trustless when it is run on a large and distributed network system like the Ethereum blockchain. Such a network infrastructure is impractical in a test environment. Therefore, the Ethernet LAN has demonstrated the solution with a permissioned blockchain framework by IBM and the Linux Foundation. Due to the trusted and permissioned nature of the test network for this study, the solution's framework in this test environment is aptly named a hybrid blockchain solution. Such a hybrid blockchain solution has been created for the testing of corporate blockchain solutions. The blockchain solution for this study is Hyperledger Fabric, a popular enterprise solution that has been created for enterprise systems that are undergoing research and development (Adhav, 2020). Suppose the nodes in this blockchain are distributed in a cloud, and the system has the time and resources to establish itself. In that case, the code can be used as permissionless, trustless, and transparent as the blockchain was created to do on economic systems. In other words, if this solution has an environment where it can build a distributed network in the cloud, it will no longer be a hybrid but fully trustless, as is the concept introduced by this study.

This experiment has been created with a permissioned blockchain framework run on the localhost of an Ubuntu 20.04 LTS desktop virtual machine. The Hyperledger Fabric blockchain

is a web application written in JavaScript. This front-end of the web application houses the application that manages access control and blockchain requests through Smart Contracts. Smart Contracts are software programs that run inside virtual machines in the blockchain and are made to perform a task (Buterin, 2014). The Smart Contracts in the hybrid framework for the study are made to perform Root ID validation. These Root IDs are pulled by an API written in the NodeJS runtime environment for JavaScript after being collected from the *root_out_proc* file created by *stpverify*. The Hyperledger Fabric framework uses this runtime environment to create an interactive front-end and back-end blockchain system with Smart Contracts. The Smart Contracts are called Chaincode in Hyperledger Fabric nomenclature.

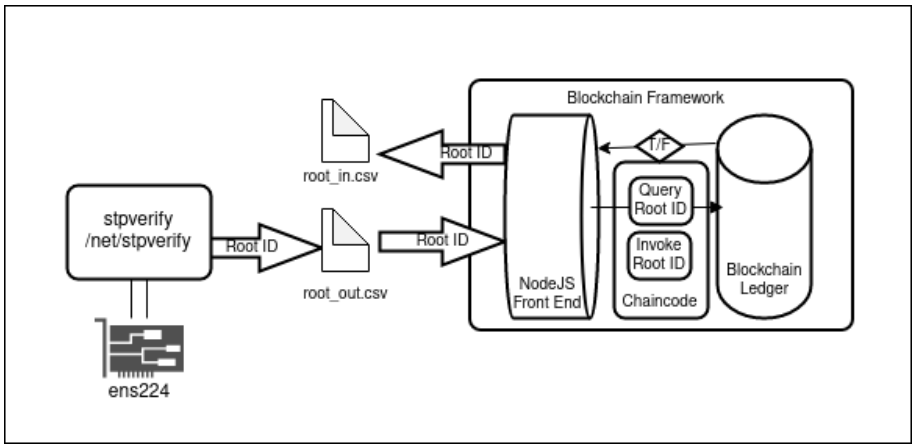


Figure 10. Process Diagram of the STP DApp

The Chaincode uses SHA256 PKI token validation to run a search for the Root Id in the blockchain. If a hashed copy of the Root ID is in the blockchain, a front-end script written in JavaScript returns Root IDs that are not in the blockchain to the *root_in* file. Frames that come from valid sources automatically forward. Per the kernel's bridge functionality, if there is no record of the Root ID in the blockchain, the front-end interface will output this value to *root_in*, and the *stpverify* modification will drop the frame to block it from the Root role.

Security Framework and Node Architecture

The study's blockchain network is driven by the Hyperledger Fabric framework of a permissioned blockchain to run on a localhost web application server in a VM. It allows for permissions setting for multiple level organizations. The servers that make up the web application are called nodes and are housed as images in docker containers. The docker containers provide secure modularity with networks that connect them. The docker containers have updated images of the required artifacts in the Hyperledger Fabric ecosystem. The ecosystem is established with an Orderer, Peers, Certificate Authority (CA) servers, and CouchDB containers to record

hashed transactions to the blockchain (Adhav, 2020). The Orderer node is a permissioned version of the consensus algorithm in the permissionless large-scale blockchains. The consensus algorithm validates transactions to add them to the blockchain, and the Orderer is a permissioned version of the consensus entity. It is made to fit with smaller business applications that require the use of automated administration. The Orderer node's role is to handle the business logic by authenticating network connections between other nodes managing the ecosystem (Adhav, 2020). When the blockchain is first deployed, the Orderer authenticates network connections using validation from the CA nodes, issuing certificates for all the nodes. The Peer nodes are used to process transactions. Each Peer can have different levels of access to the blockchain. For example, one Peer node may only have read-only access while the other has read-write access. The read-only Peer queries a blockchain, and the Peer with write access queries and adds transactions to the blockchain (Adhav, 2020). Blockchain-based Peer nodes that represent each organization handle these rights. The organizations are connected through a network channel. The channel connects nodes allowing authenticated communication between nodes. If one node does not have access to information communicated in another node, the channel handles this access control. The business logic or the main program is called the Chaincode. The Chaincode is defined on the channel and installed on the Peer nodes.

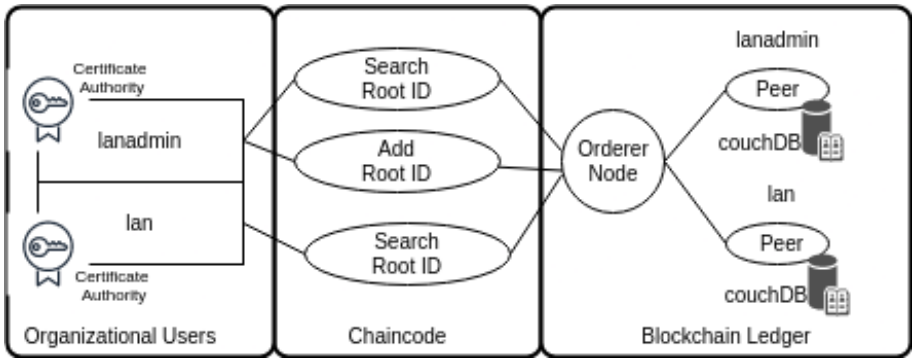


Figure 11. The Hyperledger Fabric Architecture for the STP DApp

Internal Threats to Validity

The previous section provides a plan for a procedure on the research plan. It has a detailed description of a plan to collect measurable data for the results of this study. The quantitative pre-experimental design used to observe the results of a viable solution to the research question has internal threats to its validity. Internal threats to validity pertain to incorrect conclusions drawn from observable data that can invalidate control group and variable choices

(Creswell & Creswell, 2018). The internal threats addressed in this section are the lab environment choices, Ethernet LAN, media connections, and network devices in representing the STP protocol. Further, the attack vector and observable data that are collected in this study are validated. Intentional measures have been taken to define the research environment to measure data in stable conditions. The selection of virtualized devices rather than hardware ones reduces any complications from faulty hardware or environmental factors that could affect the observed data. An environment that can provide stable conditions that are replicated with precision is possible with virtualized media. The Ethernet LAN in the control group is a virtual representation of the network infrastructure. Hence, the experiment's lab environment is not a physical representation of typical hardware and software used in home and enterprise networks. A typical home network using wired Ethernet may have a router, switch, and computers. A typical enterprise network would have either a hardware-based Ethernet LAN or a partially virtualized one with more devices. These real-world environments introduce another set of variables that are difficult to control for scientific research limited to a specific protocol. This study aims to research the validation of networked devices in a LAN based on STP topology. This type of study is accomplished in a controlled environment unbound by disruptions made by external features that are not required for the study. Further, the use of software-based switches and media connections that are exact duplicates of each other creates results based on the protocol's effects rather than the possibility of characteristic differences between the devices that could affect collected data. Lastly, this environment provides stability to run multiple tests that can be compared and analyzed with controlled precision. The STP protocol's importance as the standard is based on the popularity of the Ethernet LAN topology over the Token Ring topology introduced in Chapter 1 and its standardization by IEEE (IEEE, 2018). The Ethernet LAN topology used to represent Data Link layer traffic is supported by IEEE 802.3 and represents the Data Link Layer's most utilized topology in the overall network infrastructure affected by vulnerability IEEE standards (IEEE , 2016). The Ethernet LAN's services are contained within a hypervisor environment. The hypervisor environment provides a software-based setting that provides the required technology to run services to maintain virtualized computers, network devices, network topology, and network connections using protocols used by the network infrastructure. A hypervisor's virtual test environment was used for a comprehensive network protocol experimentation and data analysis in *BGP Route Attestation: Design and Observation using IPV6 Headers* (Ham, 2017). The hypervisor environment used in the experiment is provided by the vendor VMware. VMware has a hypervisor environment that is widely used for enterprise-level LAN environments. The experiment will utilize the latest supported version of this hypervisor environment

to test the STP Root-Takeover attack vector's validity (Browne et al., 2018). The test created in Vmware provides the researcher in this study the freedom to choose the type of resources required for optimal functionality rather than being bound by the limitations of the hardware (Kurniawan et. al., 2018). For this reason, a virtual environment provides confidence otherwise challenging to obtain from a hardware-based environment affected by hardware differences or defective media. (Blenk et. al., 2016; Kurniawan et. al., 2018). Although the STP protocol was initially designed to expand hardware switches, the STP protocol is made up of software algorithms that provide the same service to software-based switches as to hardware-based switches. The test of a security problem in the virtual data link layer does not require a physical network that consists of hardware switches that transmit and receive digital information in the Ethernet format. It is important to note that all switches, aside from minor software differences, use the same STP topology algorithm. Although some switches may have add-on features to enhance their security, the basic IEEE standard for the STP algorithm is standardized and used for this study (IEEE, 2016). A hypervisor set up in a physical datacenter can house virtualized switches that perform the same function as physical switches and media connections for a specific study of the STP protocol because IEEE standardizes the STP protocol in its requirement to maintain an adequate global network infrastructure (IEEE , 2016). Modern-day networks consist of hardware and software switches made by various vendors. Although in the spirit of competition, one vendor may claim its superiority over another, no proven data supports that one type of switch over another in terms of functionality as large-scale enterprise solutions are comprised of any variation of these switch formats. An example would be that of a vehicle. There are different types of transport such as a bicycle, motorcycle or trucks. The form of transport chosen pertains to its user's requirements rather than its ultimate task of transporting the user from point A to point B. In that respect, the software-based LEB switch must uphold the IEEE standard for the STP algorithm (IEEE, 2016). A way to check this functionality is to observe topology changes and the Root role election based on the Ethernet frames' inspection.

The network traffic analysis to the device's network interface for inspection was Wireshark, a popular tool for network traffic inspection. This inspection of the Data Link Layer traffic's format and behavior provided confidence that the STP algorithm implemented in the LEB *net/bridge* module corresponds to the IEEE standard. A test using a hardware Ethernet LAN made up of Cisco switches, and a comparison with packet inspection of Wireshark may strengthen its validity. However, this exceeds the limitations of this study. According to the Review of Literature and an analysis of its architecture, a critical difference between the LEB and a hardware-based switch would be that the former provides functionality for the Data Link

Layer and the Application Layer (Varis, 2012). Although there is a lack of literature and study of the *core* kernel module that drives the Layer 2 functionality in the LEB operating system, the network traffic's behavior upon inspection showed no difference in Ethernet's formatting or behavior after packet analysis. Currently, hardware switches work exclusively in the Data Link Layer. Therefore, the inspection of the network traffic comes from the media connection to an intermediary. Due to its ability to function at both layers, the LEB can provide access to Layer 2 network traffic through the Application layer and user-space inspection tools in the same operating system. The LEB's ability to create an environment to perform both functions in one virtual machine provided an opportunity to collect unobstructed data from external disruptions by media connections. The research study explored limitations that the Application layer had to the network traffic in the Data Link layer for the research solution in this study. The LEBs that make up the Ethernet LAN use virtual media connections rather than the physical copper connections with physical switches as set de-facto by IEEE for 802.3 (IEEE, 2016). It is important to note that the network infrastructure represented by the OSI model includes the Physical layer that creates the conversion of data from analog to digital using its protocol services and that the first complete abstract layer, Layer 2, is represented virtually in the experiment. Hence, a virtual environment can represent the protocols and services defined by the IEEE 802.3 standard for a reduced-sized Ethernet LAN. Although the nomenclature for Ethernet 802.3 is wired, the assumption that the wiring must be hardware is unnecessary for testing the protocol (IEEE, 2016). The physical wiring used in the Ethernet LAN uses Physical Layer services, and the LAN connection using the STP protocol is provided exclusively by the Data Link layer, so it does not affect the validity of the connection because the analog to digital conversion service of the Physical layer is not required in a virtual connection. This pre-experimental design is set up to test the security of the STP protocol's algorithmic design in the Ethernet LAN. Therefore, it has no dependency on hardware-based components for observable data. Virtual media connect the virtual switches to form an Ethernet LAN. This LAN is contained within a hypervisor environment. The attack vector utilized for observation used on the control group is a virtual machine connected to an Ethernet LAN made up of STP-enabled LEBs using IEEE 802.3 protocol services. The virtual machine will send the software-based attacks STP Root-Takeover attack using the latest LTS version of the Kali operating system based on a Debian Linux framework for operating systems. This operating system is commonly used in penetration testing to represent common attacks. This STP Root-Takeover attack is performed with installed software called Yersinia, a popular attack tool used for controlled MITM and STP Root-Takeover attacks. It is a tool that provides measurable data and stable conditions to perform attacks with

precision. The attack tool is open-source, which provides the opportunity for code analysis. This code analysis featured in Appendix B, is crucial to the transparency of the controlled study attacks in a lab environment. The last observation of STP Root-Takeover attacks provided by the literature review was designed with the same independent variables in this study. Lai et al. (2014) conducted their observations of this attack on an LEB using Yersinia's installation on Kali OS in 2014 in the article, *Trust-based security for the Spanning Tree Protocol* (Lai et al., 2014). This research provides an updated study on the same variables for comparison and insight.

The research solution is the STP DApp tool used to block changes in the Root role based on a list of validated devices in the blockchain ledger. The research problem indicates that the STP protocol is trust-based and requires device validation to provide a trustless solution to the STP Root-Takeover attack. The encryption algorithms used in blockchain protocol are set up to withstand common cryptanalysis attacks for standard enterprise resource availability. It provides an infrastructure of trustless algorithms to add a unique component of security to trust-based validation. One of the trustless algorithms, consensus, is based on the distributed, anonymous peer-to-peer network functionality based on a fully built and operational blockchain network. The purpose of this study was not to create a fully trustless infrastructure because it is meant to test the applicability of validation through blockchain protocol. The issue of attacks on a trust-based network is a part of security research and solutions contain ways to validate the behavior of nodes in the framework (Ma et al., 2020). Insider attacks on a trust-based hybrid blockchain framework can occur, however, algorithmic security measures can be implemented to prevent them (Cho & Cho, 2020). Hence, the DApp is a hybridized approach to the blockchain infrastructure that provides practical functionality for the research solution.

It is a hybrid framework created by IBM called Hyperledger Fabric for research and development testing of use cases in enterprise security solutions. It can be easily in a localhost network on a virtual machine. It works in a research environment without the external variables created by a distributed network. Further, it provides an additional database with the current state of blockchain transactions. The solution is designed to validate a new root network device in the Ethernet LAN by searching the blockchain for valid network devices. Typically, the blockchain traversal for a search is resource consumptive and creates latency, especially in a distributed network. This test framework allows for the addition of a database holding the blockchain's exact contents for searches eliminating the need for blockchain traversal (Xu, et.al., 2021) . The database speeds the process up and does not undermine security because it also utilizes hashing and authentication. An additional trustless feature provided by blockchain is the modularity of

the nodes that make up the framework. It is made up of Docker containers in a bridged localhost network that only functions with authentication at all endpoints (Adhav, 2020). This hybrid solution uses the NodeJS web application as an intermediary for trusted users to search and add to the blockchain through SHA256 token validation. It is not an additional feature required for the research solution. However, it provides a visual format meant to represent such a system's usability in a real-world environment. The use of a hybrid system like Hyperledger does not retain the full value of security that the Bitcoin Core does, however, the integration of a hybrid system is more compatible with today's infrastructure (Ma et al., 2020). A few security issues with this framework is it's use of third party vendor and open source solutions. The integration of these features creates more backdoors, security holes and potential zero day attacks. The upkeep of this framework would require consistent updates to maintain its security.

The encryption algorithm used for endpoint authentication, transmission, and storage in the blockchain ledger is SHA256. This solution is unique because the underlying infrastructure behind the validation is based on encryption and the Merkle tree algorithm that makes the blockchain ledger immutable. This feature provides a trustless record that an attacker cannot alter. It is trustless because it eliminates data alteration of the record of valid devices. At the time of this study, the SHA256 algorithm provided the highest encryption level that is secure and efficient (Nakomoto, 2009; Rachmawati et al., 2018). The effectiveness of SHA256 as an algorithm is based on the resource advantage of the attacker. If the attacker uses a quantum computer that can break these algorithms, it will invalidate the solution. However, the use of such resource-heavy attacks to perform an STP Root-Takeover attack would have an economic impact on the attacker, so its likelihood is slim.

External Threats to Validity

The external threats to validity pertain to assumptions made from the experiment's data that can affect past or future situations (Creswell & Creswell, 2018). The research methodology provided a route to the natural progression of resulting data collection. The research problem indicated a vulnerability in the IEEE standardized STP protocol in device validation that leads to the STP Root-Takeover attack. The study aimed to provide a practical solution and the collection of data to analyze the study results. The defense will cover a few points to validate the measured data in this study. Threats to the observed data's validity may come from the environment where the tool collected the data. The validity of the data that was observed can be questioned for its relevance to the research problem. The operating procedure during data collection may affect

the observed results, and it may have flaws. However, the controlled hypervisor environment and variables used can undermine the possibility of such questions during this study.

The virtual environment has significantly decreased external variables that can threaten the observed data's validity (Bushouse & Reeves, 2018). The use of software-based resources provided regulation to external factors that typically create hardware devices and media limitations. Hence, the attacks conducted for data collection provide insight into a reduced set of external variables in the implemented hypervisor environment (Browne et al., 2018). At this point, the only circumstance that could affect the devices' validity would be an outside contaminant from the internet. The VCloud hypervisor environment is available as a web application through a web browser. Such a contaminant could be observed through a browser attack during the time of the study. A second contaminant that could come from the Ethernet LAN's exposure to the internet could lead to software changes. External threats in both modes can skew the validity of the data collected. A live Xubuntu operating system running the latest patch updates limited the power of browser attacks and unauthorized handling of the VCloud interface from the host computer. Host modifications denied network share protocols and standard software used for remote desktop connections. Lastly, settings on the browser limited attacks on session-based cookies and tokens by deleting saved metadata from browsing. The network used with the host computer made use of encrypted VPN connections to limit host information revealed by the browser and encrypted all LAN traffic.

In light of interference from the internet provided by VCloud, the security of the virtualized application in the hypervisor was methodically observed by its limited access to the outside internet after the initial setup. The virtualized application was protected by a Pfsense firewall used to block incoming traffic to the LAN. The initial setup required secure and encrypted connections to operating system servers and installing all updated software applications for the Debian Linux-based Ubuntu and Kali distributions used in the experiment. Once all installations and upgrades were made to create the control group environment, the virtualized application remained disconnected from the outside internet. The interference of other users in the VCloud can always threaten the consistency of resource usage and can cause performance degradation (Nikounia & Mohammadi, 2018). At the time of data collection, a check on the blockchain network and incoming STP traffic was observed for consistency. Data for the attack was collected in two separate blockchain networks to ensure the framework's validity. Once the Ethernet LAN traffic stability, host network connections, and hypervisor resource availability were deemed satisfactory, the control group's attacks were repeated. The research solution was implemented into the Ethernet LAN to observe and collect data. Two forms of data were

collected: CPU usage and RAM utilization. *Vmstat*'s command provides access to CPU usage in a virtualized lab environment. The VM used for the STP DApp tool used 2 CPU cores, and the *vmstat* tool collects system measurements systematically from the *proc* file in the kernel. RAM utilization is measured with the active memory used, and the CPU usage is an added average of the two running cores using the userspace and kernel.

The hybrid framework was implemented to increase the search efficiency by using a current state blockchain ledger in Apache CouchDB that uses Merkle Tree hashing with SHA256 for validation and storage. Although there is no comparative data with a search in the blockchain, this data is meant to be compared to other validation methods rather than validation through other blockchains. At the time of this study, there were no other blockchains created to validate STP Root roles. The collection of this data is also meant to be analyzed for its efficiency in resource usage in the test environment. This data provides a basis for future research. The second form of data collected is the amount of time the STP DApp tool takes to detect the Root role change. The inspection process is measured against that of the tool. Lastly, the time required for a topology change after Yersinia stopped the forged frames is calculated from the Wireshark packet capture for perspective and more information about the lab environment. Documenting the threats to validity provides an opportunity to look at how the research has been conducted to answer the research problem with attention to details that can affect the data. All of these elements can affect the results of the study if not implemented with close attention. These last two sections provided more insight on how the STP DApp solution is implemented and how it fits with a valid test environment for the study. The next section takes a brief look at the solutions that were discussed in the Literature Review and how they compare to the STP DApp solution used in the experiment.

A Comparison of the STP DApp to Existing Solutions

The last section is an overview of internal and external threats to validity of the STP DApp. This section is another look at the previous solutions introduced in the Literature Review as compared to the STP DApp tool that was used for this study. It is a review of the cryptographic validation solutions that included a PKI and HMAC validation system for STP. The second section of comparisons are made with the STP alternates such as the Cisco configuration tools that block STP functionality, the Partitioned Infrastructure introduced by Yeung et al.(2006), an IDS/ DIDs solution and an SDN. Each of these solutions are defined with a comparison to the STP DApp tool for perspective. The last section of comparisons made are with the blockchain-

based solutions introduced in the Literature Review. They include a peer-to-peer IoT network, the Marconi Protocol and a blockchain-based approach to an SDN.

