Zero Trust Architecture with Blockchain Capabilities Report

Senior Capstone Project

Proposed by Steven Roodbeen,

On behalf of the Naval Undersea Warfare Center Division Newport

University of Massachusetts Dartmouth Team:

Team Lead: Aaron Almeida

SCRUM Master: Tuan-Son Le

Developer: David Atunlute

Developer: Erica Gomes

Table of Contents

Introduction

Traditional network security relied on perimeter based defenses to protect network resources by trusting internal traffic while verifying external traffic. However, with the increase in unsecure network endpoints, such as IoT devices, cloud services, and remote devices, networks are more vulnerable than ever. Zero trust architecture, also known as zero trust, addresses this problem with the philosophy, “Never Trust, Always Verify”. Zero trust achieves this with a combination of security principles that control network access based on a variety of factors.

The purpose of this Senior Capstone project, proposed by Steven Roodbeen from the NUWC division Newport, is to demonstrate the capabilities of zero trust architecture and blockchain technology in a virtual environment. Each of these respective technologies are built on top of various different components, each with high degrees of granularity that meet the needs of specific implementations. Therefore, this project focuses on the general capabilities of each technology in order to serve as a foundation for future projects prototyping these technologies in a virtual environment.

Due to the sensitive information present in live networks, the implementation of zero trust architecture and blockchain is built on top of a virtual network. As a result, the final architecture interfaces with this network. However, the concepts used by the developed systems are still usable for demonstration and insight. In order to highlight the difference in capabilities that zero trust and blockchain provide, a before and after implementation of the developed systems is required.

Project Vision

Initially while researching zero trust architecture with blockchain capabilities, the team found few resources that enabled the demonstration of a network architecture that used the required technologies. The team realized that the final implementation would need to be built on various different systems that were repurposed for the project. As a result, our team has focused on creating a virtual environment that can be reused or modified for future projects.

Due to the open ended nature of the project, development of the project focused on the general systems that zero trust and blockchain stand to affect the most. The team initially implemented a standard firewall after researching and prototyping network emulation. This firewall controlled network traffic via firewall rules that were configured on the frontend. Considering that standard firewall-based security has been significantly modified or replaced entirely, the team chose to implement a form of single sign-on that would utilize role-based access control. Ultimately the implementation of single sign-on demonstrated the foundational principles of zero trust.

From there various blockchain solutions were prototyped and considered. Initially the team stored firewall rules that were hashed on the blockchain application. These firewall rules initially served as a test to assure that the developed network applications could communicate with our blockchain application. Ultimately the team decided on implementing a blockchain solution that acts as a certificate authority to the single sign-on. The combination of single sign-on with role-based access control and a blockchain based certificate authority served as a thorough implementation that could be demonstrated.

The complete project executes in a virtual linux environment. This virtual environment encapsulates all of the logical components that enable the demonstration, such as the frontend, network, and blockchain. For the purpose of ease of use the project only requires the execution of the frontend. After the execution of the frontend the other systems can be executed through the startup page.

Networking

From the beginning the team realized a network would be needed in order to demonstrate the capabilities of zero trust architecture and blockchain. The ultimate capabilities of the network weren't defined, as the focus of the project was on the relevant technologies; however, the minimum network topology needed to have two nodes connected via a switch. During the initial research for the network different topologies were considered, but ultimately a star topology was chosen to streamline the demonstration.

Network Emulation

At the start of the project various network emulation tools were considered such as, TopGen/GreyBox, Common Open Research Emulator (CORE), Mininet, and VMware NSX. Initially CORE was used to prototype different network architectures. CORE provided a comprehensive graphical user interface that allowed the team to quickly prototype different topologies and configure different network services on each node. While the user interface assisted prototyping, the team realized that the network would need additional services that CORE didn't support. Furthermore, network nodes in CORE each required tedious configuration of bash scripts to have any functionality, which wasn't conducive to development or demonstrability.

Mininet was initially considered for use as the network emulator; however, at the time the final architecture wasn't clear to the team so CORE was perceived to offer more functionality. Mininet provides a command line interface that can generate different topologies of varying sizes with a single command. Alternatively the Mininet emulation can be initialized in an executable python file, which

would ultimately be useful for the demonstration. By centralizing our network configuration to one python file, instead of various bash scripts, the team was able to easily configure and modify the network.

Limitations

Both CORE and Mininet can technically be used to prototype actual network architectures. This is useful for experimenting with network architecture without having to risk the stability of the actual network. CORE and Mininet create virtual environments, technically network namespaces, for each network node that inherit the majority of functionality from the host operating system. Network namespaces are able to interact with the same network interfaces available to the host. As a result, network nodes can interact with each other and other as well as other services that have a network interface.

The main problem the team faced involving the emulated network was the lack of a clear network architecture. Initially, various services were considered such as mock applications and databases, but there was no clear way to demonstrate these services without utilizing the command line, which was not conducive to the demonstration. Furthermore, developing and connecting example services to demonstrate a hypothetical network use case took away from the actual development of zero trust and blockchain.

One of the network emulators considered, VMware NSX, seemed to streamline the development of zero trust architecture with network services; however, NSX is commercial software that requires a license to access the complete features. Also, this didn't solve the problem regarding a lack of network services. Ultimately, the team focused on a general network architecture in order to focus on the demonstration of zero trust and blockchain.

Software Defined Networking

Early into prototyping different network emulators the team found that configuring routing protocols and other services was tedious and difficult to modify. As a result, the team chose to implement software defined networking (SDN) into the network architecture. SDN enables organizations to quickly deploy and manage network applications without the typical costs. Furthermore, network administrators can easily modify network architecture without having to directly configure existing network hardware.