Cryptographic Validation

PKI – Trust-based switches with a hardware key that is validated by a CA server	Lai et al., (2014)	STP DApp validates using a CA server but validation is also done using a consensus algorithm which is geared towards the trustless nature of algorithmic validation.
HMAC – The symmetric key validates each Ethernet frame using a section of the BPDU payload. Each frame is validated using a password	Whalen and Bishop (2009)	STP DApp parses frames to perform external validation specifically for the STP Root-Takeover attack. HMAC requires agreement on a password which may not work in a larger environment

Alternates to the STP Protocol

Cisco – BPDUGuard, Root Guard, Loop Guard, BPDU filter	Cisco Systems (2007)	STP DApp does not look to block the functionality of STP on a granular level, rather, will add validation to required STP functionality.
A Partitioned Infrastructure – have an external LAN and internal LAN. The internal LAN will have a hidden topology with pseudo IDs	Yeung et al., (2006)	STP DApp parses each frame to detect and prevent an attacker from gaining root access. The solution works within a current infrastructure for validation. It does not modify the current STP protocol
An IDS/DIDS solution that detects anomalies by performing checks on the new switch. It keeps a record of its topology and tests the rogue for topology information.	Rai et al., (2011)	The STP DApp performs its detection by checking the root ID to see if its a valid network device using authentication at each endpoint. Attackers are capable of learning topology information at depth.
An SDN that does not use STP. It is another format for Layer 2 bridging. A central controller is used to inspect all network traffic in a central location due to the full virtualization of the network infrastructure.	Rietz et al., (2018)	The SDN approach is closest to the STP DApp and its test environment. STP DApp is a stepping stone that could apply to varied infrastructures before a full software-based approach.

Blockchain-based Solutions

BCR Protocol – Blockchain based IoT network for data exchange through peer-to-peer routing	Ramezan and Leung (2018)	STP DApp is geared towards the current infrastructure as a stepping stone. It does not use blockchain's peer-to-peer algorithms because it will take a long time to build and test. It works to use consensus algorithms for validation.
Marconi Protocol – rebuild the network infrastructure from Layer 2 up. Integrate encryption to secure all network traffic.	Marconi Foundation (2018)	The STP DApp is a small step towards introducing validation through encryption to fix a smaller issue. The rebuild the entire infrastructure is not feasible to fix immediate problems.
Blockchain Virtual Extensible LAN – This SDN approach builds on technology that exists in the a software based hypervisor environment. Like the previous SDN solution, this is a blockchain-based solution that fits with another SDN approach of the future	Amin (2017)	The STP DApp can work currently in any Ethernet LAN. It would require installation and resource testing, however, it can use validation on the currently used STP Protocol. The SDN approaches are ways to merge Layer 2 & 3 to eliminate the need for the STP Protocol

Summary

The last section provided a review of the literature analysis in the Review of Literature by reintroducing existing solutions to the STP Root-Takeover Attack and comparing them to the STP DApp tool. This section will provide a quick summary of this chapter. A look into the validity of the environment, its variables, and the measured data is part of taking a second look at the research problem and the methodology used to study it. Questioning the validity of the research also brings to question the choice of research methodology. Sometimes, during a study, one may surmise that a different research methodology would be appropriate to achieve conclusive results. For this study, a look into the research methodology through questioning its validity has provided more confidence in its application. Moreover, a comparative study of existing solutions for STP validation provides a perspective for the STP DApp. It is a unique research solution that takes the best of many worlds. The current infrastructure is based on a client-server model that requires a trusted authority (Gilder, 2018). To go against this fabric and build solutions that are difficult to implement will give rise to an idea that will not easily come to fruition. The reality of creating a solution that requires empirical test data fixes a network

security problem and uses the security elements of blockchain has required some compromise.

To integrate with the current network infrastructure, the requirement for a permissioned solution allows a research model that can be tested. It allows the researcher control to test the solution. Although it is not trustless under blockchain ideology, the use of validation makes the solution less trusted than a purely made of design and algorithms. The pre-experimental quantitative methodology has resulted in data that answered many questions posed in the research proposal. The permissioned, hybrid approach to a blockchain has allowed for an experiment that blocked an STP Root-Takeover attack in a software-based Ethernet LAN. The ability to perform this block provides a path for future study that can create solutions to other trust-based network problems. The requirement for measurements has provided a path for integrity in the research solution. It also created data that gives more opportunities for research and raises more questions. The choice of data is essential to the experiment because it guides the study in future research. The following chapter provides an overview of the study, details the experiment variables' limitations, and sparks the curiosity for future research ideas based on this research problem.

Chapter 4

Results

The previous chapter describes the research methodology used to answer the research question for this study. Can a reliable and trustless solution replace the trusted STP protocol, which leaves the Data Link layer open to STP Root-Takeover attacks? The research framework used to answer this question takes on three main distinctions: the approach, design, and methods (Creswell & Creswell, 2018). The quantitative pre-experimental research method provides a means to use the optimal procedure that guides data collection to gain perspective on the observation. This chapter will present a detailed analysis of the experimentation and its results after implementing the research solution. The results will be analyzed based on how it applies to the research question. Two questions were sought to be answered to provide value to the recorded measurements during the observation. Can the research problem be resolved by the trustless blockchain solution implemented in this study? How can this solution be implemented to increase its resourcefulness? In light of these questions, the measurements taken provide numerical data on CPU usage, RAM utilization, and environmental factors to provide insight into the experimental conditions.

Data Collection and Analysis

The pre-experimental method in the quantitative research approach requires two observations on one control group. The first observation sets the stage for the study by validating the research problem. In this study, the first observation was the STP Root-Takeover attack performed on the control group; an Ethernet LAN made up of three LEBs. The second observation included the STP DApp to manage the detection of the same attack and control group in the first observation. Data was not collected in the first observation because there was no apparent correlation between integrating the validation tool and the control group's attack. Instead, data was collected in Observation TWo before, during, and after the STP Root-Takeover attack on a baseline bridge and the STP DApp VM in the same Ethernet LAN as Observation One. The Ethernet frames were accessed by a popular packet capture program called Wireshark to provide

comparable data on frame captures. Lastly, the *vmstat*. the command was used to collect data on RAM and CPU utilization during the trials. This section will provide a detailed account of the data collected and an analysis of possible correlation and causality relationships detected in the data. The following two sections will provide details on the data sets and observations made on the CPU resource usage and time efficiency for the STP DApp.

Baseline: CPU and RAM Performance

The STP DApp runs traffic interception and parsing in the kernel's *dev.c* in the *net/core* folder. It intercepts the network traffic from the *__netif_receive_skb_core* function that pulls in traffic for reading (Kankipati, 2020). The *stpverify* mod isolates the BPDU frames and outputs the Root IDs to a *root_out* memory-based file in the kernel's *proc* folder (Kankipati, 2021). Simultaneously, the blockchain framework imports the Root ID and runs a Chaincode query to see if the Root ID is a validated device found in the blockchain's list of Root IDs (Adhav, 2020). The results are sent to a memory file in the kernel's *proc* folder called *root_in*. The *stpverify* modification in *dev.c* drops the frames that match the Root IDs marked as invalid. This cycle is repeated to collect information from all of the frames. While the validation tool is working, the *vmstat* tool is used to collect stats that are collected in the virtual machine's *proc* folder (Ljubuncic,2015).

In the data measurements, several processes are running that encompass CPU and RAM utilization. The *vmstat* tool is used to collect active RAM utilized in kilobytes and the CPU usage for the kernel and userspace (Malcher & Kuhn, 2019). The CPU usage is an average percentage of CPU utilization taken by adding each CPU core (Van Vugt, 2015). The kernel space utilization in the *stpverify* modification is based on network traffic inspection. Due to frequent authentication processes, the blockchain framework may be heavy in resource usage in the userspace. The literature review did not show a comparison with STP validation solutions and the time is taken to perform validation. A bridge in the experiment's Ethernet LAN was chosen as a baseline to perform the calculation because it does not have the kernel modification or the blockchain environment installed. For the baseline, CPU and RAM utilization measurements were collected in an Ethernet LANs bridge during each trial. The CPU Usage for the measurements shown is an average calculated by adding each CPU core. It is also the kernel and userspace usage added together. They are collected from the measurements collected in the virtual machine's *proc* folder and displayed using the *vmstat* tool (Ljubuncic,2015). There were six trials of attacks that were run for data collection in Observation Two. The *vmstat* tool was run every second for a hundred seconds. During this time, the STP DApp tool and rogue STP

Root-Takeover attack tool were running on separate virtual machines in the LAN.

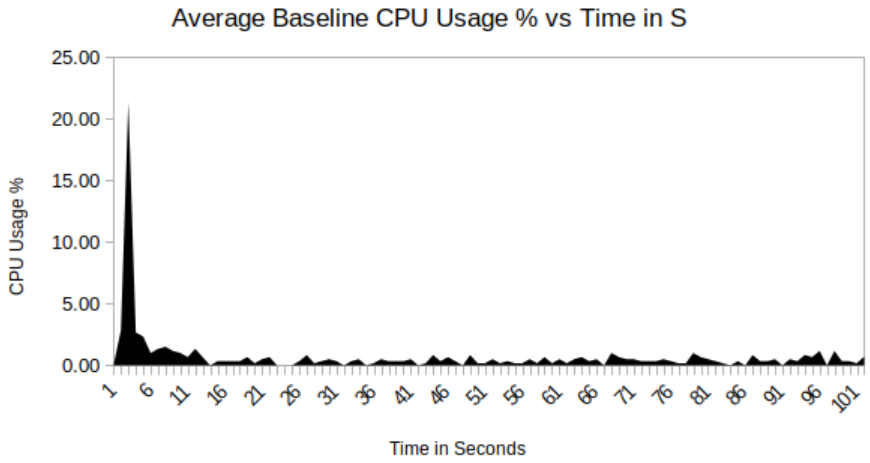


Figure 12. Average Baseline Total CPU Usage % for 100 Intervals

- Max Average CPU Usage:21%
- Min Average CPU Usage:0%
- Mean CPU Usage:0.71%

Figure 13 shows an average per trial for six trials of hundred intervals each. This provides a look at the CPU load over time to see if there is a variation in CPU utilization.

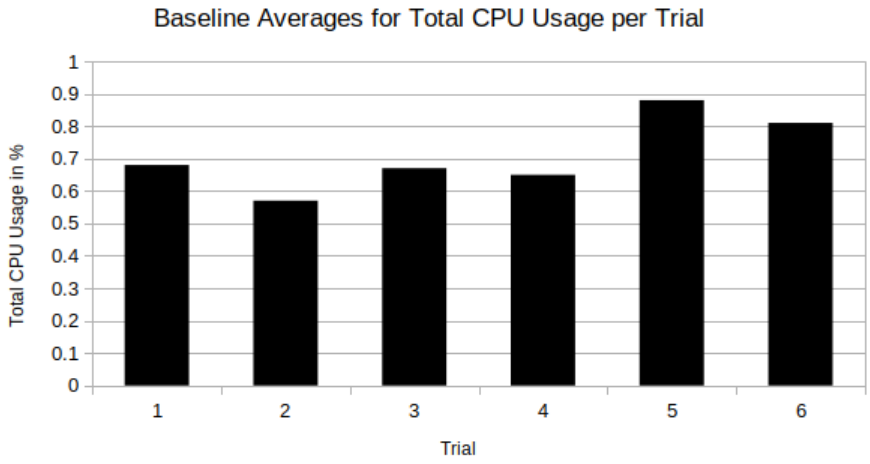


Figure 13. Average Baseline CPU Usage for each Trial

- Highest Average:0.88%
- Lowest Average:0.57%
- Variance:0.01

- Standard Deviation:0.11

The variance and standard deviation are calculated to show the CPU utilization trend during the six trials. The calculations show as follows:

Variance: s2 = SS(Sum of Squares)/(N – 1)(No. Of Trials) = 0.06/(6-1) = 0.01

Standard Deviation: s = √s2 (Square Root of Variance) = √0.01 = 0.11

This baseline shows the CPU utilization on a LAN bridge without the kernel modification or STP DApp. The vmstat tool has collected CPU usage for the system and kernel from the virtual machine's *proc* folder. These measurements were taken in one-second intervals over a hundred seconds. This baseline measurement was taken during six different attacks made on the Ethernet LAN. The bridge being measured does not have the STP DApp installed and is not running any validation tasks. The next baseline measurement shown is the active RAM utilized in kilobytes. This usage indicates the amount of active virtual memory while the data is collected. For the sake of perspective, this RAM measurement can be converted to gigabytes using the 1 KB = 0.000001 GB ratio (Schmidt, 2019). This conversion will later be utilized in the analysis of RAM utilization. It is collected from the virtual machine's *proc* folder and displayed using the *vmstat* tool. There were six trials of attacks that were run for data collection. The *vmstat* tool was run every second for a hundred seconds. During this time, the STP DApp tool and rogue STP Root-Takeover attack tool were running on separate virtual machines in the LAN.

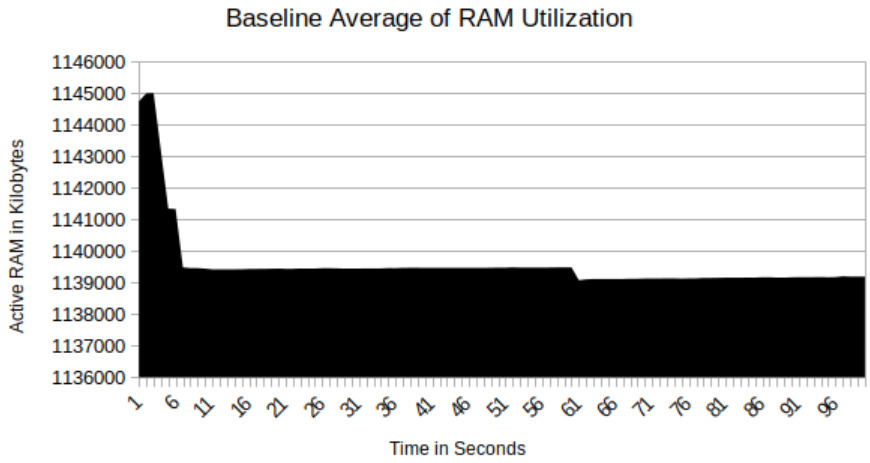


Figure 14. Average Baseline Total RAM Utilization for 100 Intervals

- Max Average Active RAM:1144993 Kilobytes
- Min Average Active RAM:1139069 Kilobytes

- Mean RAM Usage:1139565 Kilobytes

Figure 15 shows an average per trial for six trials of hundred intervals each. This provides a look at the RAM utilization over time to see if there is a variation over time.

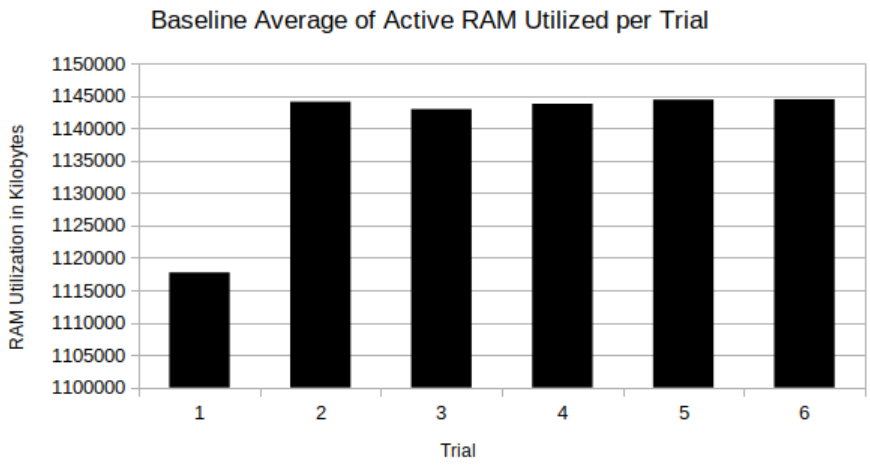


Figure 15. Average Baseline RAM Usage for each Trial

- Highest Average:.1144460 Kilobytes
- Lowest Average:1117737 Kilobytes
- Variance:.114653968 Kilobytes
- Standard Deviation:10708 Kilobytes

The variance and standard deviation are calculated to show the RAM utilization trend during the six trials. The calculations show as follows:

Variance: s2 = SS(Sum of Squares)/(N – 1)(No. Of Trials) = 573269842/(6-1) = 114653968 Standard Deviation: s = √s2 (Square Root of Variance) = √114653968 = 10708

This baseline shows the amount of active RAM utilization on a LAN bridge without the kernel modification or STP DApp. The *vmstat* tool has collected active RAM usage for a hundred intervals per second in six trials. The average RAM used out of six trials of a hundred seconds, with measurements taken per second, is 1139565 Kilobytes. This section provides a baseline for CPU and RAM utilization on a bridge in the control group. This bridge does not have the kernel modification and blockchain framework installed. It is used for data collection to provide comparative data for the STP DApp. The next section analyzes the measurements taken while the STP DApp is blocking an attack from the rogue attack tool from the VM with

the STP DApp. The STP DApp is a kernel interceptor and blockchain framework combined. It drops frames based on payload information invalidated by the blockchain.

STP DApp: CPU and RAM Performance

This section provides data collection results on a virtual machine with the STP DApp blockchain framework and kernel modification for BPDU frame interception. This section provides perspective on the number of resources consumed by the virtual machine using the blockchain framework. The results from this section can be compared to the baseline results collected in the last section. The number of resources required to run a blockchain increases with the number of peer nodes that are used (O'Dowd et al., 2020). This study provides a perspective for a general use case that utilizes consensus algorithms and endpoint validation with two peer nodes. Encryption algorithms used in the Hyperledger Fabric include public key pairs using SHA256 hash encryption (O'Dowd et al., 2020). Advanced Encryption Standard (AES) keys are used to utilize Smart Contracts in this framework (O'Dowd et al., 2020). The STP DApp provides a service that adds validation to the STP protocol's functionality in an Ethernet LAN. With blockchain validation, encryption and decryption are utilized by the framework on each BPDU frame that is parsed by the kernel. Encryption and decryption are CPU intensive; therefore, the measurement of CPU and RAM utilization by the STP DApp in the process provides valuable data (Novak et al., 2019). The interceptor bridge in the Ethernet LAN contains an STP DApp blockchain framework to validate the Root IDs and a kernel modification to intercept, parse and drop frames that are not valid according to the blockchain. During the STP DApp validation during an attack, the measurements taken for the baseline are the same as were taken for the baseline. This baseline measurement was taken on a non-STP DApp bridge in the LAN during the rogue attack. This observation indicates whether the blockchain framework creates high RAM and CPU utilization in the kernel and userspace. The data collection includes a hundred CPU usage and RAM utilization measurements taken by the *vmstat* tool per trial for six trials. The virtual machine housing the STP DApp is running on two CPU cores and 29GB of RAM. The CPU usage is the percent utilized in the kernel and userspace combined. This combination provides an overall view of the CPU usage of the STP DApp tool since it works in the kernel to intercept, parse frames to extract Root IDs, write Root IDs to *root_out* in the proc folder and block all frames matching the ID in the *root_i*n file. Figure 16 will show the average CPU Usage of the STP DApp in a hundred measurements per trial for six trials.

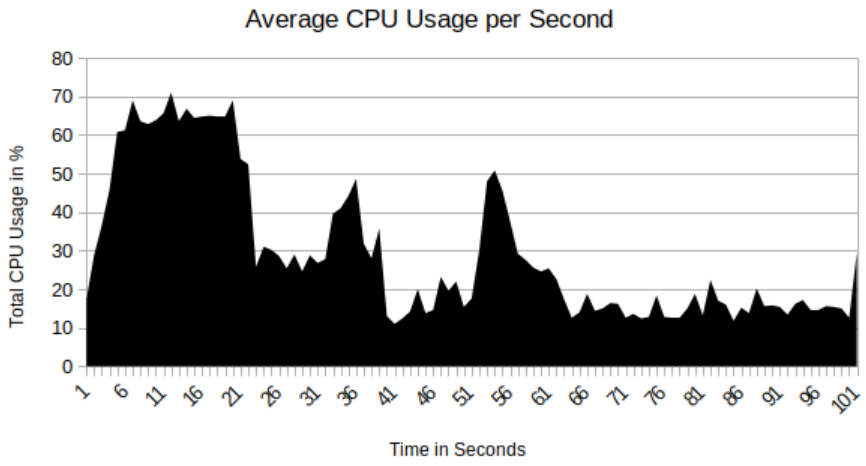


Figure 16. Average STP DApp CPU Usage for each Interval in Six trials

- Max average CPU Usage:71%
- Min average CPU Usage :11%
- Mean clock ticks:30%

Figure 17 shows an average per trial of a hundred intervals of one second each. This provides a look at the CPU load over a short time to see if there is a variation in CPU usage during the six trials. This provides insight into the behavior of the CPU while the STP DApp is running validation.

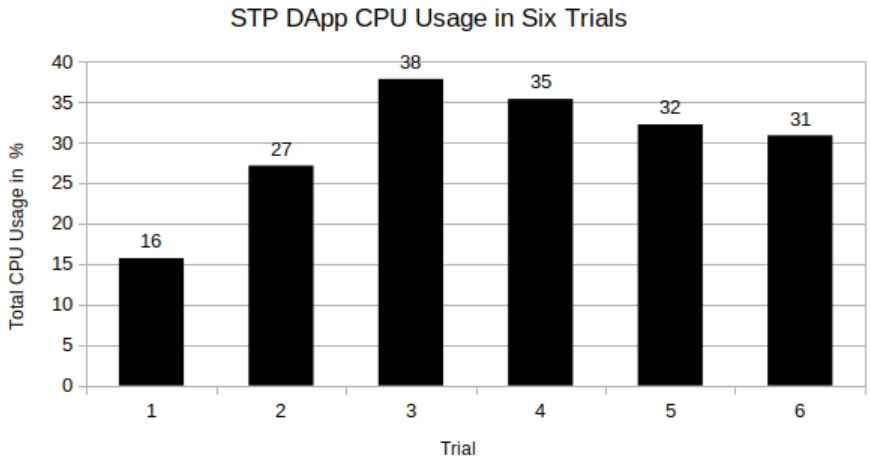


Figure 17. Average CPU Usage for each trial

- Highest Average:38%
- Lowest Average:16%
- Variance:60

- Standard Deviation:8

The variance and standard deviation are calculated to show the trend in a hundred CPU percentages in each of the six trials. The calculations show as follows:

Variance: s2 = SS(Sum of Squares)/(N – 1)(No. Of Trials) = 299/(6-1) = 60

Standard Deviation: s = √s2 (Square Root of Variance) = √60 = 8

The data collection shows that the average amount of CPU usage varies slightly while the STP DApp runs in the system. This correlation may be attributed to other processes that are run in the system. Another correlative trend shown in Figure 17 is that the number of CPU clocks increases from 16% CPU Usage in trial one to 38% CPU usage in trial three before steadily going back down by trial six. This data set marks an increase of 138% between trials 1 and 3, setting the variance up to 60. The increase in variance lowers the predictability of the CPU usage in this data set. The average CPU usage in the baseline is 0.71%. When the STP DApp is running, the average CPU usage goes up to 30%, which is a 4125% increase in CPU load. Both bridges are running 2 CPU cores at the time of data collection. This data shows that the STP DApp has significantly increased CPU usage. This trend may be attributed to a higher CPU load sent to the blockchain framework as it begins Chaincode validation in the blockchain ledger. The validation mechanism runs hash encryption and decryption at each endpoint, providing for the additional CPU load added by the STP DApp (Novak et al., 2019). In addition to CPU usage, the active RAM utilized during STP DApp validation was measured and collected using the *vmstat* tool in one-second intervals.

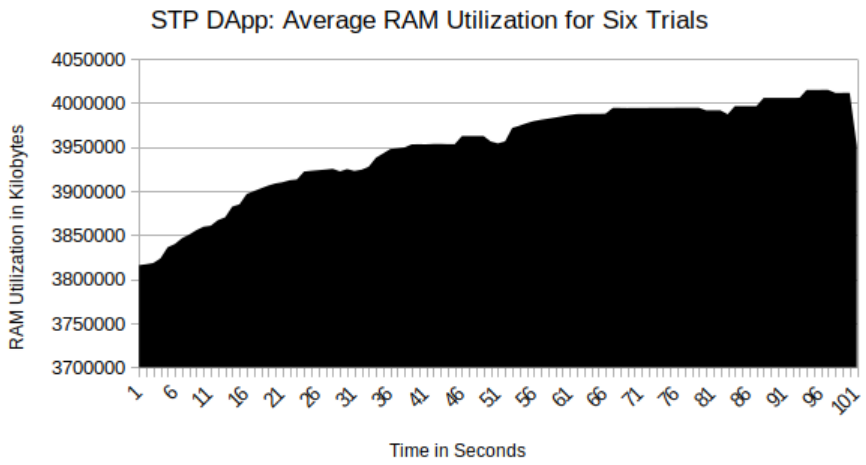


Figure 18. Average Active RAM Utilization in Six Trials

- Max Active RAM usage:4014874 Kilobytes
- Min Active RAM usage:3815978 Kilobytes
- Mean Active RAM usage:3950978 Kilobytes

The utilization of active RAM shows a slight increase over time. These measurements per trial show an average kilobyte of RAM taken per second for 100 seconds. The lowest average is measured in the first second of the trials, and the highest average is in the last second. The increase in RAM utilization is 5% over a hundred seconds. The data shows no spikes or events that show a marked increase or decrease in the amount of RAM utilized during the six trials.

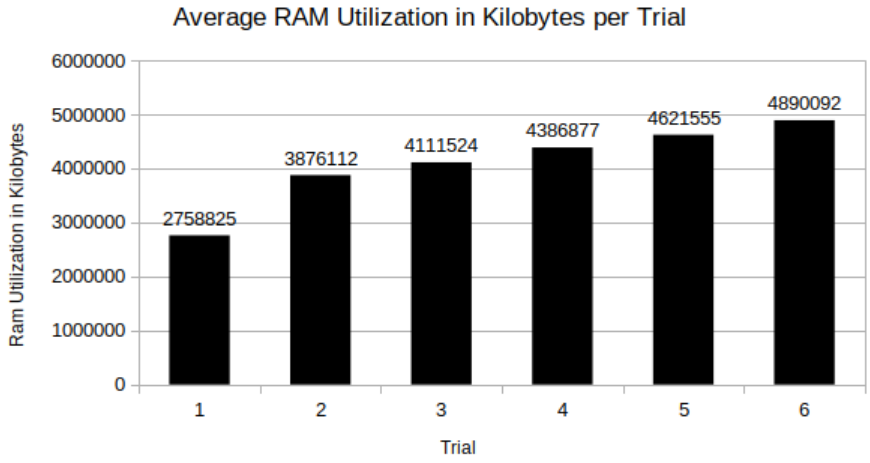


Figure 19. Average RAM Utilized for each Trial

- Highest Average:4890092 Kilobytes
- Lowest Average:2758825 Kilobytes
- Variance:565447028865
- Standard Deviation:4.2

The variance and standard deviation are calculated to show the trend in a hundred measurements of RAM usage in each of the six trials. The calculations show as follows:

Variance: s2 = SS(Sum of Squares)/(N – 1)(No. Of Trials) = 2827235144325.5/(6-1) = 565447028865.1

Standard Deviation: s = √s2 (Square Root of Variance) = √565447028865.1 = 751962.1

This data set variance increase is because of the 77% rise in average active RAM utilization from the first trial to the sixth trial. To put this information into perspective, a simple calculation can be made to convert the kilobytes to gigabytes using a conversion ratio of 1GB per 1,000,000 kilobytes (Schmidt, 2019). Although a correlation between RAM utilization and the STP DApp functionality may exist, any causal relationship between the two is unclear. This section included data collected on for RAM utilization and CPU Usage for the Observation Two attack with the STP DApp blocking it. There were six trials taken and measurements were recorded using the vmstat tool's stats every second for a hundred seconds. In each trial, a baseline reading was done on a non-STP DApp bridge and the bridge with the STP DApp and kernel modification installed. The following measurements were collected and analyzed:

- Baseline: CPU average- 0.71%
- Baseline: RAM average- 1.14GB
- STP DApp: CPU average- 30%
- STP DApp: RAM average- 3.96GB

The measurements show that the STP DApp usage shows a marked increase from the baseline bridge. The CPU usage increased by 4125% and the RAM utilization increased by 247%. The next section analyzes the amount of time difference between the arrival of the first and last forged frames while the STP DApp tool invalidates and drops the frames. The function to detect the absence of a bridge and remove it is based off of three hello_time message attempts according to Wojdak's (2003) modification of the STP protocol. It can be inferred that the amount of time required to receive a new valid Root ID would include the time required to remove the invalid device from the Ethernet LAN.

Time Efficiency

A software application's overall efficiency can be measured using resources and the amount of time it requires to function. These two variables are often used to measure the overall efficiency of an application. This section provides a comparison for future research and STP validation solutions. The first data set for time measurement calculates the detection speed of a forged BPDU by the STP DApp. Several methods are used to capture Ethernet frames in an LEB. The kernel modification called *stpverify* uses *dev.c's* internal function that intercepts network traffic to identify and parse BPDU frames for its elected Root ID. This Root ID is added to a *proc* file called *root_out*. The IDs are read and validated through the blockchain framework, and those that are not validated in the blockchain are written to a proc file called *root_in*. *Stpverify*

is then used to drop the frames that have this Root ID.

The kernel logs written to the */var/log/syslog* file in a Debian Linux operating system provides timestamps for all STP frames found and rogue Ethernet frames that are dropped. Stpverify modification writes this information to the kernel log using C's *printk()* function. Unfortunately, the Yersinia tool does not effectively provide system logs for getting the time for the first rogue frame's launch. Its associated Yersinia.log file does not contain timestamps for the launch of attacks. The Yersinia.log file only logs the time Yersinia is opened and closed with a notification of the network interface status. The exact time between the launch of the frames in Kali is only possible by manually tracking the time of frame launch as indicated in the GUI interface of Yersinia. Its detection in the STP DApp VM is available through packet analysis in Wireshark. The Wireshark tool is a direct approach for frame arrival times because it is also connected to the kernel to intercept and read network traffic (Bock, 2019). With Wireshark, it is possible to get the arrival time of the first bogus Ethernet frame and the time when the last rogue frame is captured. In six trials of data collection, the STP Root-Takeover is launched in Yersinia. The validation checks begin at the STP DApp tool, and Wireshark is used to note the arrival time for the first and last forged frame from Yersinia. Figure 20 provides the time the STP DApp takes to block the STP Root-Takeover attack from the Yersinia tool.

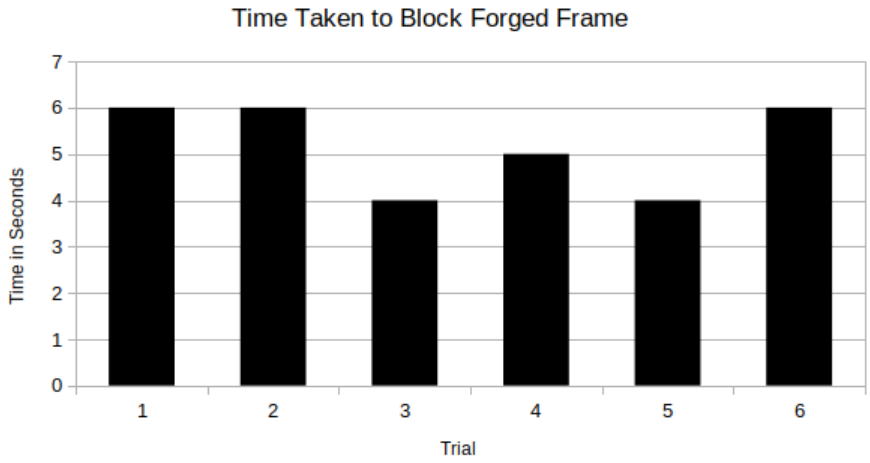


Figure 20. STP DApp: Time to Block the Forged Frame

The time required to block the frame was calculated by noting the Yersinia attack launch and a Wireshark frame analysis for arrival times. The noted times for each trial were the time of launch, the time of the first forged frame's capture, the time of the last forged frame captured, and the time of the frame with a new valid Root ID. In every trial, the time of the first forged frame's launch in Yersinia matched the first forged frame's capture in the STP DApp VM to a hundredth of a second. The Ethernet frame capture by the STP DApp is instantaneous, so there

was no chart made to show an event with no values. The next time collected was capturing the arrival time of the last forged frame in the STP DApp VM. The difference between the last forged frame and the first forged frame is calculated as the time taken to block the STP Root-Takeover attack. The results for six trials are shown in Figure 20. The next measurement of note is the arrival time of the first valid Root ID after the STP Root-Takeover attack. The time to revert to a valid Root ID is measured by the difference between the first valid frame and the last forged frame. Figure 21 provides the results of six trials and measurements drawn from the Wireshark packet analysis tool.

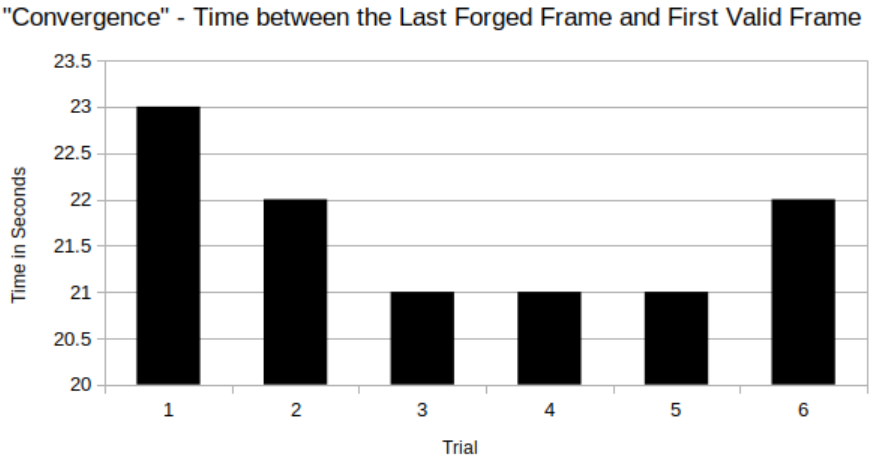


Figure 21. Time Taken for Valid BPDUs after Invalid Frames are Dropped

Figure 21 shows the amount of time it takes for a new valid Root ID to be captured in the STP DApp validation bridge. It is shown as a ”convergence” time; however, it is not official ”convergence” because that normally requires a TCN from all of the bridges in the LAN. Due to the nature of the study, the STP DApp was not fully deployed to every bridge in the Ethernet LAN to completely block the attack. The time to capture a new valid ID shows an average of 21.7 seconds over six trials. The time required to block the attack on the host with the deployed STP DApp tool and kernel modification is 5.2 seconds on average. The measured trials are based on the successful STP Root-Takeover attack in which the host bridge first accepts the forged frame before blocking its frames. Other trials, which remain unrecorded, blocked the acceptance of the forged Root ID completely. This section provided the results of six trials made using the STP DApp tool and kernel modification to block the STP Root-Takeover attack. The STP DApp and kernel modification’s CPU and RAM utilization on an Ethernet LAN bridge were collected. A baseline measurement was taken to provide comparative data on the resource usage of the research solution. Time efficiency measurements were taken from successful STP Root-Takeover attacks that were blocked within seconds to show the behavior of the detection

and prevention of an attacker that claims the Root role in the experimental Ethernet LAN. The next section is a summary of the data collection and its analysis. The summary will go over all of the results and what they imply.

Summary

With the nature of the blockchain framework and its mathematical processes, the assumption was that the STP DApp's use would show a significant increase in CPU utilization (Novak et al., 2019). The baseline CPU usage measured by the sum of all CPU cores in the kernel and userspace indicates an average of 0.71% used. On the other hand, on the host with the STP DApp installed with the kernel modification to manipulate Ethernet frames, the average usage was at 30%. This figure shows an average increase of 4125%. The correlation between the CPU resources and the STP DApp tool shows a causal increase due to the cryptographic calculations required by the system during validation at each endpoint. The average baseline figures versus the STP DApp host's figures for active RAM utilization shows a different pattern. The baseline average shows 1139565 Kilobytes or 1.14 GB of RAM utilized to run an average bridge in the control group's Ethernet LAN. The average RAM utilized while running the STP DApp and kernel modification shows an average of 3950978.6 Kilobytes or 3.95GB of RAM. The average RAM utilization indicates high usage, considering the experiment's blockchain runs on only two peer nodes. The findings also indicated a steady increase of RAM utilization of 5%. This finding creates a question as to the causality of this increase. Further, will such an increase continue with more trials, or will it plateau at a certain level of RAM usage?

After a look at the CPU and RAM utilization, time efficiency was measured for six trials. In these trials, the time of detection and prevention of the STP Root-Takeover attack was measured. The average time taken to block a successful STP Root-Takeover attack in this study was 5.2 seconds. The next measurement taken was the host's time to produce valid BPDU frames after it had accepted the forged BPDU frames. The time between the last forged frames before they were dropped and the first valid frame showed an average of 21.6 seconds. This chapter provides an interpretation of the data that was recorded by the STP DApp, Wireshark, and the NodeJS application. The data was based on trials that collected CPU clock ticks and system time logs on relevant events. The data is insufficient to provide an accurate depiction of correlative and causal relationships derived from the data. Instead, this data creates a basis to drive further research and study. The data provides a springboard for resource usage and the behavior of a blockchain framework tool that detects a change in network traffic. This chapter has provided some interesting results to CPU and RAM utilization by the STP DApp tool to

validate the STP-enabled LAN devices. It effectively prevents the STP Root-Takeover attack, and data collection shows its resource usage in doing so. The data collection also provided results for time efficiency in blocking the STP Root-Takeover attack. Measurements are taken in the acceptance and block of an attacker claiming the Root role shows a quick defense of the attack and returning to a valid Root within seconds of the attack. The next chapter wraps up this study with an overview of this research and its solutions. It will provide an overview of the assumptions and limitations that guided this journey and a detailed account of the research solution. The conclusion will include a security analysis of the research solution and ideas for future research based on this study.

Chapter 5

Conclusion

Overview

STP is a trust-based Data Link Layer protocol created as an automated algorithmic solution to provide network expansion. However, it does not validate new bridges' election, making it vulnerable to modern-day attack vectors that create a gateway for attacks that affect the entire network infrastructure (Lin et al., 2017; Strawn, 2019). One such attack vector is the STP Root-Takeover attack which allows the integration of new network devices to control the Ethernet LAN without proper validation (Mahmood et al., 2020; Rahalkar, 2018; Younes, 2017). The protocols that govern the services in the Data Link Layer of the network stack are not designed with security in mind. Consequently, the vulnerabilities incur a risk to every subsequent layer in the OSI referenced network stack. As a fundamental weak link, a security solution is recommended to maintain the network's integrity, with end-to-end secure communications starting from the Ethernet in the Data Link layer (Gilder, 2018; Strawn, 2019). The internet infrastructure as it stands today is the weakest link, and its solution is a trustless peer-to-peer based architecture for blockchain technology (Gilder, 2018). Carrying Bitcoin Core's blockchain's original ideology, the built-in consensus protocol in a permissionless blockchain is the basis for the efficiency and security in the blockchain system (Zhang et al., 2019).

This experiment was designed to test a potentially trustless feature to the STP protocol by using blockchain algorithms by introducing end-to-end encryption and immutability to Ethernet device records to block new Ethernet devices from entering without validation. This study introduces a hybrid blockchain framework that can emulate a large-scale peer-to-peer blockchain validation protocol in a controlled virtual Ethernet LAN environment. A modern attack tool called Yersinia was used to perform the STP Root-Takeover attack to prove the attack's validity on current Ethernet devices. The experiment's success indicated that little change was made to the network topology and devices to prevent the STP Root-Takeover attack's root cause.

The research solution to the STP Root-Takeover attack performed on the virtual Ethernet LAN is a hybrid permissioned blockchain that uses Smart Contract programming. Permissioned validation is emulated with a UI provided for a network administrator to add a new Ethernet device. A */net* module modification called *stpverify* is used to intercept STP-specific Ethernet traffic called BPDUs and drop invalidated ones to block STP Root-Takeover attacks. These BPDUs provide Root ID information which is queried and validated through a blockchain framework.

Limitations

The most exciting part of the Proposal Defense's journey to the Dissertation Defense's completion is working through research limitations. Some limitations were predictable because the ideas behind them did not seem feasible. However, as the research study was fine-tuned from the initial idea to the proposal defense, the research plan included implementing new types of software and programming languages. The research process sheds light on the reality behind a logical progression of an experiment. Typically, a procedure that seems standard may create unforeseen roadblocks. This section is an overview of significant roadblocks that have created challenges in achieving the research study's objective. These limitations are not ordered, and all of them have required forging a new branch in the research path. Part of this study was to test the validity of the research problem, and the implementation of the solution was uncharted territory. The procedure was taken step-by-step, partially by hope and part by luck. Once the research study was validated by verifying the research problem's significance, unforeseen problems with Ethernet frame access with userspace-based frame interception led to the creation of two versions of the STP DApp. The first one only provided detection of an invalid frame providing an IDS service for the STP Root-Takeover attack. Access to Ethernet frames in this version came from a socket API connected to the bridge interface. Although access to parse and extract the Root ID from the frames was possible, the ability to then functionality to drop those frames for Layer 2 STP protocol did not exist (Kankipati, 2021). The second version of the STP DApp uses a kernel modification called *stpverify* to directly access the frames through the *dev.c* application in the kernel's */net/core* module. The second version was an IPS that successfully blocked the STP Root-Takeover attack.

Trusted instead of Trustless

The purely trustless, immutable, and transparent blockchain system creates a large-scale infrastructure change to any current system it replaces. The peer-to-peer network is built over time through distributed resources worldwide that gain rewards from validating transactions. Such a network takes time and effort to build and is challenging to use in a controlled environment for scientific research. This limitation required a hybrid system called Hyperledger Fabric which automates the framework of a permissioned blockchain system. The blockchain framework is built in a localhost bridged network with two peers. The peer-to-peer validation in anonymous distributed networks is emulated through a UI for manual entry after validation. This framework is meant to provide an alternative means for Ethernet device validation through encryption and immutable device records. It introduces a foundation based on encryption with the potential of a peer-to-peer blockchain framework in a distributed environment. The use of this hybrid approach is easier to integrate into a small-scale modern network infrastructure. However, it can create a trustless network infrastructure that is not affected by the client-server model's trusted nature used in the current infrastructure. The trustless part of the STP validation tool is that an unrecognized Root ID will be dropped automatically.

Bridged Network Systems

The Hyperledger Fabric framework's hybrid system requires servers run in docker containers that create a local TLS based bridged network using the virtual machine's primary interface. This system posed challenges to the program used to intercept Ethernet bridges because the bind created to intercept traffic disconnected the blockchain framework's localhost bridge network system when using the first version of the STP DApp with a socket connected to the same bridge interface for network interception. This issue required that the interceptor directly access the kernel rather than the interface for binding and streaming traffic. This limitation was not a part of the research solution as it was a tool to intercept and filter network traffic for the BPDU frames used for the STP topology. The Hyperledger Fabric framework that requires the use of a bridged network of docker containers in localhost was used due to the limitations of creating a test environment to test the effectiveness and validity of a concept. Neither of these issues is related to the actual research problem or solution, but due to the framework and technology used to test the test environment's solution.

At the time of setup, the framework used to create the test environment's research solution seemed pragmatic. However, the Hyperledger Fabric network is designed for hybrid

adoption and does not have the simplicity of standard and traditional blockchains. It may be a quicker turn-key solution to demonstrate on a localhost network for permission-based validation, quicker searches, and an easy-to-use interface. However, the way it was constructed interfered with the use of the interceptor tool its network connectivity. In hindsight, the research of a more straightforward open-source blockchain may have been more effective.

Kernel Modification for Layer 2 Interception

This study required a lot of trial and error for the implementation of an interceptor. The interception and manipulation of Layer 2 traffic seemed to have little support on the userspace level with the current kernel version of Ubuntu 20.04 LTS at the time of the study. Several standard methods used to access and manage network traffic had failed. The first attempt was made with a Socket API connection using the *socket()*, *ioctl()*, and *bind()* functions to access header and Root ID information in Layer 2 traffic through the virtual machine's bridged network interface. Although the information could be accessed, common forms of dropping frames to perform the block of invalid frames were not functional. The effectiveness of firewall configuration tools like *ebtables* and *iptables* did not provide the proper documentation or support at the Layer 2-level. Moreover, the *ebtables* and *iptables* have been replaced by *nftables* which also has little support and no control of the traffic. For example, after accessing the network traffic through essential filtration with the socket connection, the ability to drop invalid frames did not pass any testing using *netfilter* configurations for ingress traffic (Kankipati, 2020). Inevitably, this research suggested that a socket connection to Layer 2 readily allows inspection rather than interception. The next attempt to directly manipulate the Ethernet traffic to the interceptor was with the *netfilter* hooks. The connection to the Layer 2 bridge functionality existed in the kernel */net* module; however, on testing, the access to STP network traffic was not available through the interface. There was documentation to support bridge layer traffic; however, the functionality did not pass testing because it did not pass the frame traffic to the userspace. Lastly, an attempt was made to intercept traffic directly in the *net/core/dev.c* file where frames are received.

The socket logic used in the IDS version of STP DApp was translated into the kernel's */net/core/dev.c* module with a few modifications, traffic manipulation was possible. The *goto drop* functionality that calls the function to drop Ethernet frames in ingress and structure them properly is the only option to block invalid Ethernet traffic from Layer 2 attack tools like Yersinia. The use of kernel modification made blocking invalid and forged Ethernet frames from Yersinia possible. In this section, some limitations made the ideal solution difficult to

implement; however, the journey towards arriving at unexpected conclusions provided much insight into carrying out a research study. For instance, the choice in methodology required experimentation and data with measurable results. Hyperledger Fabric's unique solution had surprising elements that allowed for an ideal test environment. As a corporate-based solution, it had features that made it practical to use in a software-based network infrastructure. Its integration with the modifications required in the kernel was almost seamless; it allowed for an easy infrastructure to adapt to the research study. As the future brings software-based network infrastructures, the hardware-based protocols that exist due to hardware-based problems can be phased out (Amin, 2017; Rietz et al., 2018). In the end, many of the limitations led to more discoveries and opportunities for research. The following section will provide a deeper analysis of the STP DApp solution and its limitations.

Security Analysis of the STP DApp

The STP DApp solution was created to provide validation to STP-enabled Ethernet LANs. A significant contributor to attacks in the network infrastructure is based on the vulnerabilities in the Data Link Layer (Younes, 2017). This research aimed to mitigate the STP Root-Takeover attack using a form of validation used by the blockchain protocol. The blockchain protocol is known for its immutability in securing data (Nakomoto, 2009). The concept of immutability is about data integrity. Immutability does not allow easy modification, and in blockchain, all modifications are trackable (Kereki, 2020; Lantz & Cawrey, 2020). Immutability due to the consensus algorithm that provides Merkle tree hashing gives blockchain a unique value add for validation (Parisi, 2020). Validation can be made from various lists available for checks in a network devices forwarding table to a standard database. However, the data integrity in that table is not certain unless it is secured in a proven way. To this day, applications are created with security as an add-on. The concept of *security by design* is where an application is built with security as an integral part of a strategy for secure hardening (Deogun et al., 2019). Blockchain is a proven technology that is *secure by design* and has security mechanisms that can ensure less vulnerability than average data storage provides (Nakamoto, 2009). The STP DApp is built to be *secure by design* using blockchain consensus algorithms for validation. The inevitable direction of the research study's methodology requires a working product that can be used in experimentation as the pre-experimental quantitative approach requires (Creswell & Creswell, 2018). The choice of a Hyperledger Fabric framework to use various blockchain elements without the limitation of the peer-to-peer distributed node technology provides signif-

icant security using consensus protocols for immutability. It is different from Bitcoin Core and Ethereum's decentralized networks. However, Gaur et al. (2020) describe Hyperledger Fabric's unique combination of security elements:

- It employs the use of certificate authorities (CA) to manage the access of users
- SHA256 hashing and Elliptical Curve Cryptography (ECDSA) to use for consensus validation on the data's stored contents.
- Digital signatures are tracked for effective non-repudiation.
- Distributed file systems have a copy to prevent ransomware attacks.
- Smart Contracts are used to ensure the integrity of business logic.
- Its permissioned nature adds trust, and different variations of it form a modified consensus algorithm.
- All network nodes are built with a modular design that requires endpoint authentication at each use which makes it different from the average trust-based server infrastructure.

Due to these security elements, the inherent security built into the Hyperledger Fabric's blockchain allows for modifications that may alter the security ideology from which it is based; however, it is built with *security by design* (Gaur et al., 2020). A kernel modification called *stpverify* compiled into the Linux Ethernet Bridge (LEB) allows it to parse all network traffic for BPDU frames that provide information about the Root ID. The Root ID is the identifier for the bridge that controls the Ethernet LAN's topology. *Stpverify* puts these Root IDs into a file called *root_out* which the STP DApp reads to validate each ID as part of the Ethernet LAN. If one is unrecognized, it is put into a *root_in* file where *stpverify* will drop the ingress of the frames, so it is completely removed from the Ethernet LAN's topology. This action blocked Yersinia's STP Root-Takeover attack in Observation Two. The STP DApp provides a specific role in blocking STP Root-Takeover attacks performed by the Yersinia tool. With this tool, BPDU flooding, claiming the Root role, and bridge priority attacks are constructed. A function for the attack sniff's frames to gain access to the Ethernet LANs topology.

Based on the BPDU frame's information about the current Root, the attacker behind Yersinia decrements its ID to gain automatic acceptance in the LAN as Root role. This tool can be used for any attack that uses a different Root ID from the validated ones. The STP DApp is not created to prevent MAC spoofing, MITM or DoS attacks by Yersinia. It only validates the Root ID and no other metadata that comes from the BPDU frames. This tool can block malformed BPDU frames, and Root hijacking. If a frame is sent with the same Root ID, the current solution

will accept the lesser priority frame unless it has a decremented Root ID. This solution has only been tested in the Ethernet LAN in a virtualized environment using Yersinia. The results may be different in a non-virtual environment or with a different attack tool. Currently, the experiment is limited to a virtual LAN using LEBs configured with the STP DApp to block an attack to claim the Root role from Yersinia installed on a Kali VM. It has not been tested in hardware-based environments, so it is difficult to predict whether it will have the same ability to block attacks in the current test environment. Lastly, the STP DApp requires full deployment in the Ethernet LAN to be fully effective. A full deployment requires that each bridge in the Ethernet LAN will require the kernel upgrade with the stpverify modification and Hyperledger Fabric network in the NodeJS runtime environment to be fully effective. A full deployment in the Ethernet LAN ensures that all of the bridges in the LAN will block an invalid Root ID to prevent repeated attacks or a divided LAN. (Marro, 2003; IEEE, 2016).

Contribution

The integration of the security-based algorithmic framework of blockchain is at its grassroots worldwide. As a new technology at its birth of adoption, it is only at its research and development phase. As it stands, in large-scale adoption of corporate systems, a complete upheaval of the current infrastructure is required. A proven use case is difficult to find; however, the technology can solve many Cybersecurity problems. The root cause of several large-scale attacks may be prevented by manually validating Ethernet devices in a hybrid blockchain system. Infrastructure changes are required for complete validation in a peer-to-peer network. Such a change would prevent the MITM attack and Mac spoofing. The prevention of these two attacks would allow an automated validation system for the STP Root-Takeover attack. This study is strictly based on the STP Root-Takeover attack in the data link layer. The prevention of MAC spoofing and the MITM attack is a solution for a different research problem. Large-scale automation would require a total transformation of the entire network infrastructure to validate and encrypt. Implementing a different network infrastructure is not an easy solution and a costly one, but every new technology starts somewhere even if its just an artist's conceptualization through science fiction.

Cybersecurity faces a problem because the basic infrastructure for the transmission, maintenance, and storage of digital data is not inherently secure. With the increase of human reliance on digital information comes the increase of fraud and cyber-based attacks. An infrastructure change with new technology is inevitable to meet the increasing demand for digital

information and automation. Creating a new infrastructure takes decades to consider accessible use cases, but this technology has the potential to create solutions for today's Cybersecurity issues. Permissionlessness, trustlessness, and immutability are creative new security solutions, and research may drive its adoption. As a step towards adopting inherently security data transport and storage is imminent, a hybrid approach in some environments is better than doing nothing and could secure its user from common attacks. As these hybrid solutions are adopted, and cyberattacks are created to circumvent them, the demand for a larger scale change will be imminent. Currently, blockchain can be used to solve security issues at the Physical Layer with identity management, the Data Link Layer by trustless validation, and the Network Layer with layer-based encryption integrated into the infrastructure. Better database management using a network of encrypted transmission with modular storage devices are possible with the Hyperledger Fabric network and other blockchain systems. This hybrid approach challenges the researcher to imagine a secure network topology, a new hybrid from the bottom up rather than application layer security patching used to fix new cyber attacks as they come. It will challenge the Cybersecurity community to see security as a foundation rather than persistent fires that are put out one at a time. Adopting secure-by-design solutions such as blockchain will increase like the ripple effect in the Merkle tree system of validation.

Recommendations

This study provides a unique perspective to secure transactions and storage in a web application designed to validate Ethernet devices. The applicability of such a perspective can be used to solve many security issues with trust-based systems; however, from a networking perspective, this framework provides an effort to dissolve the trusted nature of the Data Link Layer by forcing the validation of all network traffic. Although firewall filters can be used for the same purpose, it is not straightforward or efficient. The framework of blockchain algorithms in this study introduces elements that provide recommendations to solve trust in the STP protocol and its effectiveness in maintaining a network of valid Ethernet devices. Elements unique to a blockchain framework such as encryption, validation, and immutability provide support for security as inherent and not as an add-on. The added hybrid approach provides a pragmatic solution to use as a stepping stone for testing and further research. It speeds up queries by adding quicker traversal of an encrypted current state database separate from the blockchain and a user interface for permissioned management by institutional users.

Encryption

All blockchain frameworks use encryption for data transmission and storage. The web application created by the Hyperledger Fabric framework uses SHA256 hash algorithms for PKI-based authentication for users and servers in the network for every transaction. Java-based token authentication is used as is standard for a web application in the NodeJS runtime environment. The use of encryption inherent in the bridged network connections for the blockchain system provides security as a foundation. The use of token authentication in the Hyperledger Fabric framework emulates the inherent nature of encryption in a traditional blockchain. The framework uses a database that matches the current state of the blockchain. It is comprised of a hashed list of transactions that is encoded/decoded with token authentication. All stored data is encrypted and can be decoded when managed in the UI for accessibility.

Validation

The root cause of the STP Root-Takeover attack is the lack of validation of new network devices added to the Ethernet LAN. This concept is called a permissioned blockchain (Nakamoto, 2009). The STP protocol is a trust-based automation algorithm for LAN topology management. Validation provides a quasi-trustless solution that requires trusted users to manually validate and add new bridges to the Ethernet LAN. The STP DApp in the test environment creates a hybrid approach to trustless validation by adding valid devices to the network with manual management through a UI. This research solution can provide control over the LAN devices and a software solution for a network administrator to add valid devices to the network in a small network. A BPDU frame is automatically invalidated and blocked if the Root ID has not been added to the blockchain. Although it is trusted in nature at this scale, it moves towards a trustless framework bound by secure automation.

Immutability

Immutability is unique to the blockchain framework and is a secure solution to preserve data integrity in storage. Every transaction that is stored in the blockchain is hashed and saved as a hash. Its predecessor's block will have a hash in its header to incorporate the new block. The Merkle tree algorithm is how transactions in the blockchain are validated, so there is no way to delete a blockchain transaction. Such a characteristic of security provides secure record manageability. Hybrid solutions Although the Hyperledger Fabric does not have the exact characteristics of the traditional blockchain, its emulation of these characteristics provides an

approach to securing digital information without the complete upheaval of the infrastructure. A hybrid solution is a pragmatic approach to building a foundation with enhanced security without the expense of using a traditional blockchain. These solutions are also under research and development but provide options to test secure records management with a web application interface for usability. It also provides solutions to lower resource usage for efficient use on a large scale. Two such solutions for usability and integration in current environments are the NodeJS runtime environment and docker containers to manage a secure framework. The second is the addition of a current state database to avoid a blockchain query, which is not time efficient.

Recommendations for Future Research

Creating testbeds for the grassroots blockchain technology sets the stage for research into new security solutions with encryption algorithms. The use of a hybrid approach of this grassroots technology to solve a problem provides a means for pragmatic solutions that are easier to experiment with and test. Although the solution came with many problems, the learning component came with ideas for future research. The solution was to test a hybrid framework to emulate a difficult technology test in a controlled environment. The test to manage network traffic by blocking STP frames with unknown Root IDs can solve a standard attack using a different framework. In the same light, this introduces the technology to solve many Cybersecurity problems. The research solution demonstrates how a blockchain solution can increase the Confidentiality, Availability, and Integrity of an Ethernet LAN. It provides Confidentiality with SHA256 PKI encryption at every endpoint and Availability by preventing a rogue device from taking Root in an Ethernet LAN. The inherent immutability of the blockchain system with Merkle Tree and consensus algorithms provide the Integrity required to provide a fully secure cloud-based database system. This type of research provides a foundation for robust solutions to solve more significant problems.

There is a curiosity spurred by using the Hyperledger Fabric framework in the business sector (Parisi, 2020). It works for corporate research proposals but is not a stable solution in terms of security and growth. It is open source but full of third-party dependencies. All of these dependencies are difficult to manage and update for vulnerabilities. A more robust framework could be run without constant updates and dependencies if it does not use the NodeJS runtime environment and its libraries. A stable blockchain that integrates the authentication and transactions with the SHA256 algorithm and a local test network would provide an even more secure foundation for the blockchain system. Resource usage can be lowered with a tool that was not

in two parts but rather, one system built in C. Although the UI and database added usability, it added bulk to the solution that was not required. This tool was modified to open source systems to create a unique solution to a common problem. The interaction between an interceptor tool written in C to interact with a modified Hyperledger Fabric template causes inefficiency issues and is not streamlined.

Although the MITM attack was not a part of the solution, it is a gateway that is an even bigger problem for the Data Link layer. Further filtration in the interceptor could be made to prevent some common Mac spoofing attacks. The STP validation tool can also be made to identify all source addresses for frames to identify the bridge IDs for all LAN devices. All traffic can be stopped from an unidentified address to make it a more effective tool for attacks. A prevention system would make this tool much more potent as a type of layer two firewall approach. The solution could also be integrated with a firewall. The STP validation tool is made to replace the trusted nature of the BPDU frames. It is also made to replace the use of the MAC table, which is referenced in switches. Layer 3 devices use ARP tables, and such a validation tool could be configured to filter IPv4 traffic at the interceptor to validate IP addresses. In this light, blockchain systems can be researched for other layers in the network infrastructure.

The Ethernet LAN was run with the Linux Ethernet Bridge, which provided traffic at the Application Layer. Although the LEB is a switch, it is also run in the Application Layer. Hence, the LEB provides an ideal platform for a userspace tool with an Application Layer blockchain. Effectively, if the blockchain is run within the LEB with a kernel-based interceptor, it is technically compiled as a switch with a blockchain inside of it. As they move towards virtualized switching increases for the large-scale enterprise, an Application Layer switch may meet future demand in the increasingly popular SDNs.

The solution could also be tested with hardware Ethernet LANs made up of Cisco switches using the STP topology. It could be measured against the *Root Guard* and *BPDUguard* technology used to replace STP in some Ethernet LANs. Could such a solution be integrated into Cisco IOS to add an inherent add-on to the switch in Layer 2 rather than in an outside interceptor in the LAN?

Summary

Although at the grassroots level, blockchain technology is growing as a solution to today's problems. It is well known for providing a solution to today's economy. Not only does it provide a platform for a global currency, but it also solves solutions in institutional economic

systems that cause inflation. Each aspect of the blockchain system of protocols solves a different problem. The network infrastructure was built with protocols made to expand and speed up networks. The automation of Ethernet LANs through the STP protocol was a breakthrough that aided in today's global network. Unfortunately, the grid-based network system infrastructure uses the same technology and protocols it has had for a long time with no real efficient solution for a change. The 'if it ain't broke, don't fix it" approach is not is now sounding more like "it's broke, how do we fix it?" with cyberattacks on the rise. Although it is a solution, the biggest problem with integrating blockchain into any system is the expense of a change in the infrastructure.

On the other hand, cyberattacks caused by the current network infrastructure are also costing businesses money. At the moment, most research and development have been seen in industries that require blockchain protocol for risk management. Examples of these industries would include banking and healthcare. Although the research problem tested in the study is for the STP Root-Takeover attack and the STP protocol, a similar solution can be for other network attacks. It also prevents attacks to subsequent layers in the network by keeping control of the Ethernet LAN. This study is meant to validate the attack is still an issue on modern systems and provide a unique solution that can prevent it. It is also meant to bring to light a problem with trust that requires a unique solution that can be researched and improved. Due to the increased reliance on digital information and wide-scale use of the network infrastructure, security solutions have to catch up with mainstream technology. Blockchain provides an approach that would provide integral security as part of the infrastructure rather than providing add-ons to patch up a problem until another one pops up. The network infrastructure problem is similar to the story of the three little pigs and the big, bad wolf. The most industrious pig built a house of a strong foundation and was not eaten. Where the pigs using accessible things like hay, mud, and sticks continually patch up holes and put out fires created by the big, bad wolf.

References

Abuelenain, K., Doyle, J., Karneliuk, A., & Jain, V. (2021). *Network Programmability and Automation Fundamentals*. Cisco Press.

Adhav, P. (2020, April 5). *Introduction to Course :Create Basic Network with Hyperledger Fabric v2.0.* Retrieved from https://www.youtube.com/watch?v=SJTdJt6N6Ow&list=PLSBNVhWU6KjW4qo1RlmR7cvvV8XIILub6

Amin, M. (2017). *Blockchain: The Hidden Gem of Data Networking: BLAN*. ASIN.

Anu, P., & Vimala, S. (2018). A Survey on Sniffing Attacks on Computer Networks. *Proceedings of 2017 International Conference on Intelligent Computing and Control, I2C2 2017*, *2018-Janua*, 1-5. doi: 10.1109/I2C2.2017.8321914

Aumasson, J.-P. (2017). *Serious Cryptography*. No Starch Press.

Bartlett, J. (2016). Cypherpunks Write Code. *American Scientist*, *104*, 120. Retrieved from https://www.americanscientist.org/article/cypherpunks-write-code doi: 10.1511/2016.119.120

Bashir, I. (2020). *Mastering Blockchain - Third Edition*. Packt Publishing.

Bazari, A. S., Aggarwal, A., Asif, W., Lestas, M., & Rajarajan, M. (2019). Node Criticality Assessment in a Blockchain Network. *BlockSys 2019 - Proceedings of the 2019 Workshop on Blockchain-Enabled Networked Sensor Systems*, 22-27. doi: 10.1145/3362744.3363343

Bell, D., & Padula, L. L. (1976). *Secure Computer System: Unified Exposition and Multics Interpretation.*

Blenk, A., Basta, A., Reisslein, M., & Kellerer, W. (2016, 4). Survey on Network Virtualization Hypervisors for Software Defined Networking. *IEEE Communications Surveys Tutorials*, *18*. doi: 10.1109/COMST.2015.2489183

Bock, L. (n.d.). *Learn wireshark*. Packt Publishing.

Boonkrong, S. (2020). *Authentication and Access Control: Practical Cryptography Methods and Tools*. Apress.

Bresnahan, C., & Blum, R. (2019). *CompTIA Linux+ Study Guide*. Wiley. doi: 10.1002/9781119556060

Browne, A. F., Watson, S., & Williams, W. B. (2018). Development of an Architecture for a Cyber-Physical Emulation Test Range for Network Security Testing. *IEEE Access*, *6*. doi: 10.1109/ACCESS.2018.2882410

Bull, R. L. (2016). A Critical Analysis of the Layer 2 Network Security in Virtualized Environments.

Buterin, V. (2014). A Next-Generation Smart Contract and Decentralized Application Platform. *Etherum*, 1-36. Retrieved from http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf

C, S. (2016). *VMware NSX Network Essentials*. Packt Publishing.

Chantzis, F., Calderon, P., Deirmentzoglou, E., & Woods, B. (2021). *Practical IoT Hacking*. No Starch Press.

Chu, A. (2019). *CCNA Cyber Ops SECOPS - Certification Guide 210-255*. Packt Publishing.

Cisco. (2007, 5). *Configuring Spanning Tree PortFast, BPDU Guard, BPDU Filter, UplinkFast, BackboneFast, and Loop Guard Cisco* . Retrieved from https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4000/8-2glx/configuration/guide/stp_enha.html

Cisco, A. N. (2020). *Introduction to Networks Companion Guide (CCNAv7)*. Cisco Press.

Cohen, E. A., & Kahn, D. (1997). The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. *Foreign Affairs*, *76*, 129. doi: 10.2307/20048054

Connelly, L. M. (2014). *Use of Theoretical Frameworks in Research* (Vol. 23).

Conti, M., Dragoni, N., & Lesyk, V. (2016). A Survey of Man in the Middle Attacks. *IEEE Communications Surveys and Tutorials*, *18*, 2027-2051. doi: 10.1109/COMST.2016.2548426

Cox, K., & Gerg, C. (2004). *Managing Security with Snort IDS Tools*. O'Reilly Media, Inc.

Creswell, J. W., & Creswell, J. D. (2018). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches Fifth Edition*. Sage Publications.

Dasari, M. (2017). Real Time Detection of MAC Layer DoS Attacks in IEEE 802.11 Wireless Networks. *2017 14th IEEE Annual Consumer Communications and Networking Conference, CCNC 2017*, 939-944. doi: 10.1109/CCNC.2017.7983259

Davidson, S., Filippi, P. D., & Potts, J. (2018). Blockchains and the Economic Institutions of Capitalism. *Journal of Institutional Economics*, *14*, 639-658. doi: 10.1017/S1744137417000200

Davies, G. (2019). *Networking Fundamentals*. Packt Publishing.

Day, J. D., & Zimmermann, H. (1983). The OSI Reference Model. *Proceedings of the IEEE*, *71*, 1334-1340. doi: 10.1109/PROC.1983.12775

DeJonghe, D. (2018). *Load Balancing in the Cloud*. O'Reilly Media, Inc.

Deogun, D., Johnsson, D. B., & Sawano, D. (2019). *Secure by Design*. Manning Publications.

Dooley, J. (2013). *A Brief History of Cryptology and Cryptographic Algorithms*. Springer

International Publishing. doi: https://doi.org/10.1007/978-3-319-01628-3

Dotson, C. (2019). *Cloud Security Fundamentals*. O'Reilly Media, Inc.

Easttom, C. (2018). *Network Defense and Countermeasures: Principles and Practices* (3rd ed.). Pearson IT Certification.

Embry, J., Manson, P., & Milham, D. (1990). An Open Network Manage Architecture: OSI/NM Forum Architecture and Concepts. *IEEE Network*, *4*, 14-22. doi: 10.1109/65.56547

Evans, C. L. (2018). *Broad Band : The Untold Story of the Women Who Made the Internet*. Portfolio.

Fall, K. R., & Stevens, R. W. (2011, 11). *TCP/IP Illustrated, Volume 1: The Protocols* (2nd ed.). Addison-Wesley Professional.

Farooq'i'Azam, M. (2016, 10). A Comparison of Existing Ethernet Frame Specifications. Retrieved from `http://arxiv.org/abs/1610.00635`

Fenrich, K. (2008, 2). Securing Your Control System. *Power Engineering*, *112*.

Forshaw, J. (2018). *Attacking Network Protocols*.

Foundation, M. (2018). *Marconi Protocol.*

Gaur, N., O'Dowd, A., Novotny, P., Desrosiers, L., Ramakrishna, V., & Baset, S. (2020). *Blockchain with Hyperledger Fabric - Second Edition*. Packt Publishing.

Gerald F. March, S., Salvatore T. (1964). Design and natural science research on information technology. , *10*, 501-509.

Gilder, G. (2018). *Life After Google: The Fall of Big Data and the Rise of the Blockchain Economy*. Gateway Editions.

Gooley, J., Edgeworth, B., Hucaby, D., & Rios, R. G. (2019). *CCNP and CCIE Enterprise Core ENCOR 350-401 Official Cert Guide*. Cisco Press.

Gregg, M., & Watkins, S. (2006). *Hack the Stack : Using Snort and Ethereal to Master the 8 layers of an Insecure Network*. Syngress Pub.

Gómez-Arevalillo, A. D. L. R., & Papadimitratos, P. (n.d.). Blockchain-Based Public Key Infrastructure for Inter-Domain Secure Routing. *International Workshop on Open Problems in NetworkSecurity (iNetSec)*.

Ham, M. (2017). BGP Route Attestation: Design and Observation using IPV6 Headers. *Masters Theses & Doctoral Dissertations*. Retrieved from `https://scholar.dsu.edu/theses/308`

Haughwout, J. (2017). *Blockchain: A Single, Immutable, Serialized Source of Truth.*

Howser, G. (2020). *Computer Networks and the Internet: A Hands-On Approach*. Springer Nature. doi: https://doi.org/10.1007/978-3-030-34496-2

Huffman, F. (2013). *Practical IP and Telecom for Broadcast Engineering and Operations*. Routledge.

Hughes, E. (1997). *A Cypherpunk's Manifesto.* 978-0-471-12297-5.

IEEE. (2016). IEEE Standard for Ethernet - Section Two. *IEEE Standard for Ethernet*, 1-400. doi: 10.1109/IEEESTD.2012.6419735

IEEE. (2018). *IEEE Standard for Ethernet.* IEEE-SA Standards Board. doi: 10.1109/IEEESTD .2012.6419735

Kankipati, K. (2015, November 4). *Linux Kernel Networking Sub-system - episode4 net/core/dev.c - Packet Transmit, Receive APIs.* Retrieved from `https://www.youtube .com/watch?v=5k1et72efdY`

Kankipati, K. (2020, August 7). *0x20a Adding your own Kernel Modules into Linux Kernel Source | Part 1 | Linux Kernel Programming.* Retrieved from `https://www.youtube .com/watch?v=J4W2ZvROfZw`

Kankipati, K. (2020, September 3). *x20f Code with Kiran - Live Coding | Linux Kernel Programming | Kernel Libraries | Part 1.* Retrieved from `https://www.youtube.com/ watch?v=_OLwS79Z_SI&list=PL2TXDotVKyDAs76VRlBeGLRfbFsfW3Kxo`

Kankipati, K. (2021, March 3). *x231 Porting my /proc old Linux Kernel code to new Linux Kernel version OperatingSystem Porting.* Retrieved from `https://www.youtube.com/ watch?v=Cx3TN0iFdd0`

Keeriyattil, S. (2019). *Zero Trust Networks with VMware NSX*. Apress. doi: 10.1007/978-1 -4842-5431-8

Kereki, F. (2020). *Mastering JavaScript Functional Programming - Second Edition*. Packt Publishing.

Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC: Keyed-Hashing for Message Authentication.*

Lai, Y., Pan, Q., Liu, Z., Chen, Y., & Zhou, Z. (2014). Trust-Based Security for the Spanning Tree Protocol. *Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS*, 1338-1343. doi: 10.1109/IPDPSW.2014.150

Lammle, T. (2020). *CCNA Certification Study Guide*. Wiley. doi: 10.1002/9781119660286

Lantz, L., & Cawrey, D. (2020). *Mastering Blockchain*. O'Reilly Media, Inc.

Lawrence, T. (2017). *Blockchain for Dummies*. John Wiley and Sons.

Lepore, C., Ceria, M., Visconti, A., Rao, U. P., Shah, K. A., & Zanolini, L. (2020, 10). A Survey on Blockchain Consensus with a Performance Comparison of PoW, PoS and Pure PoS. *Mathematics*, *8*. doi: 10.3390/math8101782

Lin, C. T., Wu, S. L., & Lee, M. L. (2017). Cyber Attack and Defense on Industry Control Systems. *2017 IEEE Conference on Dependable and Secure Computing*, 524-526. doi: 10.1109/DESEC.2017.8073874

Liu, S. (2019). MAC Spoofing Attack Detection Based on Physical Layer Characteristics in Wireless Networks. *2019 IEEE International Conference on Computational Electromagnetics, ICCEM 2019 - Proceedings*, *2019-Janua*, 1-3. doi: 10.1109/COMPEM.2019.8779180

Liu, Y., Liu, J., Zhang, Z., & Yu, H. (2020). A Fair Selection Protocol for Committee-based Permissionless Blockchains. *Computers and Security*, *91*. doi: 10.1016/j.cose.2020.101718

Ljubuncic, I. (2015). *Problem-solving in High Performance Computing: A Situational Awareness Approach with Linux 1st Edition* . Morgan Kauffmann.

Ljubuncic, I., & Litterer, T. (2019). *Do No Harm.* Apress. doi: 10.1007/978-1-4842-4988-8_9

Machi, L. A., & McEvoy, B. T. (2016). The Literature Review: Six Steps to Success (3rd edition). *Academic Journal of Research and Scientific Publishing* |, *3*.

Mahmood, S., Mohsin, S. M., & Abrar, S. M. (2020). An Overview Network Security Issues of Data Link Layer.

Malcher, M., & Kuhn, D. (2019). *Pro Oracle Database 18c Administration: Manage and Safeguard Your Organization's Data*. Apress.

Marro, G. M. (2003). Attacks at the Data Link Layer.

McKelvey, F., & Driscoll, K. (2019). ARPANET and its Boundary Devices: Modems, IMPs, and the Inter-Structuralism of Infrastructures. *Internet Histories*, *3*, 31-50. Retrieved from `https://doi.org/10.1080/24701475.2018.1548138` doi: 10.1080/24701475.2018.1548138

Mehra, R., & Krishnan, K. V. (2018). Cloaking Malware with the Trusted Platform Module. *IEEE Transactions on Education*, *15*, 113-129. doi: 10.1016/0167-9236(94)00041-2

Moffett, N. (2020). *Creating a Framework for Dissertation Preparation* (N. L. Moffett, Ed.). IGI Global. doi: 10.4018/978-1-5225-9707-0

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* Retrieved from `www.bitcoin.org`

Newnham, D. (2016). *Female Innovators at Work*. Apress.

Nijim, M., Albataineh, H., Khan, M., & Rao, D. (2017). FastDetect: A Data Mining Engine for Predecting and Preventing DDoS attacks. *2017 IEEE International Symposium on Technologies for Homeland Security, HST 2017*, 1-5. doi: 10.1109/THS.2017.7943451

Novak, K., Valsecchi, P., Cartwright, H., Mauro, A., Brown, M., & Gavanda, M. (2019). *The*

Complete VMware vSphere Guide. Packt Publishing.

NSA. (2021). *National Centers of Academic Excellence in Cybersecurity.* Retrieved from https://www.nsa.gov/resources/students-educators/centers-academic-excellence/

Ornaghi, A., & Valleri, M. (2003). Man in the Middle Attacks.. Retrieved from http://www.blackhat.com/presentations/bh-usa-03/bh-usa-03-ornaghi-valleri.pdf

Parisi, A. (2020). *Securing Blockchain Networks like Ethereum and Hyperledger Fabric*. Packt Publishing.

Perlman, R. (1985). An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN. In (p. 44-53). ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=319056.319004 doi: 10.1145/319056.319004

Perlman, R. (1999). An Overview of PKI Trust Models. *IEEE Network*, *13*. doi: 10.1109/65.806987

Perlman, R. (2009). Folklore of Robust Network Routing. *Proceedings of the 2009 7th International Workshop on the Design of Reliable Communication Networks, DRCN 2009*, 21-28. doi: 10.1109/DRCN.2009.5340030

Pingle, B., Mairaj, A., & Javaid, A. Y. (2018, 10). Real-World Man-in-the-Middle (MITM) Attack Implementation Using Open Source Tools for Instructional Use. In (Vol. 2018-May, p. 192-197). IEEE Computer Society. doi: 10.1109/EIT.2018.8500082

Pujolle, G. (2020). *Software Networks*. Wiley ISTE.

Rahalkar, S. (2018). *Network Vulnerability Assessment*. Packt Publishing.

Rahalkar, S., & Jaswal, N. (2019). *The Complete Metasploit Guide*. Packt Publishing.

Rai, A., Barbhuiya, F. A., Sur, A., Biswas, S., Chakraborty, S., & Nandi, S. (2011). Exploit Detection Techniques for STP using Distributed IDS. *Proceedings of the 2011 World Congress on Information and Communication Technologies, WICT 2011*, 939-944. doi: 10.1109/WICT.2011.6141374

Ramezan, G., & Leung, C. (2018). A Blockchain-Based Contractual Routing Protocol for the Internet of Things Using Smart Contracts. *Wireless Communications and Mobile Computing*, *2018*, 1-14. doi: 10.1155/2018/4029591

Rietz, R., Cwalinski, R., König, H., & Brinner, A. (2018, 11). An SDN-Based Approach to Ward Off LAN Attacks. *Journal of Computer Networks and Communications*, *2018*. doi: 10.1155/2018/4127487

Roberts, L. G., & Wessler, B. D. (1970). Computer Network Development to Achieve Resource Sharing. In (Vol. 36, p. 543-549). Advanced Research Projects Agency. doi: 10.1145/

1476936.1477020

Rubin, L. H., & Shakamuri, H. K. (2014). *Method for Parsing Network Packets having Future Defined Tags* (Vol. 2). United States Patent.

Sak, B., & Ram, J. R. (2016). *Mastering Kali Linux Wireless Pentesting*.

Santos, O. (2020). *CCNP and CCIE Security Core SCOR 350-701 Official Cert Guide*. Cisco Press.

Schmidt, C. (2019). *Complete A+ Guide to IT Hardware and Software: A CompTIA A+ Core 1 (220-1001) and CompTIA A+ Core 2 (220-1002) Textbook, 8th Edition*. Pearson IT Certification.

Sgantzos, K., & Grigg, I. (2019, 8). Artificial Intelligence Implementations on the Blockchain. Use Cases and Future Applications. *Future Internet*, *11*. doi: 10.3390/fi11080170

Sharma, G., Pandey, N., Hussain, I., & Kathri, S. K. (n.d., 4). Design of Framework and Analysis of Internet of Things at Data Link Layer, volume = 2018-January, year = 2018,. In (p. 1-4). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/TEL-NET .2017.8343520

Shrivastava, S., Srivastav, N., & Arora, K. (2020). *Solutions Architect's Handbook*. Packt Publishing.

Singh, G. (2019). *Learn Kali Linux 2019*. Packt Publishing.

Singh, G. D., Michael, V., & Anandh, V. (2018). *CCNA Security 210-260 Certification Guide*. Packt Publishing, Limited.

Sinha, P., Jha, V. K., Rai, A. K., & Bhushan, B. (2018, 3). Security Vulnerabilities, Attacks and Countermeasures in Wireless Sensor Networks at Various Layers of OSI Reference Model: A Survey. In (Vol. 2018-January, p. 288-293). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/CSPC.2017.8305855

Siswanto, A., Syukur, A., Kadir, E. A., & Suratin. (2019). Network Traffic Monitoring and Analysis using Packet Sniffer. *Proceedings - 2019 International Conference on Advanced Communication Technologies and Networking, CommNet 2019*, 1-4. doi: 10.1109/COMMNET.2019.8742369

Soyinka, W. (2020). *Linux Administration: A Beginner's Guide* (8th ed.). McGraw-Hill.

Straub, D., Gefen, D., & Boudreau, M.-C. (2004). Quantitative, Positivist Research Methods in Information Systems Website. Retrieved from `http://dstraub.cis.gsu.edu:88/quant/default.asp`

Strawn, G. (2019). Blockchain. *IT Professional*, *21*, 91-92.

Thompson, G. (2019). Ethernet: From Office to Data Center to IoT. *Computer*, *52*, 106-109.

doi: 10.1109/mc.2019.2930099

Trabelsi, Z. (2012). Switch's CAM Table Poisoning Attack : Hands -on Lab Exercises for Network Security Education. , 113-120.

Tran, V.-H., & Bonaventure, O. (2019, 5). Beyond Socket Options: Making the Linux TCP Stack Truly Extensible. IEEE. doi: 10.23919/IFIPNetworking46909.2019.8999401

UcedaVelez, T., & Morana, M. M. (2015). *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. Wiley.

Varis, N. (2012). Anatomy of a Linux Bridge.. Retrieved from `https://aaltodoc.aalto.fi/handle/123456789/8945`

Vmware. (2019, 9). *Switch Types and Network Connectivity for NSX-T Workload Domains.* Retrieved from `https://docs.vmware.com/en/VMware-Validated-Design/5.1/sddc-architecture-and-design-for-vmware-nsxt-workload-domains/GUID-A7CF1DFE-9C2D-4483-8F68-49C76135E460.html`

Vugt, S. V. (2015). *Red Hat RHCSA/RHCE 7 Cert Guide: Red Hat Enterprise Linux 7 (EX200 and EX300)*. Pearson IT Certification.

Vyncke, E., Paggen, C., Bhandari, R., Carter, E., & Maudlin, H. (2008). *LAN Switch Security: What Hackers know about your Switches* (1st ed.). Cisco Press.

Wojdak, W. (2003). Rapid Spanning Tree Protocol: A New Solution from an Old Technology. *Compact PCI Systems*.

Yeung, K., Fung, D., & Wong, K. (2008). Tools for Attacking Layer 2 Network Infrastructure..

Yeung, K. H., Yan, F., & Leung, T. C. (2006). Improving Network Infrastructure Security by Partitioning Networks Running Spanning Tree Protocol. *International Conference on Internet Surveillance and Protection, ICISP'06*, *00*, 0-3. doi: 10.1109/ICISP.2006.13

Younes, O. S. (2017). Securing ARP and DHCP for Mitigating Link Layer Attacks. *Sadhana - Academy Proceedings in Engineering Sciences*, *42*, 2041-2053. doi: 10.1007/s12046-017-0749-y

Zaharaddeen, A. B. A. B., M, & Shehu, S. (2020, 2). Design of a Virtual Area Network for National Open University Sokoto Study Center. *International Journal of Wireless and Microwave Technologies*, *10*. doi: 10.5815/ijwmt.2020.01.03

Zhang, R., Xue, R., & Liu, L. (2019). Security and Privacy on Blockchain. *ACM Computing Surveys*, *52*. doi: 10.1145/3316481

Zimmerman, J., & Spurgeon, C. (2014). *Ethernet: The Definitive Guide, 2nd Edition*. O'Reilly Media, Inc.

Zinin, A. (2001). *Cisco IP Routing: Packet Forwarding and Intra-domain Routing Protocols*.

Addison-Wesley Professional.

Zubairi, J. A., & Mahboob, A. (2011). *Cyber Security Standards, Practices and Industrial Applications: Systems and Methodologies*. doi: 10.4018/978-1-60960-851-4

Appendix: Definition of Terms

Data Encapsulation - a Layer 2 MAC sub-layer protocol service that organizes data for transport. The Data Link layer service encapsulates data into an Ethernet frame

Frame Synchronization - a Layer 2 MAC sub-layer protocol service that is used to validate Ethernet frame data.

Flow Control - a Layer 2 MAC sub-layer protocol service that controls the orderly flow of frames to avoid congestion, overwhelm and collision

Access Control - a system where authentication is used to validate a user or a process

Collision - occurs when two or more frames are transmitted at once which requires frame re-transmission leading to a lag and denial of service

Convergence - in STP protocol topology management, all traffic is stopped while a new Root is verified by all LAN switches

Spanning Tree Protocol (STP) - Spanning Tree Protocol is a data link layer protocol and algorithm created to automate the management of Ethernet LANs

802.1d - the IEEE standard based on the STP protocol algorithm

802.1w - he IEEE standard based on the rSTP protocol algorithm for metropolitan networks

Ethernet LAN Topology - -an organized structure of Ethernet LAN switches based on efficient message transmission

Hyperledger Fabric - Hyperledger Fabric is a software-based corporate solution created by IBM based on the Linux Foundation's open-source Hyperledger blockchain framework

Switch - a device that works on an Ethernet LAN in the Data Link layer to connect networked devices to ports that can be managed. The words switch and bridge are used interchangeably in this study. **MAC** - Media Access Control - a protocol at the Data Link layer that provides a unique identifier for networked devices.

Frame - the protocol data unit for data transmitted on the data link layer using MAC

addresses

Packet - the protocol data unit for data transmitted in the Network layer using the IP protocol

CIA triad - an information security term to represent the principles of confidentiality, integrity and availability. It is often represented as three parts of a triangle.

Confidentiality - data that is private

Integrity - data that remains unchanged from unauthorized entities

Availability - data or resources that remain usable for intended users

Authenticity - validation is used to prove that users are who they say they are

MITM - Man in the middle - an attack where a malicious user has infiltrated a network to intercept communications between or among legitimate devices.

DoS - Denial of Service - when resource or data availability is compromised from an attack

DDoS - Distributed Denial of Service - when several resources are compromised during an attack

Bridge - A Layer 2 network device used to transport Layer 2 traffic or Ethernet between two LANs. The words bridge and switch are used interchangeably in the study.

STP - Spanning Tree Protocol - a protocol in the data link layer used in a LAN to prevent loops and expand resource efficiency

RSTP - Rapid Spanning Tree Protocol - an update to the STP protocol in 2001 that increased computational efficiency

STP Root-Takeover, Root-Takeover - the Root-Takeover attack is when an attacker uses a rogue device to take over the function of a legitimate Root in the STP/RSTP protocol

BPDU - Bridge Protocol Data Unit - these are messages sent by switches using the STP/RSTP protocol to maintain a self-automated network topology

Yersinia - an attack tool used for various data link layer and MITM attacks

Ettercap - an attack tool used to perform data link layer attacks.

Blockchain - a technology underlying the original Bitcoin electronic cash system that keeps record of transactions.

Decentralized - a feature of bitcoin's blockchain. It features no central authority or trusted authority to provide a service

Immutable - cannot be changed or altered

Centralized - an entity governed by a central authority

Trustless - an entity that is not reliant on a trusted authority and is maintained with no

trusted entities

Permissionless - an entity that does not rely on a trusted or central authority for function

Trust based - a system that requires a trusted authority to function

PKI - public key infrastructure - an asymmetric algorithm that validates an item using a random number generated private key and public key. The private key validates a transaction with a digital signature

Digital signature - a validation mechanism used by PKI where a transaction is signed with a private key

Encryption - when an item is altered using a pattern created by a cryptographic algorithm.

Bitcoin - the electronic cash system created by Satoshi Nakamoto

Bitcoin Core- the electronic cash system to include the blockchain created by Satoshi Nakamoto

OSI reference model - an abstract layering diagram to describe the various services performed within the network infrastructure

Data link layer (layer 2) - the second layer in the OSI reference model provides services including MAC addressing and encapsulates data into a protocol data unit called a frame

Root Bridge - the bridge in STP/RSTP elected as the head of the Ethernet LANs network topology based on lowest ID

Designated Bridge - a switch that is has the Root Path Cost (RPC) to the Root

Port - a channel on a switch that connects a network device with its own MAC address

Root Port - the port on the Designated Bridge that is closest to the Root

Designated Port - the port on the Root that is closest to the Designated Bridge

Topology change notification (TCN) - a type of BDPU message sent from the root bridge to notify all the switches of a topology change, typically, a change in the root bridge

Topology change - when a new root bridge is elected in STP/RSTP

Metasploit - a package of tools that can be used to perform various social engineering, network and system attacks

Kali - an operating system that can be used to perform attacks with various social engineering network and system attack tools.

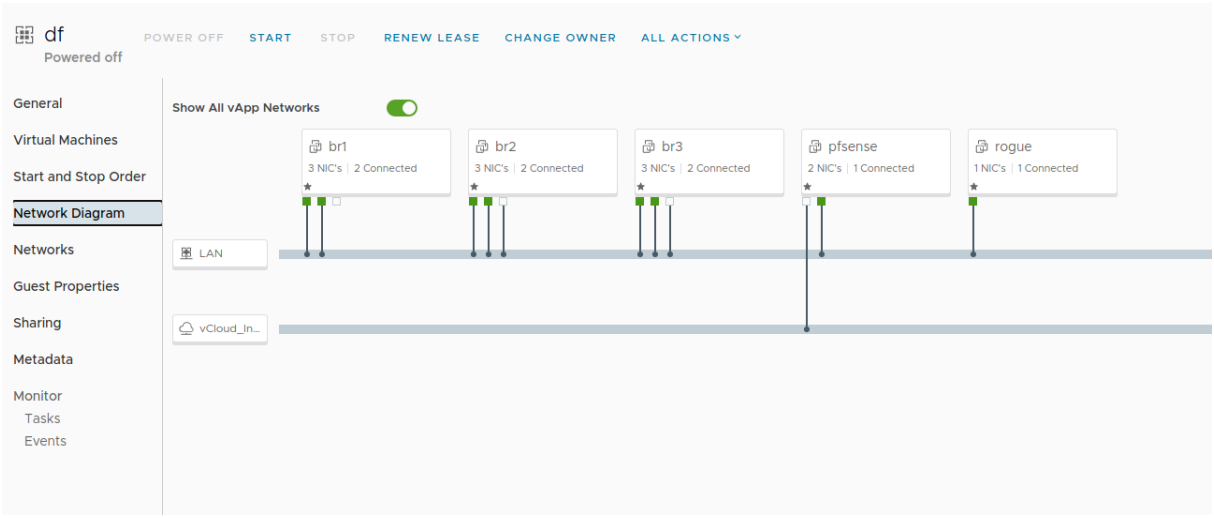
Smart Contract A Smart Contract is software-based business logic. It allows practical uses for a blockchain, such as supply chain management, identity management, and contracts by adding a software program to the blockchain

Chaincode Hyperledger Fabric's version of the Smart Contract

Appendix: Technical Documentation

Ethernet LAN Configuration

The hypervisor used to host the Ethernet LAN is called VMWare Vcloud on the current version 10.2. It provides the resources required to conduct the experiment with integrity and observe results due to its resource scalability and availability. Vmware provides the resources to host the Ethernet LAN. Switches are devices that pass Ethernet traffic called frames in the virtualized data link layer. Some of the Ethernet traffic is specific to the STP protocol for switch topology management. The requirement for the switches in the control group is that they can be enabled to pass STP-specific frames. The Debian Linux kernel comes with bridge modules that can provide switch services enabled with STP. Although the nomenclature can be confusing, the words bridge and switch can be used interchangeably. When configuring the Debian Linux kernel for bridge mode, it is being configured for switch functionality. Hence, it can be enabled with STP and pass Ethernet frames. The Debian Linux distribution used for the Ethernet LAN is Ubuntu 20.04 LTS Desktop edition, the latest version at testing. Ubuntu is a widespread Debian Linux distribution known for its ease of use. Three Ubuntu 20.04 Desktop virtual machines have been configured as LEBs to create the Ethernet LAN. Each virtual machine is connected with two network interfaces to provide optimal bridge functionality. The first interface is configured with a LAN and gateway IP address, and the second interface is configured to create a connection between the physical device and the bridge.



Linux Ethernet Bridge Configuration

Network Utilities: Network Manager

NetworkManager is a popular network management tool that can create a persistent bridge configuration on the LEB. With network persistence, the bridge configuration is already written into the system's network configuration files, so it does not have to be reconfigured on restart. The virtual machine's NICs are two network interfaces for each bridge. Each NIC had a MAC address to form the bridge ID nomenclature to provide a unique identifier for each network device in the LAN. One interface was used to create an outgoing LAN interface, and the other interface was configured as the bridge interface for LAN traffic.

```
leb1@leb1:~$ nmcli con add type bridge ifname leb1 ip4 192.168.11.2/24 gw4 192.
168.11.1
Connection 'bridge-leb1-1' (cee28c72-f5fc-4f62-98aa-7e5a6447a5f2) successfully
added.
leb1@leb1:~$ nmcli con add type bridge-slave ifname ens160 master bridge-leb1-1
Connection 'bridge-slave-ens160-1' (21f706d6-e0ee-4216-ba21-656439d1d76d) succe
ssfully added.
leb1@leb1:~$
```

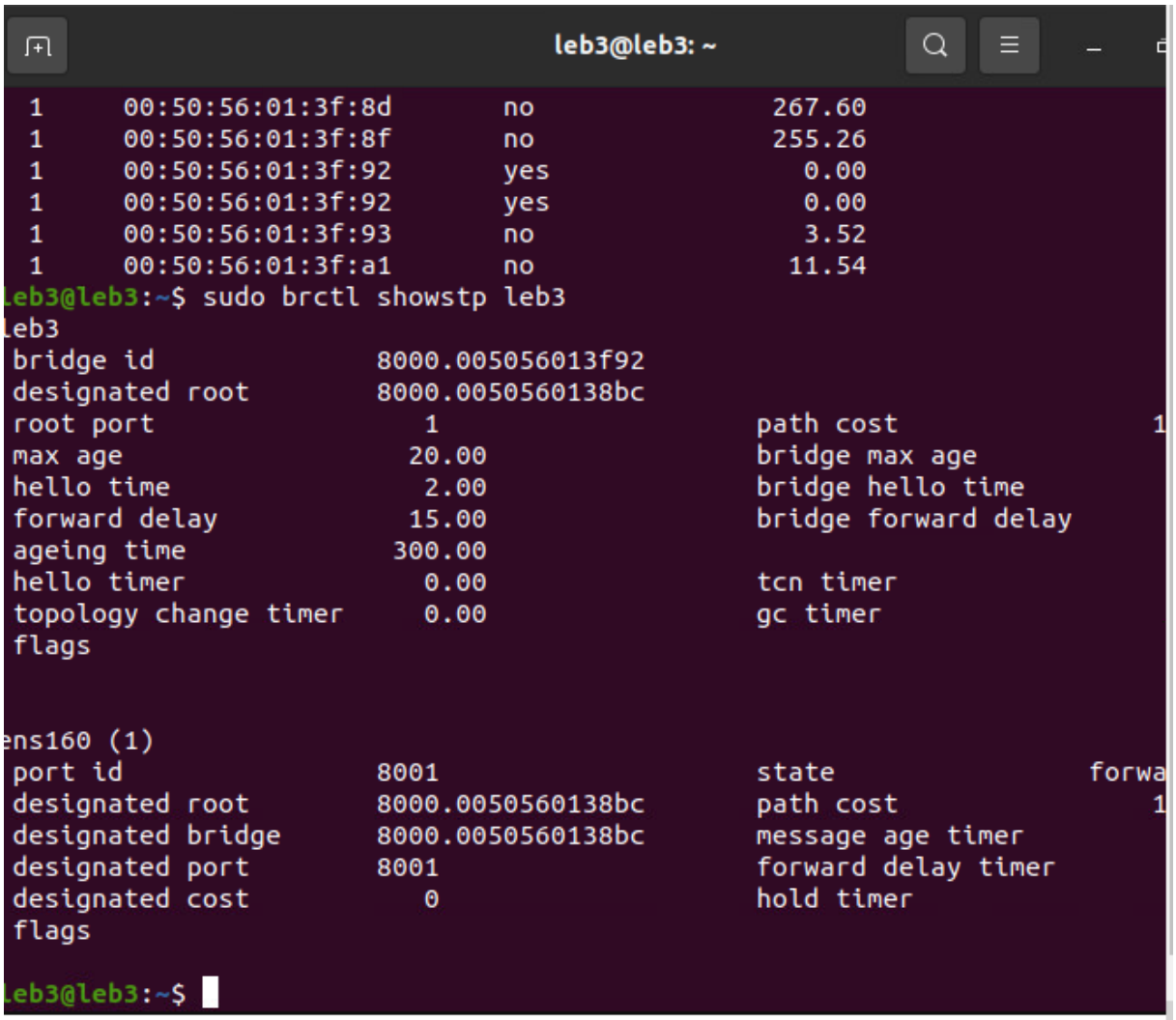
The resulting configuration file is written in the Network Manager

```
leb2@leb2:/etc/NetworkManager/system-connections$ sudo cat bridge-leb2.nmconnec
tion
[sudo] password for leb2:
[connection]
id=bridge-leb2
uuid=2eed723e-9afb-4507-8e98-632f3273360e
type=bridge
interface-name=leb2
permissions=
autoconnect=yes

[bridge]

[ipv4]
address1=192.168.22.2/24,192.168.22.1
dns-search=
method=manual

[ipv6]
addr-gen-mode=stable-privacy
dns-search=
method=auto

[proxy]
leb2@leb2:/etc/NetworkManager/system-connections$
```

Network Utilities: Bridge Utils Each Ubuntu VM was provided an installation of the *bridge-utils* tool. *Bridge-utils* is a utility package that allows the user to run simple commands to provide some configuration and bridge information. An essential part of setting up the bridge for this study is to add STP protocol functionality. As it is not set up to use STP by default, it can be turned on by the *bridge-utils* tool. The *bridge-utils* tool provides details about the Ethernet LAN topology based on the STP protocol. The details provided match those sent in the BPDU frame as part of the decision-making in the Root election, and this is how the bridges are monitored to see what it recognizes as the *designated root*. The *designated bridge* is based on the NIC mac address used for Ethernet bridge traffic and a unique identifier for the bridge. The *bridge-utils* application provides this information.

```
leb1@leb1:~$ brctl showstp leb1
leb1
 bridge id              8000.005056013fa0
 designated root        8000.0050560138bc
 root port                 1                    path cost                100
 max age                  20.00                 bridge max age           20.00
 hello time                2.00                 bridge hello time         2.00
 forward delay            15.00                 bridge forward delay     15.00
 ageing time             300.00
 hello timer               0.00                 tcn timer                 0.00
 topology change timer     0.00                 gc timer                199.02
 flags


ens160 (1)
 port id                8001                   state                forwarding
 designated root        8000.0050560138bc      path cost                   100
 designated bridge      8000.0050560138bc      message age timer         18.32
 designated port        8001                   forward delay timer        0.00
 designated cost           0                   hold timer                 0.00
 flags

leb1@leb1:~$ brctl showmacs leb1
```

```
leb2@leb2:~$ brctl showstp leb2
leb2
 bridge id              8000.005056013f8f
 designated root        8000.0050560138bc
 root port                 1                    path cost                100
 max age                  20.00                 bridge max age           20.00
 hello time                2.00                 bridge hello time         2.00
 forward delay            15.00                 bridge forward delay     15.00
 ageing time             300.00
 hello timer               0.00                 tcn timer                 0.00
 topology change timer     0.00                 gc timer                 82.12
 flags


ens160 (1)
 port id                8001                   state                forwarding
 designated root        8000.0050560138bc      path cost                   100
 designated bridge      8000.0050560138bc      message age timer         18.75
 designated port        8001                   forward delay timer        0.00
 designated cost           0                   hold timer                 0.00
 flags

leb2@leb2:~$
```



The *bridge id* designation shows the id of the bridge which is 8000.005056013fa0. The *bridge ID* is based on the MAC address of the NIC that is used for Ethernet bridge traffic and it is also a unique identifier for the bridge. Below the *bridge id*, the *designated root* is also a key element in the study. It is the elected Root for the Ethernet LAN topology. Right now, it is 8000.0050560138bc which is different from the bridge id which is 8000.005056013fa0. This means that the bridge, leb1, is a bridge in the Ethernet LAN but is not the elected Root. The details provided show the *path_cost* and *hello_time* which are key elements to Root election in the STP algorithm. When there is little difference among these key elements, the Root election is based on setting the bridge ID in numerical order.

Yersinia: Code used for the STP Root-Takeover attack

Yersinia is an open-source tool that can work with attacks on various network protocols. Although this study's focus is its ability to perform attacks based on the STP protocol, it

has robust functionality that attacks other network protocols. For this study's purpose, it also can perform attacks in varying types of Ethernet LANs. These Ethernet LANs can be made up of popular vendors, virtual switches, or open-source software bridges such as the Linux Ethernet Bridge. The tool's ability to pick up information from varying frame formats makes it a powerful tool. At the time of the study, the URL to access the latest version of Yersinia and its source fragments is *https://github.com/tomac/yersinia/tree/master/src*. This study's objective is to examine the attacks based on the STP protocol within the folder structure. These files are called *xstp.c* and *xstp.h*. At the time of the study, these files have not been updated in four years. Curiously, no changes were required to these files. They continue to maintain successful attacks on network devices with updated security patches such as those on the Ethernet bridges in this study. The attack vector device listens for STP Ethernet frames on the network at the beginning of the STP Root-Takeover attack. This stage is the passive information gathering MITM attack described in the introduction chapter. The goal is to gather information about the current Root ID by collecting its information from the BPDU messages passed among the LAN bridges. Due to the nature of the experimental environment, the broadcasted configuration frames were collected by all of the bridges of the LAN, showing the forged Root ID within a second because the election through brute force did not use a topology change notification due to the nature of the trust-based STP protocol implemented on the LAN. In the *xstp.c*file, the function $xstp\_learn\_packet$ searches all of the interfaces for packets and parses them for STP information. This function filters for packets that hold STP-based Ethernet frames called BPDUs. More specifically, it looks for $BPDU\_conf$ frames that match those for STP and RSTP protocols by parsing the packets for indicators for the 802.3 Ethernet frames used for the STP topology. It structures the data it needs for a forged frame, such as the *source mac address, destination mac address, bridge id, root id, hello time, forward delay, max-age,* and *root path cost*.

```
1046  xstp_learn_packet(struct attacks *attacks, char *iface, u_int8_t *stop, void *data, struct pcap_pkthdr *header )
1047  {
1048      struct stp_data *stp_data = (struct stp_data *)data;
1049      struct interface_data *iface_data;
1050      struct libnet_802_3_hdr *ether;
1051  #ifdef LBL_ALIGN
1052      u_int16_t aux_short;
1053      u_int32_t aux_long;
1054  #endif
1055      u_int8_t *packet, *stp_conf;
1056      int8_t got_bpdu_conf = 0;
1057      dlist_t *p;
1058
1059
1060      if (iface)
1061      {
1062         p = dlist_search(attacks->used_ints->list, attacks->used_ints->cmp, iface);
1063         if (!p)
1064            return -1;
1065
1066         iface_data = (struct interface_data *) dlist_data(p);
1067      }
1068      else
1069         iface_data = NULL;
1070
1071      packet = (u_int8_t *)calloc( 1, SNAPLEN );
1072
1073      if ( ! packet )
1074          return -1;
1075
1076      while (!got_bpdu_conf && ! (*stop) )
1077      {
1078          interfaces_get_packet(attacks->used_ints, iface_data, stop, header, packet, PROTO_STP, NO_TIMEOUT);
```

This first part of the function creates the structure to collect the required data for the attack using a BPDU structure defined in line 1048: $structstp\_data$ *stp_data = *(struct stp_data *)data*. To pick up BPDU information, the $interfaces\_get\_packet$ function is called by the $xstp\_learn\_packet$ function. This function is used for passively sniffing information by gathering packets on the Ethernet LAN. The packet is parsed and identified as a BPDU frame. The contents of information such as the source mac address and destination mac address are copied from the packet to the stp_data structure for future use in the STP Root-Takeover attack.

```
1085
1086          stp_conf = (packet + LIBNET_802_3_H + LIBNET_802_2_H);
1087
1088          switch (*(stp_conf+3))
1089          {
1090              case BPDU_CONF_STP:
1091              case BPDU_CONF_RSTP:
1092
1093                  got_bpdu_conf = 1;
1094
1095                  ether = (struct libnet_802_3_hdr *) (packet);
1096
1097                  memcpy((void *)stp_data->mac_source, (void *)ether-> 802_3_shost, ETHER_ADDR_LEN);
1098                  memcpy((void *)stp_data->mac_dest, (void *)ether->_802_3_dhost, ETHER_ADDR_LEN);
1099
1100  #ifdef LBL_ALIGN
1101                  memcpy((void *)&aux_short,stp_conf,2);
1102                  stp_data->id = ntohs(aux_short);
1103  #else
1104                  stp_data->id = ntohs(*(u_int16_t *)stp_conf);
1105  #endif
1106
1107                  stp_data->version = *((u_int8_t *)stp_conf+2);
1108
1109                  stp_data->bpdu_type = *((u_int8_t *)stp_conf+3);
1110
1111                  stp_data->flags = *((u_int8_t *)stp_conf+4);
1112
1113                      memcpy((void *)stp_data->root_id, (void *)(stp_conf+5), 8);
1114
1115                      memcpy((void *)stp_data->bridge_id, (void *)(stp_conf+17), 8);
1116
```

The code in lines 1097 and 1098 shows how the source mac address (mac_source) is set in the stp_data structure for the forged frame. The code in lines 1113 and 1115 shows the same method used to copy the *bridge ID* ($bridge\_id$) and *root ID* ($root\_id$) into the stp_data structure using the *memcpy* function. Once the $xstp\_learn\_packet$ function completes the preliminary

work, it is requested by the $xstp\_th\_nondos\_role$ function defined in line 648 of the xstp.c file. This function requests the *source mac address, destination mac address, bridge id, root id, hello time, forward delay, max-age*, and *root path cost* of the current designated Root ID. This information is saved to use in the forged frame created to implement an attack to claim the Root role. All of the information is inserted into the forged frame except for the Root ID. The Root role functionality is claimed by providing a Root ID with all of the same information except for the Root ID's bottom-up numerical order. The Root ID's numerical order is only used to decide on a new Root Bridge's election when the other information matches. The decremented bridge ID is created using the function $xstp\_decrement\_bridgeid$, which is implemented in line 688.

```
648  void xstp_th_nondos_role( void *arg )
649  {
650      struct attacks *attacks = (struct attacks *)arg ;
651      struct stp_data *stp_data;
652      struct pcap_pkthdr header;
653      struct timeval now;
654      u_int8_t flags_tmp;
655      u_int8_t *packet, *stp_conf;
656      sigset_t mask;
657
658      pthread_mutex_lock(&attacks->attack_th.finished);
659
660      pthread_detach(pthread_self());
661
662      sigfillset(&mask);
663
664      if (pthread_sigmask(SIG_BLOCK, &mask, NULL))
665      {
666         thread_error("xstp_th_nondos_role pthread_sigmask()",errno);
667         xstp_th_nondos_role_exit(attacks);
668      }
669
670      gettimeofday(&now,NULL);
671
672      header.ts.tv_sec = now.tv_sec;
673      header.ts.tv_usec = now.tv_usec;
674
675      stp_data = attacks->data;
676
677      if (xstp_learn_packet(attacks, NULL, &attacks->attack_th.stop,stp_data,&header) < 0)
678          xstp_th_nondos_role_exit(attacks);
679
680      xstp_decrement_bridgeid(stp_data);
```

The key to the election of a new Root in an STP topology is that if the path cost elements are the same, the bridge with the lower numerical bridge ID will be elected as Root. The use of this function provides a BPDU frame that is set up to create a topology change. After this function has been used, the fabricated BPDU frame with a lower Bridge ID is sent out into the Ethernet LAN to incite a topology change. This topology change will use the fake Bridge ID in the frame and provide the rogue device control of the Ethernet LAN. Now the attacker gains the ability to gather more information, route traffic, and eventually gain control of layer three devices such as routers. Considering the information provided on how the attack is performed on an Ethernet LAN, this study's first question would be to figure out Yersinia's ability to attack the control group's Ethernet LAN. Can an old tool that has not been updated in four years still perform the STP Root-Takeover attack on currently updated Ethernet LAN devices? This experiment is conducted as Step 1: The validation of the STP Root-Takeover attack.

STP DApp Code Analysis

The research solution is the STP MAC validation software tool used to prevent an STP Root-Takeover attack performed by Yersinia in the Ethernet LAN defined by the pre-experimental Control Group. The first part of the experiment demonstrates the STP Root-Takeover attack using the control group's attack vector variable and evaluates its effectiveness on the Ethernet LAN variable. The second part of the study proposes a trustless solution that can be used to prevent the experimental attack on the same Ethernet LAN and attack vector variables in the control group. In this light, the research solution was administered on a virtual machine created with the Ubuntu 20.04 LTS Desktop operating system and configured as a bridge like the Ethernet LAN's other bridges. The STP DApp and attack are run from the virtual machine simultaneously to prove effective detection of the STP Root-Takeover attack administered by the Yersinia attack tool in the virtual machine running the Kali 2004.3 LTS operating system. It is made up of two parts to create a whole system. The first part is a kernel modification called *stpverify* added to *dev.c*. in the */net/core* folder of the Ubuntu 5.8.1 kernel. The second part of the blockchain framework in the Nodejs runtime environment. Both parts run on a single operating system on a virtual machine. The source code is made up of several open-source frameworks and is available in Github at *https://github.com/milapaul/stpmacverify*. Validation begins with the kernel modification in dev.c to track the established Root ID and new bridge IDs added to the Ethernet LAN. A blockchain will hold hashed values of bridge IDs that have been validated in the Ethernet LAN. The *stpverify* mod written in C will output the bridge ID to a *proc* file called *root_out*. Javascript interface in the NodeJS runtime environment. NodeJS is the runtime environment for the blockchain framework by Hyperledger Fabric. The framework includes a UI for user interaction with the blockchain. The interaction with the blockchain is managed by Smart Contracts called Chaincode, and they manage a query or an addition to the blockchain. The STP DApp begins its functionality with the *stpverify* mod and then connects to the blockchain framework in the NodeJS runtime environment for validation. The results are sent back to the *proc* file *root_in*. If the discovered bridge ID is established in the Ethernet LAN is validated by an entry in the blockchain, the network traffic is allowed to continue in the Ethernet LAN. If the bridge ID is new to the Ethernet LAN, its network traffic will be dropped until a network administrator can validate and add it to the blockchain. The kernel modification written as part of the STP DApp is called *stpverify* and is located in the */net* folder of the kernel directory.

main.c

The kernel modification is registered and initiated in the *main.c* file. The definition shown notifies the kernel that this modification will be used. The code is a modification of the open-source traffic analyzer module called *trafficsqueezer*.

Credits for TrafficSqueezer

```
/*
TRAFFICSQUEEZER provides dual licenses, designed to meet the usage and distribution requirements of different types of users.

GPLv2 License: Copyright (C) (2006-2015) Kiran Kankipati (kiran.kankipati@gmail.com) All Rights Reserved.

TrafficSqueezer is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License Version 2, and not
any other version, as published by the Free Software Foundation. TrafficSqueezer is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
more details.You should have received a copy of the GNU General Public License along with TrafficSqueezer; see the file COPYING. If not, write
to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

BSD License (2-clause license): Copyright (2006-2015) Kiran Kankipati. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
   1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
   2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY KIRAN KANKIPATI ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KIRAN KANKIPATI OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing
official policies, either expressed or implied, of Kiran Kankipati.

* This license is applicable exclusive to the TrafficSqueezer components. TrafficSqueezer may bundle or include other third-party open-source
components. Kindly refer these component README and COPYRIGHT/LICENSE documents, released by its respective authors and project/module owners.
** For more details about Third-party components, you can also kindly refer TrafficSqueezer project website About page.
*/
#include <linux/types.h>
#include <linux/mm.h>
#include <linux/skbuff.h>
#include <linux/ip.h>
#include <linux/in.h>
#include <linux/in6.h>
#include <linux/icmp.h>
#include <linux/netdevice.h>
#include <net/sock.h>
#include <net/ip.h>
#include <net/tcp.h>
#include <net/udp.h>
#include <net/icmp.h>
#include <linux/tcp.h>
#include <linux/udp.h>
#include <linux/netfilter_ipv4.h>
#include <net/checksum.h>
#include <linux/route.h>
#include <net/route.h>
#include <net/xfrm.h>
#include <net/protocol.h>
#include <linux/ctype.h>
#include <linux/gfp.h>

#include <net/stpverify/core.h> //include stpverify
#include <net/stpverify/proc.h> //create stpverify file system
#include <net/stpverify/init.h> //include stpverify_init() to initialize proc files
```

If *stpverify* is defined in the program, required initialization is found in these include files. *#ifdef* is used to define the *stpverify* modification in *main.c*. In *main.c* it runs its initialization.

```
#ifdef CONFIG_STPVERIFY //additional functionality added if stpverify is defined
#include <net/stpverify/core.h> // stpverify is defined use this core.h function stpverify_init()
#include <net/stpverify/init.h> // stpverify is defined use this init.h function stpverify_init()
#endif
```

init.h

Variables used in *stpverify* are initialized in *init.h*. The use of the *stpverify* module can be verified in the kernel syslog.

```
void stpverify_init()   //register stpverify as kernel mod
{       int i;
        stpverify_ctr=0;
        root_out_buf_len=0;
        root_in_buf_len=0;

        stpverify_init();
        printk("@ stpverify_init() - end\n");
}
```

proc.h

Memory-based files used in *stpverify* are created in *proc.h*. The files in the *stpverify* module can be accessed in the kernel */proc* folder.

```
int stpverify_init_proc(void)   //initialize ID mem buffer in proc.h for read access
{       root_out = proc_create( "root_out", 0444, NULL, &root_out_fops); //proc_create allows read/write access
        if(root_out==NULL) {    printk(KERN_ALERT "Error: Could not initialize %s\n", "root_out"); return FALSE; }

        root_in = proc_create( "root_in", 0666, NULL, &root_in_fops); //initialize root_in  for read access
        if(root_in==NULL) {     printk(KERN_ALERT "Error: Could not initialize %s\n", "root_in"); return FALSE; }

        memset(block_root_bridge, 0x00, 6);     //initialize to zero
}
```

dev.c

The core logic for the *stpverify* module is defined inside the *__netif_receive_skb_core* function. This is a function in *dev.c* that receives Layer 2 traffic and processes it in the Layer 7 kernel. It is here where the data is organized into the structure of an Ethernet frame and parsed for its Root ID. The list of Root IDs is created in the *root_out* file. The root_ID is also checked for a match from the root_in file which holds the blockchain's invalidated root_IDs. The *goto drop* statement calls the function that will ultimately drop the frame from the network traffic ingress.

```
        __be16 type;

        net_timestamp_check(!netdev_tstamp_prequeue, skb);

        trace_netif_receive_skb(skb);

#ifdef CONFIG_STPVERIFY         //when network traffic is received, stpverify mod will check for BPDU frames
                                //and drop those that match the root_in buffer value

           int is_stp=0;        //variable check for BPDU frame
           struct ethhdr *eth;  //struct for ethernet frame in ether.h
           eth = (struct ethhdr*)skb_mac_header(skb);   //typecast header to ethernet type struct

           printk("src mac %pM, dst mac %pM\n", eth->h_source, eth->h_dest);
           printk("%02x%02x%02x%02x%02x%02x - %02x%02x%02x%02x%02x%02x \n", //print frame header src&dst to kernel log
           //write src and dest to mem buffer in ethernet frame struct
           skb->data[0], skb->data[1], skb->data[2], skb->data[3], skb->data[4], skb->data[5],
           eth->h_dest[0], eth->h_dest[1], eth->h_dest[2], eth->h_dest[3], eth->h_dest[4], eth->h_dest[5]);

                {
                        if(eth->h_dest[0]==0x01 && eth->h_dest[1]==0x80 && eth->h_dest[2]==0xc2 && //if dest is broadcast BPDU type, set is_stp=1
                        eth->h_dest[3]==0x00 && eth->h_dest[4]==0x00 && eth->h_dest[5]==0x00)
                        {
                                is_stp=1;
                                printk("STP Frame found \n");  //kernel log shows BPDU found
                        }


                        if(is_stp==1)   //if frame is BPDU
                        {
                                spin_lock(&root_out_buf_lock); //lock thread to write to root_out
                                if(root_out_buf_len<=(STPVERIFY_BUF_MAX_SIZE - STPVERIFY_BUF_RESERVE_SIZE)) //if space available in memory buffer
                                        {
                                                sprintf(root_out_buf, "%s%i,%02x%02x%02x%02x%02x%02x\n", root_out_buf, stpverify_ctr,    //write counter BPDU root ID to root_out
                                                skb->data[10], skb->data[11], skb->data[12], skb->data[13], skb->data[14], skb->data[15]); //root ID offset 10-15

                                                root_out_buf_len = strlen(root_out_buf); //track length of root_out memory buffer
                                                spin_unlock(&root_out_buf_lock);        //open thread after writing to root_out
                                                stpverify_ctr++;        //add counter ID after each BPDU found

                                                int drop_frm=0; //initialize dropped frames to 0
                                                spin_lock(&root_in_buf_lock);   //lock CPU for reading root_in
                                                if(!memcmp(block_root_bridge, (unsigned char *)(skb->data+10), 6))  //if ID matches root_in buffer (typecast ID to string)
                                                        {
                                                                drop_frm=1; //set variable: drop this frame
                                                        }
                                                spin_unlock(&root_in_buf_lock); //unlock CPU for reading root_in

                                                if(drop_frm==1) //if frame set to be dropped
                                                {
                                                        printk("STP Verify: dropping stp frame\n"); //kernel log shows frame drop
                                                        goto drop;      //NET_INGRESS drop function reads skb data and sends to NET_RX_DROP
                                                }
                                        }

                        }
        }
#endif
```

Hyperledger Fabric Configuration

The procedure for the initial setup for the environment is on https://hyperledger-fabric.readthedocs.io/en/release-2.2/. The latest release at the time of the study is 2.2. The environment setup includes installing the latest *git, curl, python, go, nodejs, docker-ce, docker-ce-clip*, and *docker-compose* to set up the runtime environment and nodes. The latest hyperledger fabric framework's base framework is to be cloned into a local repository from https://github.com/hyperledger/fabric-samples.git. Next, the latest docker virtual machine images for the nodes were downloaded from the latest production release in Github. Lastly, the hyperledger-specific binaries for the network setup are downloaded and placed in a /bin subdirectory. Regardless of the project-specific nomenclature added for some personalization, the framework's main functionality is available on a template. For the study, the base code was renamed to personalize the open-source project. Further, open source node-modules were downloaded for running the application, UI, and Root ID input script in the blockchain framework.

The creation of the blockchain is broken down into the following stages:

- artifact creation
- node creation
- channel creation

- deploy Chaincode

- initialize application

Artifacts are created to initialize the creation of the blockchain framework in Hyperledger Fabric. These artifacts are the access control configurations for rights to the blockchain and the certificates used to implement those rights. The initial block in the blockchain is also created in this stage. After the artifacts are created, the latest Hyperledger Fabric virtual machine images are pulled from Github and installed as virtual machines to create the network's nodes. For this experiment, these nodes emulate distributed servers in a blockchain network. The base network requires a peer server, CA server, CouchDB server for each organization, and an orderer. . A network connection is established through creating a channel. The channel manages the endpoints to each node through SHA256 token authentication. Next, the channel is connections to the nodes are initialized to manage access to the Chaincode. Lastly, the Chaincode is deployed using CA certs to manage authentication of the Chaincode to the peer nodes to manage access control to the blockchain. The last part of setting up the blockchain network is generating a network connection between the NodeJS application and the peer organizations through CA certificates. After this, the web application is initialized through the JavaScript front end. It manages access control and transactions from the NodeJS application to the UI, Chaincode, and blockchain. The Hyperledger Fabric framework provides a small-scale and manageable environment allowing the researcher to observe data and analyze the solution's effectiveness. It provides the framework for a secure validation mechanism that provides immutability for its transactions. Two organizations are created to provide access to the blockchain. The first organization, Org1, is delegated as admin and can validate and add a bridge ID to the blockchain in read-write mode. The second organization can log in and run searches of the established Root IDs in the blockchain in read-only mode. This search can also be evoked manually in the browser UI for user-friendly interaction with the blockchain ledger. The Org1 user can also log into the UI and add an entry to the blockchain. The UI written in JavaScript provides user login access to the blockchain. These users are registered with the organizations based on their access level. Encryption is established at all endpoints in the framework through SHA256 key/pairs and validated through the CA nodes.

The Hyperledger Fabric nodes used in the blockchain network include:

- One Orderer Node

- Two Peer Nodes

- Two CA Servers

- Two Database Nodes

The Blockchain Framework

The blockchain framework is an ecosystem that contains four parts. The first part is the interface between *stpverify* and the blockchain framework. The blockchain framework is housed in a Nodejs runtime environment that runs JavaScript. Therefore, the interface is written in JavaScript. The second part is the main NodeJS App that handles the front-end functionality. It can be seen as the control center or the brains of the Hyperledger Fabric framework. It is the first step to interaction with the blockchain, hence an intermediary that forms the front-end of a blockchain web application. Functions that make up the App set up the network connections with the various nodes and run functions that will search and add transactions to the blockchain ledger. It also handles users and tokens for authentication for access control among the nodes. Lastly, it connects to the user interface (UI) written in a JavaScript web interface framework called React. The UI is part of the front-end program that connects to the back-end blockchain ledger using the NodeJS App as an intermediary. In the construct of an average web application, this scenario is akin to a web interface that connects to a database such as a login page. A web application usually includes a UI in a browser as part of the front-end program. Entries submitted in the UI are sent to a database where queries are used to manipulate or process the data.

The third part of the blockchain framework is the Smart Contract. The Chaincode performs the programming that manages queries and modifications to the blockchain ledger. The last part of the framework is the blockchain ledger. The blockchain ledger comprises a database and blocks that hold encrypted transactions using algorithms based on immutability. As described in Chapter 1, immutability is the algorithmic assurance that attackers can make no changes to the blockchain without a complete record of it. It is made up of two parts: a world state database and the blocks. The former is a current state copy of the blockchain, and the latter makes up the blockchain. Both record formats utilize a SHA256 hash of the transaction and a Merkle tree for validation. The current state copy is a part of the framework because it provides a database query method that consumes a lot of energy. The STP validation tool requires quick searches because it manages each STP frame in the Ethernet LAN

The capabilities and options for node configuration are defined in the *config.YAML* file in *artifacts/channel*. The figure shows options available for setting rights for Org1. The read-write capabilities for Org1 are established when the nodes are deployed with this script. The orderer node is used to establish a channel and create the genesis block in the blockchain. It sub-

sequently organizes blocks and the transactions inside of them. The orderer node also establishes authentication with the peer nodes by defining access control to their associated port numbers. The peer nodes on this network are called *peer0.lanadmin.stpverify.com* and *peer0.lan.stpverify.com*. *Org1* uses the peer node named *lanadmin* with write access. This peer node manages the transactions to write or invoke a transaction to the blockchain. It runs its copy of the Chaincode through a network channel called *mychannel*. It also has an authenticated connection to one of the database nodes called *couchdb0.stpverify.com*. This database has a hashed list of transactions that are written to the blockchain. Org1 uses a human network administrator to validate new bridges added to the Ethernet LAN and submit their bridge IDs to the blockchain. The peer node called *lan* has read access and is defined as *Org2*. It also has an installed copy of the Chaincode, a connection through *mychannel* for authenticated querying, and an authenticated connection to the *couchDB1.stpverify.com* database node. The *Org2 lan* peer is available to run searches in the blockchain. The research solution requires an automated search of the current bridge IDs extracted from the $root\_out$ file in *proc*. Parts 2, 3, and 4 are a part of a typical Hyperledger Fabric framework that includes a NodeJS app, Chaincode, and blockchain ledger. The Chaincode is proprietary nomenclature for Smart Contracts used in Hyperledger Fabric, an open-source Linux Foundation project.

Part 1: The Javascript Interface

The */scripts* folder in the Github root directory holds *checkData.js*. This file collects the Root ID that is output from *root_out*. It sends the data to the Chaincode to run a search in the blockchain to see if there are earlier records of this bridge ID in the Ethernet LAN. If there is a record, it returns '0' to the $root\_in$ file. If there is no record of this bridge ID previously, it returns a '1'. This data is later processed in the kernel to complete the functionality of this tool. The JavaScript interface initiates a connection as a user with read-only rights to the blockchain by retrieving a token for user validation. The user login information is located in a *CouchDB* container holding user information on localhost. It returns a token *resp.data.token* as identity verification and authenticates the user. The JWT token is used to manage access to the Chaincode for identity verification at each endpoint in the transaction. After identity verification through PKI, the *checkRootIdAvailability()* function retrieves one or more Root IDs from $root\_out$.csv and sends them to the Chaincode to run a read-only search to the CouchDB current state database in the Blockchain Ledger for matches. The result is an *isAvailable* variable that marks zero for false and one for true. This result is written to a file called $root\_in$.csv as validation output.

Part 2: The NodeJS app

The NodeJS app is written in Javascript and manages the modules required to create a web interface that connects to the Chaincode and blockchain artifacts through PKI verified connections. It connects to a UI that provides a visual interface that increases the accessibility of the blockchain's functionality to institutional users. Due to the test environment's limitations, the nodes and connections in the framework are accessed by a localhost network inside the virtual machine housing the STP validation tool. The NodeJS front-end code is available in the */api-2.0* folder of Github's root directory. The blockchain framework's front-end NodeJs application is initialized with token verification through permissioned organizational usernames established for the blockchain network. The *jwt.verify()* function is token authentication to all endpoints for every transaction made to query and invoke the blockchain. The server connection is initialized on localhost port 4000. The user *mila* is enrolled and verified with read-write access through *Org1* using token verification. The user tokens are stored in *api-2.0/org1-wallet/*.

Chaincode connections are token authenticated for adding and querying bridge IDs in the blockchain using the app.post() and *app.get()* functions. When an organizational user with read-write access submits a Root ID to the user interface, this function will add it to the blockchain. The app.get() function is used to query or search CouchDB current state database, and the *app.post()* function is used to add a transaction to the CouchDB current state database and the blockchain. All transactions are hashed using SHA256 and validated through Merkle Tree hashing after they have been added. Per access control rights managed by the Chaincode, lanadmin users have read-write access and can act as the network administrators that manually add new Bridge IDs to the network. Read-only rights are available to lan users who can run a search. Although managed by the front-end application using *app.post()* and *app.get()*, the UI can search and add Root IDs.

Part 3: The Chaincode

In the blockchain framework made by Hyperledger Fabric, the Chaincode is the programming that runs the decision-making and programming for the blockchain. A Smart Contract is software-based business logic. It allows practical uses for a blockchain, such as supply chain management, identity management, and contracts by adding a software program to the blockchain. For example, a Smart Contract can be used to ensure a monthly rental payment. It will keep track of the funds available and issue the digital payment based on a programmed time stamp. This contract uses digital payments and allows transparency, so the cash transaction re-

ceiver is assured that the funds are available or that the payer will not go back on the contract. In the same light, the STP validation tool's contract will pull the Root ID, run a search, and allow a network administrator to add new ones to the blockchain manually. The Chaincode is written in a file called bridge.go. In main, it imports open-source Hyperledger Fabric libraries that can be used to automate basic asset transactions connected to a blockchain. SmartContract is an open-source Hyperledger Fabric proprietary library of functions for Smart Contract development. The DesignatedBridge object defined in lines 19-25 creates the assets queried and added to the blockchain. Each asset added to the blockchain will have a Bridge ID, the authenticated user who added it when added, and the user's organization. The Chaincode is written an open-source template provided by Hyperledger Fabric written in Go, a language by Google. Go, as a language for Smart Contracts, provides efficient integration into the Hyperledger Fabric blockchain network. The original Smart Contracts, written in solidity for Ethereum-based or ERC-20 blockchains, require the use of a virtual machine to provide a platform to execute a Smart Contract. The Go package comes with executables that allow Smart Contracts written in Go to compile and translate to machine code, making it as efficient as C. It also integrates with Javascript objects for running the blockchain app and UI. Integrating Chaincode written in Go will execute in the NodeJS runtime environment to allow for the secure modularity required in the peered blockchain framework increasing the overall efficiency and speed of Chaincode execution. The SmartContract library available by Go, has functions used for the management of the DesignatedBridge asset. A Chaincode consists of initialization, where it is first deployed to the blockchain network and connected to the peers. Each of the peers run in separate docker containers with its instance of the Chaincode. Figure x shows the initialization of a new asset object called contractapi in the main() function. Line 11 includes the package contractapi used to manage the Smart Contract and its contents.

Once the network connections among the Chaincode, orderer, and peers are authenticated, assets that are managed by the blockchain can be queried or invoked. The Chaincode is the program that has function definitions and calls to run the queries and invocations. In a query, the blockchain can be searched, and an invocation would allow a modification to the blockchain, such as a new transaction that is added.

The CreateBridgeID() function is used to invoke the new bridge ID to the blockchain. It passes in the Smart Contract object, called ctx contractapi. The package contractapi included from fabric-contract-api-go/contractapi defines the parameters that are used for the Chaincode. TransactionContextInterface is used to create an interface for transactions to manage the identity of the creator. The last parameter, bridgeData string, holds the details that will be added.

Part 4: The Blockchain Ledger

There are two parts to the management of the Hyperledger Fabric blockchain ledger; the record of all of the transactions made to the blockchain. An invocation to the blockchain is a transaction added to it. It is added to the world state and the blockchain. The world state keeps a copy of all of the transactions made and a hashed value that maintains an updated record of all previous hashed transactions. However, it has the format of a traditional database with differences that provide more security for its record-keeping. The transactions are added to the world state using the Merkle tree algorithm. This study's world state is an Apache CouchDB container and provides a bridge to the JSON structure used by the Hyperledger Fabric framework. It is object-based and integrates with the NodeJS App control center and UI for querying and invoking transactions. The blockchain holds the transactions in the form of blocks, each holding a set number of transactions. As implemented by the Merkle tree algorithm, each block has a unique identifier: the hashed value of all of the hashed transactions within it. Each subsequent block holds an identifier of the hashed identifiers of blocks previous to it. Due to the Merkle tree algorithm, the blockchain is immutable, and any changes made to any transactions will show a record of the change and its previous state.

The World State

The world state is provided by an Apache CouchDB database that holds the transactions and their hashed contents in a container. It holds a copy of the blockchain's current state and all of the transactions in it for an accessible search and view. It provides a UI that allows the user to look at the hashed contents of the blockchain ledger and see the transactions that have been invoked to the blockchain. As does the blockchain, the contents of the CouchDB database include the Merkle tree hashes that ensure the database's immutability. It also makes for faster queries because parsing through the blocks of a blockchain is commonly slow. The use of the database makes for a quicker way to run searches for Bridge IDs, hence, speeding up the STP validation process. The use of the world state as an integral part of the blockchain ledger is a part of the Hyperledger Fabric network. For this study, the option of database queries from the Chaincode to the CouchDB world state allows more efficient speeds for Ethernet traffic. The CouchDB structure works hand-in-hand with the JSON structure of the NodeJS framework making authenticated queries and invoke calls from the main App.js and UI direct and efficient.

The Blockchain

The blockchain is the primary security foundation of the framework. It could be classified as a database; however, the algorithms drive it to provide additional functionality. The blockchain

is named as such because it is made up of blocks. The blocks are made up of a fixed set of transactions based on size. Typically, the block size is set in the algorithm based on efficiency. Authentication is required to add a transaction to the database. The transaction is hashed using the SHA256 has an algorithm and added to the block. Every transaction hash in the block is then combined with SHA256 to create one hash that creates the unique identifier in the block's header. This unique identifier is then hashed with its predecessor's unique identifier. Like a ripple effect, the Merkle tree will change the header SHA256 hash values of the previous blocks to integrate with the blockchain's latest addition. Ultimately, the genesis block, or the first block in the blockchain's header hash value, will change to integrate every blockchain's hash value. This algorithm and the hash values assignment to the headers make any change to a transaction in the blockchain a herculean effort. It is nearly impossible because one change to a transaction will change every block's hash values in its path and, ultimately, the genesis block. If the Smart Contract requires a change in the transaction, blockchain is designed to allow that change and a different version. In this light, a record of the initial transaction will be kept in the blockchain, and the change will be added after that to make the change transparent. Blocks are saved as small fixed-sized.

Appendix: Data Collection

Appendix C is a place for all of the data collection that was used for measurements, graphs and charts that were made. The raw data was organized into trials to make calculations of averages. The appendix will include data collected for the baseline CPU averages for six trials, baseline RAM averages for six trials, STP DApp averages for CPU usage, STP DApp averages for RAM usage, Time to block the Forged Frames and the Average times per trial out of six trials.

Baseline CPU Averages

							total
Sec	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**avg**
1	7	2	2	2	2	2	**2.83**
2	23	20	22	20	21	22	**21.33**
3	2	2	2	2	4	4	**2.67**
4	1	5	3	3	1	1	**2.33**
5	0	0	1	1	1	3	**1**
6	2	1	2	2	1	0	**1.33**
7	1	1	1	2	3	1	**1.5**
8	1	1	1	1	2	1	**1.17**
9	1	0	1	1	1	2	**1**
10	1	0	2	1	0	0	**0.67**
11	1	1	1	1	1	3	**1.33**
12	1	1	0	1	1	0	**0.67**
13	0	0	0	0	0	0	**0**
14	0	1	0	0	1	0	**0.33**
15	0	1	0	0	0	1	**0.33**
16	1	0	1	0	0	0	**0.33**
17	0	1	0	0	1	0	**0.33**
18	0	0	0	2	2	0	**0.67**
19	0	0	1	0	0	0	**0.17**
20	0	0	2	1	0	0	**0.5**
21	0	0	0	0	1	3	**0.67**
22	0	0	0	0	0	0	**0**
23	0	0	0	0	0	0	**0**
24	0	0	0	0	0	0	**0**
25	0	0	1	0	1	0	**0.33**
26	1	2	0	0	1	1	**0.83**
27	0	0	1	0	0	0	**0.17**
28	0	0	0	0	2	0	**0.33**
29	0	0	0	3	0	0	**0.5**
30	0	0	2	0	0	0	**0.33**
31	0	0	0	0	0	0	**0**
32	0	0	0	0	0	2	**0.33**
33	2	1	0	0	0	0	**0.5**
34	0	0	0	0	0	0	**0**
35	0	0	1	0	0	0	**0.17**
36	2	0	0	0	0	1	**0.5**
37	0	2	0	0	0	0	**0.33**
38	0	0	0	0	2	0	**0.33**
39	0	0	0	1	1	0	**0.33**
40	0	0	2	1	0	0	**0.5**
41	0	0	0	0	0	0	**0**
42	0	0	0	1	0	0	**0.17**
43	0	0	0	0	0	5	**0.83**
44	0	0	0	0	0	2	**0.33**
45	0	0	0	0	1	3	**0.67**
46	2	0	0	0	0	0	**0.33**
47	0	0	0	0	0	0	**0**
48	1	2	0	0	2	0	**0.83**
49	1	0	0	0	0	0	**0.17**
50	0	0	1	0	0	0	**0.17**
51	0	0	0	2	0	1	**0.5**
52	0	0	0	0	0	1	**0.17**

							total
53	0	0	0	0	0	2	**0.33**
54	0	0	0	1	0	0	**0.17**
55	0	0	0	0	1	0	**0.17**
56	2	0	0	0	1	0	**0.5**
57	0	1	0	0	0	0	**0.17**
58	0	1	0	0	2	1	**0.67**
59	0	0	0	1	0	0	**0.17**
60	0	0	2	1	0	0	**0.5**
61	0	0	0	0	0	1	**0.17**
62	1	0	0	2	0	0	**0.5**
63	2	0	1	0	0	1	**0.67**
64	0	0	0	1	0	1	**0.33**
65	0	0	3	0	0	0	**0.5**
66	0	0	0	0	0	0	**0**
67	2	1	0	0	0	3	**1**
68	0	0	0	2	2	0	**0.67**
69	0	1	1	0	0	1	**0.5**
70	0	0	2	0	0	1	**0.5**
71	0	0	0	0	0	2	**0.33**
72	0	2	0	0	0	0	**0.33**
73	0	0	0	1	0	1	**0.33**
74	1	0	0	0	2	0	**0.5**
75	0	0	0	0	0	2	**0.33**
76	0	1	0	0	0	0	**0.17**
77	0	1	0	0	0	0	**0.17**
78	3	1	0	0	2	0	**1**
79	1	0	0	1	2	0	**0.67**
80	0	0	2	0	1	0	**0.5**
81	0	0	0	0	1	1	**0.33**
82	0	0	0	0	1	0	**0.17**
83	0	0	0	0	0	0	**0**
84	0	0	0	2	0	0	**0.33**
85	0	0	0	0	0	0	**0**
86	1	0	1	0	1	2	**0.83**
87	0	2	0	0	0	0	**0.33**
88	0	0	0	0	2	0	**0.33**
89	2	0	0	1	0	0	**0.5**
90	0	0	0	0	0	0	**0**
91	0	0	2	1	0	0	**0.5**
92	0	0	1	0	0	1	**0.33**
93	1	0	0	0	4	0	**0.83**
94	0	0	0	0	4	0	**0.67**
95	0	0	2	2	3	0	**1.17**
96	0	0	0	0	0	0	**0**
97	0	2	0	0	4	1	**1.17**
98	0	0	0	0	2	0	**0.33**
99	0	0	0	1	0	1	**0.33**
100	1	0	0	0	0	0	**0.17**
avg	**0.68**	**0.57**	**0.67**	**0.65**	**0.88**	**0.81**	**0.71**

Baseline RAM Averages

totram

Secondt1	t2	t3	t4	t5	t6	average
1	1118456	1143728	1142780	1154144	1154496	1154776	**1144730**
2	1118704	1144096	1142984	1154360	1154744	1155024	**1144985**
3	1118704	1144096	1142984	1154360	1154768	1155044	**1144993**
4	1118704	1144008	1142984	1143416	1154804	1155044	**1143160**
5	1118704	1144008	1142984	1143424	1154804	1144092	**1141336**
6	1118632	1144008	1142984	1143424	1154804	1144092	**1141324**
7	1118632	1144008	1142984	1143336	1143808	1144092	**1139477**
8	1118632	1144008	1142984	1143336	1143668	1144092	**1139453**
9	1118632	1144008	1142984	1143336	1143668	1144092	**1139453**
10	1118632	1144008	1142872	1143348	1143672	1144092	**1139437**
11	1118632	1144008	1142872	1143348	1143672	1143948	**1139413**
12	1118632	1144008	1142880	1143348	1143672	1143948	**1139415**
13	1118632	1144008	1142880	1143348	1143672	1143948	**1139415**
14	1118632	1144008	1142880	1143348	1143672	1143948	**1139415**
15	1118632	1144016	1142880	1143348	1143672	1143948	**1139416**
16	1118640	1144016	1142880	1143348	1143672	1143996	**1139425**
17	1118640	1144016	1142884	1143348	1143672	1143996	**1139426**
18	1118648	1144024	1142884	1143344	1143676	1143996	**1139429**
19	1118648	1144024	1142900	1143344	1143676	1143996	**1139431**
20	1118648	1144024	1142896	1143360	1143676	1144020	**1139437**
21	1118648	1144024	1142896	1143360	1143676	1143972	**1139429**
22	1118648	1144032	1142896	1143360	1143676	1143972	**1139431**
23	1118648	1144032	1142896	1143360	1143724	1143972	**1139439**
24	1118648	1144032	1142896	1143360	1143724	1143972	**1139439**
25	1118648	1144032	1142896	1143360	1143724	1143972	**1139439**
26	1118648	1144032	1142896	1143408	1143740	1143972	**1139449**
27	1118648	1144032	1142896	1143408	1143740	1143972	**1139449**
28	1118648	1144052	1142896	1143408	1143692	1143972	**1139445**
29	1118648	1144052	1142896	1143360	1143692	1143972	**1139437**
30	1118648	1144052	1142896	1143376	1143692	1143972	**1139439**
31	1118648	1144052	1142896	1143376	1143692	1143972	**1139439**
32	1118652	1144052	1142896	1143376	1143692	1143972	**1139440**
33	1118652	1144052	1142896	1143376	1143692	1143972	**1139440**
34	1118652	1144060	1142896	1143384	1143692	1143972	**1139443**
35	1118652	1144060	1142948	1143384	1143696	1143976	**1139453**
36	1118648	1144060	1142948	1143384	1143696	1143976	**1139452**
37	1118680	1144056	1142948	1143384	1143720	1143976	**1139461**
38	1118680	1144056	1142948	1143384	1143732	1143976	**1139463**
39	1118680	1144056	1142948	1143384	1143732	1143976	**1139463**
40	1118680	1144056	1142900	1143408	1143732	1143976	**1139459**
41	1118680	1144056	1142900	1143408	1143732	1143976	**1139459**
42	1118680	1144056	1142900	1143408	1143732	1143976	**1139459**
43	1118680	1144056	1142900	1143408	1143732	1143980	**1139459**
44	1118680	1144056	1142900	1143408	1143732	1143980	**1139459**
45	1118680	1144060	1142900	1143408	1143732	1143980	**1139460**
46	1118680	1144060	1142900	1143408	1143732	1143980	**1139460**
47	1118680	1144060	1142900	1143408	1143732	1143980	**1139460**
48	1118680	1144056	1142900	1143408	1143732	1143980	**1139459**
49	1118680	1144080	1142900	1143408	1143732	1143980	**1139463**
50	1118696	1144080	1142900	1143408	1143732	1143980	**1139466**
51	1118696	1144080	1142900	1143416	1143732	1143980	**1139467**
52	1118696	1144080	1142900	1143416	1143732	1144036	**1139477**

totram

53	1118696	1144080	1142900	1143416	1143732	1143992	**1139469**
54	1118696	1144080	1142900	1143416	1143732	1143992	**1139469**
55	1118696	1144080	1142904	1143416	1143732	1143992	**1139470**
56	1118704	1144080	1142904	1143416	1143732	1143992	**1139471**
57	1118704	1144080	1142904	1143416	1143732	1143992	**1139471**
58	1118704	1144088	1142904	1143416	1143740	1143992	**1139474**
59	1118704	1144088	1142904	1143416	1143740	1143992	**1139474**
60	1118704	1144088	1142900	1143416	1143740	1143996	**1139474**
61	1116276	1144088	1142900	1143416	1143740	1143996	**1139069**
62	1116280	1144100	1142960	1143452	1143772	1144028	**1139099**
63	1116304	1144116	1142960	1143452	1143772	1144044	**1139108**
64	1116304	1144116	1142960	1143452	1143772	1144040	**1139107**
65	1116304	1144116	1142960	1143452	1143772	1144040	**1139107**
66	1116304	1144116	1142960	1143452	1143772	1144040	**1139107**
67	1116300	1144116	1142960	1143468	1143772	1144040	**1139109**
68	1116300	1144112	1142960	1143468	1143776	1144084	**1139117**
69	1116300	1144112	1142960	1143468	1143776	1144084	**1139117**
70	1116300	1144112	1142984	1143468	1143792	1144084	**1139123**
71	1116300	1144112	1142984	1143468	1143796	1144084	**1139124**
72	1116300	1144112	1142984	1143468	1143796	1144084	**1139124**
73	1116312	1144112	1142984	1143476	1143796	1144084	**1139127**
74	1116312	1144112	1142984	1143476	1143796	1144084	**1139127**
75	1116312	1144112	1142984	1143476	1143796	1144048	**1139121**
76	1116312	1144144	1142984	1143476	1143796	1144048	**1139127**
77	1116312	1144144	1142984	1143484	1143796	1144048	**1139128**
78	1116376	1144152	1142984	1143484	1143800	1144048	**1139141**
79	1116376	1144152	1142984	1143484	1143800	1144048	**1139141**
80	1116376	1144160	1142984	1143484	1143800	1144068	**1139145**
81	1116376	1144160	1142984	1143484	1143820	1144068	**1139149**
82	1116376	1144160	1142984	1143496	1143820	1144068	**1139151**
83	1116376	1144160	1142984	1143496	1143820	1144068	**1139151**
84	1116376	1144160	1142984	1143496	1143820	1144072	**1139151**
85	1116376	1144160	1142984	1143504	1143820	1144072	**1139153**
86	1116376	1144160	1142984	1143504	1143880	1144068	**1139162**
87	1116376	1144160	1142984	1143504	1143880	1144068	**1139162**
88	1116376	1144160	1142984	1143504	1143816	1144068	**1139151**
89	1116380	1144160	1142984	1143504	1143816	1144068	**1139152**
90	1116380	1144160	1142984	1143560	1143816	1144068	**1139161**
91	1116380	1144160	1143016	1143560	1143816	1144068	**1139167**
92	1116380	1144160	1143016	1143560	1143816	1144068	**1139167**
93	1116380	1144160	1143016	1143560	1143816	1144068	**1139167**
94	1116400	1144160	1143016	1143560	1143816	1144068	**1139170**
95	1116400	1144160	1143016	1143512	1143816	1144068	**1139162**
96	1116400	1144188	1143024	1143512	1143816	1144068	**1139168**
97	1116400	1144188	1143072	1143524	1143896	1144092	**1139195**
98	1116400	1144184	1143072	1143524	1143832	1144092	**1139184**
99	1116400	1144184	1143072	1143524	1143832	1144092	**1139184**
100	1116400	1144184	1143072	1143524	1143832	1144092	**1139184**
AVERA	**1117737**	**1144082**	**1142942**	**1143755**	**1144411**	**1144460**	**1139565**

STP DApp CPU Averages

StpDApp_Avgs

Seconds	t1	t2	t3	t4	t5	t6	average
1	15	17	18	18	19	20	**18**
2	14	77	16	18	19	16	**27**
3	11	80	38	14	40	46	**38**
4	22	83	36	14	74	70	**50**
5	25	80	80	43	76	80	**64**
6	12	84	76	58	76	74	**63**
7	38	77	78	76	76	79	**71**
8	8	78	81	74	77	77	**66**
9	8	80	77	77	72	74	**65**
10	12	75	76	82	74	78	**66**
11	12	80	81	78	77	79	**68**
12	38	75	81	80	81	81	**73**
13	9	76	78	80	75	75	**66**
14	8	82	84	81	79	82	**69**
15	9	75	81	78	79	74	**66**
16	9	82	83	74	76	73	**66**
17	9	78	80	83	75	72	**66**
18	11	79	76	78	80	79	**67**
19	11	79	82	80	72	70	**66**
20	36	79	78	79	73	74	**70**
21	7	80	79	76	27	79	**58**
22	11	74	80	84	13	32	**49**
23	11	12	77	17	12	14	**24**
24	27	20	81	15	12	28	**31**
25	38	10	77	14	12	14	**28**
26	8	12	82	14	27	32	**29**
27	10	10	79	14	14	71	**33**
28	9	10	79	32	15	75	**37**
29	7	13	72	16	15	74	**33**
30	13	13	14	75	29	78	**37**
31	18	11	12	78	15	75	**35**
32	15	11	10	78	25	43	**30**
33	18	11	11	82	76	13	**35**
34	34	10	11	77	73	16	**37**
35	41	11	17	77	75	14	**39**
36	60	11	14	83	75	15	**43**
37	21	14	13	32	79	16	**29**
38	28	12	13	13	74	44	**31**
39	67	13	13	15	71	13	**32**
40	12	11	12	15	15	19	**14**
41	9	11	12	11	12	12	**11**
42	13	11	10	14	14	15	**13**
43	21	12	12	13	13	16	**15**
44	27	13	12	36	12	14	**19**
45	14	11	15	16	13	14	**14**
46	13	11	14	20	15	14	**15**
47	18	27	20	14	37	17	**22**
48	34	24	12	13	15	15	**19**
49	33	13	12	30	22	13	**21**
50	10	13	16	17	21	12	**15**
51	15	13	28	16	16	12	**17**
52	39	12	74	14	12	13	**27**

								StpDApp_Avgs
	53	23	13	81	67	56	14	**42**
	54	12	13	76	76	77	14	**45**
	55	10	14	78	53	73	11	**40**
	56	29	9	77	14	59	15	**34**
	57	28	14	77	15	12	17	**27**
	58	12	15	83	13	15	14	**25**
	59	8	16	76	14	14	15	**24**
	60	11	11	74	13	14	14	**23**
	61	8	11	82	14	12	11	**23**
	62	14	13	59	14	13	14	**21**
	63	14	33	13	13	14	15	**17**
	64	14	12	12	11	14	16	**13**
	65	11	14	16	14	15	14	**14**
	66	39	14	13	12	16	12	**18**
	67	14	12	13	12	21	15	**15**
	68	9	13	14	11	28	13	**15**
	69	14	17	13	14	24	16	**16**
	70	13	30	10	13	15	14	**16**
	71	9	13	14	13	14	20	**14**
	72	11	13	13	17	14	14	**14**
	73	10	11	15	13	13	16	**13**
	74	11	14	14	12	13	13	**13**
	75	9	12	14	44	13	13	**18**
	76	10	12	14	12	16	22	**14**
	77	10	11	14	14	14	14	**13**
	78	7	13	14	16	13	15	**13**
	79	10	16	11	24	14	33	**18**
	80	11	14	14	39	16	23	**20**
	81	8	15	11	17	15	18	**14**
	82	10	13	32	44	13	40	**25**
	83	7	13	19	33	13	15	**17**
	84	10	16	14	28	12	42	**20**
	85	6	9	12	20	12	13	**12**
	86	8	12	26	16	14	16	**15**
	87	8	12	18	18	13	14	**14**
	88	8	12	15	53	13	15	**19**
	89	7	18	17	23	13	17	**16**
	90	11	15	14	27	12	16	**16**
	91	10	11	13	31	12	13	**15**
	92	8	14	12	20	13	15	**14**
	93	9	9	12	39	12	14	**16**
	94	7	11	21	34	13	14	**17**
	95	11	12	12	23	15	16	**15**
	96	9	14	12	22	16	13	**14**
	97	7	12	13	32	14	13	**15**
	98	11	12	16	26	12	15	**15**
	99	8	16	15	23	13	12	**15**
	100	8	8	11	23	13	13	**13**
average		**15.7**	**27**	**37.8**	**35**	**32.2**	**30.9**	**29.845**

STP DApp RAM Averages

bcram

Second	t1	t2	t3	t4	t5	t6	average
1	2732440	3763412	3920164	4221588	4442288	4731688	**3968596.667**
2	2732640	3767540	3920492	4221792	4442456	4732008	**3969488**
3	2732648	3771288	3920520	4221976	4445392	4734952	**3971129.333**
4	2732672	3775620	3929548	4222000	4458076	4748048	**3977660.667**
5	2732672	3782992	3937696	4222392	4505648	4750908	**3988718**
6	2719748	3790880	3941564	4236624	4511036	4755408	**3992543.333**
7	2732676	3793612	3947320	4244600	4514760	4758624	**3998598.667**
8	2732724	3798476	3951864	4252588	4517840	4761644	**4002522.667**
9	2732764	3802000	3958004	4262144	4523512	4766784	**4007534.667**
10	2732764	3806116	3963264	4268792	4527060	4770488	**4011414**
11	2719756	3809256	3970936	4274684	4529428	4774000	**4013010**
12	2732736	3814516	3976004	4278412	4533624	4819920	**4025868.667**
13	2732772	3818204	3980704	4281232	4538140	4824460	**4029252**
14	2732772	3822508	3985192	4330164	4541716	4827932	**4040047.333**
15	2732772	3822668	3989804	4334756	4544876	4831668	**4042757.333**
16	2732772	3826040	4036096	4339260	4549340	4835344	**4053142**
17	2732772	3830264	4040192	4342784	4553632	4838512	**4056359.333**
18	2732772	3834312	4044356	4346800	4557584	4842288	**4059685.333**
19	2732784	3837472	4048596	4350724	4561764	4845768	**4062851.333**
20	2733084	3840996	4053424	4354684	4561212	4845648	**4064841.333**
21	2733084	3843848	4053208	4358236	4561988	4849504	**4066644.667**
22	2733084	3847484	4058600	4360780	4561988	4850396	**4068722**
23	2733084	3847720	4062276	4360996	4562000	4850396	**4069412**
24	2733100	3889340	4065592	4360904	4562000	4850480	**4076902.667**
25	2733112	3889456	4069772	4360936	4562012	4835504	**4075132**
26	2733112	3889456	4073508	4360548	4562072	4839304	**4076333.333**
27	2733132	3889456	4076688	4360996	4562136	4851140	**4078924.667**
28	2733140	3889456	4080776	4361036	4562148	4855156	**4080285.333**
29	2733080	3889456	4083440	4343888	4562148	4859444	**4078576**
30	2733080	3889480	4083452	4356860	4562148	4863812	**4081472**
31	2733280	3889332	4083452	4360432	4548252	4868240	**4080498**
32	2733308	3889436	4083464	4364100	4551156	4869952	**4081902.667**
33	2733364	3889436	4083520	4367564	4564056	4870100	**4084673.333**
34	2734864	3889444	4083520	4371220	4610088	4870044	**4093196.667**
35	2749668	3889500	4083548	4376176	4614456	4870048	**4097232.667**
36	2765552	3889512	4083560	4379984	4619740	4870104	**4101408.667**
37	2765528	3889516	4083568	4380920	4623504	4855324	**4099726.667**
38	2765536	3889516	4083568	4380932	4628100	4868192	**4102640.667**
39	2780080	3889532	4083568	4380932	4631356	4868360	**4105638**
40	2780080	3889532	4083576	4380940	4631728	4913332	**4113198**
41	2779744	3889540	4083612	4380940	4631728	4913332	**4113149.333**
42	2781328	3889540	4083620	4380940	4631868	4913324	**4113436.667**
43	2781692	3889540	4083648	4380952	4631868	4913372	**4113512**
44	2780048	3889620	4083656	4381052	4631964	4913400	**4113290**
45	2780052	3889632	4083672	4381064	4631964	4913408	**4113298.667**
46	2780052	3889644	4083672	4427284	4631972	4913416	**4121006.667**
47	2780052	3889644	4083696	4427304	4631944	4913468	**4121018**
48	2780052	3889776	4083700	4427308	4631948	4913476	**4121043.333**
49	2780108	3889776	4083700	4427308	4631948	4913476	**4121052.667**
50	2780116	3889796	4069228	4410748	4632060	4913496	**4115907.333**
51	2780076	3889796	4073356	4410824	4616520	4913496	**4114011.333**
52	2779984	3889796	4084972	4410892	4616408	4913496	**4115924.667**

bcram

53	2779984	3889796	4133776	4424584	4630320	4913500	**4128660**
54	2779996	3889796	4138996	4428552	4633612	4913500	**4130742**
55	2779996	3889796	4143928	4431540	4640020	4913500	**4133130**
56	2780072	3889680	4149332	4431540	4646260	4913512	**4135066**
57	2780072	3889680	4157176	4431588	4646276	4913528	**4136386.667**
58	2780072	3889688	4163060	4431596	4646308	4913552	**4137379.333**
59	2780100	3889704	4169896	4431600	4646308	4913564	**4138528.667**
60	2780100	3889704	4176944	4431600	4646476	4913564	**4139731.333**
61	2780100	3889720	4183688	4431600	4646496	4913584	**4140864.667**
62	2780112	3889720	4189376	4431644	4646496	4913584	**4141822**
63	2780108	3889824	4189404	4431532	4646596	4913620	**4141847.333**
64	2780212	3889824	4189508	4431536	4646696	4913644	**4141903.333**
65	2780128	3889832	4189604	4431536	4646696	4913664	**4141910**
66	2781248	3889832	4189620	4431540	4646780	4913676	**4142116**
67	2768268	3889840	4189664	4431568	4692712	4913728	**4147630**
68	2768256	3889740	4189744	4431568	4692744	4913756	**4147634.667**
69	2767472	3889784	4189820	4431568	4692764	4913724	**4147522**
70	2767496	3889836	4189848	4431572	4692844	4913604	**4147533.333**
71	2767496	3889836	4189924	4431572	4692876	4913744	**4147574.667**
72	2767508	3889844	4190044	4431584	4692964	4913744	**4147614.667**
73	2767508	3889844	4190152	4431584	4693056	4913780	**4147654**
74	2767512	3889856	4190172	4431640	4693056	4913788	**4147670.667**
75	2767512	3889856	4190216	4431664	4693172	4913888	**4147718**
76	2767428	3889868	4190244	4431680	4693240	4959500	**4155326.667**
77	2767428	3889720	4190376	4431692	4693292	4959676	**4155364**
78	2767428	3889800	4190396	4431696	4693316	4959716	**4155392**
79	2767428	3889800	4190436	4431704	4693464	4959752	**4155430.667**
80	2767428	3874496	4190532	4431756	4693472	4959896	**4152930**
81	2767456	3874496	4190584	4431756	4693620	4959908	**4152970**
82	2767456	3874524	4190760	4431792	4693636	4959988	**4153026**
83	2767456	3874524	4190812	4409452	4693712	4947228	**4147197.333**
84	2767456	3920188	4190924	4409296	4693840	4959964	**4156944.667**
85	2767456	3920256	4191008	4409316	4693848	4960272	**4157026**
86	2767456	3920260	4191024	4409316	4693892	4960328	**4157046**
87	2767464	3920260	4190948	4409344	4693936	4960416	**4157061.333**
88	2767464	3920260	4191072	4455272	4694068	4960528	**4164777.333**
89	2767464	3920252	4191176	4455352	4694092	4960576	**4164818.667**
90	2767476	3920280	4191020	4455352	4694100	4961716	**4164990.667**
91	2767488	3920280	4191032	4455352	4694100	4961784	**4165006**
92	2767488	3920288	4191240	4455360	4694268	4961944	**4165098**
93	2767496	3920288	4191312	4455380	4694268	4961956	**4165116.667**
94	2767496	3920292	4236436	4455340	4694356	4962056	**4172662.667**
95	2767496	3920300	4236556	4455340	4694356	4962108	**4172692.667**
96	2767512	3920304	4236576	4455344	4694508	4962124	**4172728**
97	2767532	3920304	4236660	4455360	4694512	4962160	**4172754.667**
98	2767532	3920296	4219436	4455360	4694368	4961596	**4169764.667**
99	2767540	3920312	4219456	4455372	4694372	4962128	**4169863.333**
100	2767540	3920320	4219580	4455384	4694648	4962056	**4169921.333**
Averag	**2758825.04**	**3876111.8**	**4111524.16**	**4386876.68**	**4621554.64**	**4890092.08**	**4107497.4**

Times Collected for Six Trials

Trial	Time of Launch	WS detection	Drop Time	New Root ID
1	10:47:11 AM	10:47:11 AM	10:47:17 AM	10:47:40 AM
2	10:48:38 AM	10:48:38 AM	10:48:44 AM	10:49:06 AM
3	10:53:56 AM	10:53:56 AM	10:54:00 AM	10:54:21 AM
4	10:55:17 AM	10:55:17 AM	10:55:22 AM	10:55:43 AM
5	10:58:10 AM	10:58:10 AM	10:58:14 AM	10:58:35 AM
6	11:00:47 AM	11:00:47 AM	11:00:53 AM	11:01:15 AM

Trial	Block Time	"Convergence"
1	6	23
2	6	22
3	4	21
4	5	21
5	4	21
6	6	22