Software defined networking works by implementing a programmable control plane that interfaces with the network hardware via a southbound interface. This control plane can also interface with other network applications via a northbound interface in order to control the flow of network traffic.

The separation of southbound and northbound interfaces enabled the team to separate development between the network and the network applications. Ultimately the team found that the modularity and ease of use of SDN made it the right direction to take the project.

Ryu

Ryu is a component-based software defined networking framework that is used to quickly create and prototype controllers. These controllers interface with the OpenFlow protocol, a southbound interface, that handles the flow of network traffic. Ryu provided the project the necessary components to develop our network applications and connect them to our frontend.

Zero Trust Architecture

The standard cybersecurity model involves a trusted network encompassed by a security perimeter with all traffic coming from the outside being untrusted. This approach is problematic due to its assumptions that threats are usually only external. In general zero trust architecture as a concept, is understood to be the shift from the perimeter based standard cyber security model to a perimeterless and trustless model. The key tenant defined by the founder of zero trust architecture, John Kindervag, is, “Never trust, always verify”.

Organizations implement this key tenant by constantly reevaluating access to resources regardless of who or where the user is. Access is granted if a user and the user’s request are verified. Verifying a user’s identity is usually done via a form of multi-factor authentication, such as passwords, SMS, biometrics, or location data. In general the more forms of authentication an organization supports, the harder it is to fake one’s identity. On the other hand, a user’s request is verified based on various factors. These factors can vary based on the level of authentication a user has, but generally the permissions associated with the user and the resource being accessed are considered. The system can also consider the behavior of the user, such as if the user is accessing work resources during appropriate times. Ultimately, zero trust principles enable security that takes into account external and internal threats.

Our Approach

The implementation of zero trust is typically based around a set of principles, but implementations of zero trust can vary greatly based on the needs and capabilities of the network. Typically zero trust architectures contain the following concepts: least privilege access policy, microsegmentation, and constant monitoring. Least privilege access policy is only giving enough access

to users to accomplish their tasks. Microsegmentation involves segmenting a network in order to reduce the impact of unauthorized access. Our approach involves highlighting these principles to gain insights into the virtues and limitations of zero trust architecture. The team decided to implement various systems in order to demonstrate the capabilities of these principles.

Firewall

Traditional network security relied on a firewall that served as the security perimeter of the network. Generally, traffic inside the security perimeter was considered trusted, and traffic outside the security perimeter was considered untrusted. This approach worked well when networks were insulated to offices and workspaces that were limited to a building or location; however, multiple flaws with this approach became evident as networks expanded in size and capabilities. Traditional firewalls relied on static rules that were tedious to configure and scale. As networks expanded in size to contain personal devices, IoT devices, remote employees, and other untrusted devices the overall trustworthiness of devices inside the network perimeter declined. Another issue with traditional firewalls was if an attacker bypassed the wall, they would have full access to the protected data and resources. Threats from inside the security perimeter could occur from threat actors gaining unauthorized access to user credentials. Threat actors could gain access to these credentials via social engineering, man-in-the-middle attacks, or other threat vectors.

Single Sign-On

Many organizations have incorporated a form of Single Sign-On (SSO) in order to streamline access control policies between different users and services. An SSO centralizes traffic to one portal that directs traffic to relevant destinations, such as services and databases. As a result, SSO is a scalable system that can centralize network traffic, which enables constant monitoring of network traffic. Furthermore, an SSO acts as a central point to deny access to unauthorized resources.

Role-Based Access Control

Large organizations that require granular access control typically implement some form of role-based access control (RBAC). RBAC is a form of role-based identity and access management that is implemented by creating roles with relative permissions assigned to them. Typically, RBAC is implemented with either discretionary or mandatory access control. Discretionary access control (DAC) allows the owner of a resource to specify the accessibility of the resource, whereas mandatory access control (MAC) sets security labels for users and resources. Overall, MAC provides higher data confidentiality and granularity to sensitive resources than DAC, but DAC is easier to maintain and scale.

For the purpose of our project the team chose to implement RBAC with MAC. Considering the confidential environment that most networks operate in, the demonstration of DAC was deemed more applicable. Furthermore, demonstration of least privilege access policy is straightforward with DAC, unlike MAC.

Limitations

Overall, the developed systems focus on identity and access management (IAM). While IAM is essential for handling network access control to resources, IAM doesn't necessarily handle threat vectors that target networks. The threat vectors that face a network are based on the security systems, or lack thereof, that the network utilizes. Considering the limitations of the emulated network, choosing a specific security system to implement and demonstrate seemed irrelevant, since the network services would be specific to the emulated network and difficult to demonstrate. IAM can and should be augmented with security systems to improve a network's security capabilities, but IAM is not necessarily needed for all security systems.

Blockchain

Recently, blockchain technology has caught the public's attention due to the proliferation of cryptocurrencies, such as Bitcoin and Ethereum. Cryptocurrencies are a logical use of blockchain technology considering the immutable principle that makes blockchain essentially tamper proof; however, blockchain technology has multiple applications that aren't limited to finances. Blockchain technology has multiple other applications, such as supply chain management, record management, identity management, and most importantly cybersecurity. For the purpose of this project the blockchain applications that were considered related to cybersecurity and identity management.

Consensus Algorithm

In a short period of time blockchain technology has disrupted various industries. The vast capabilities of blockchain come from the various technologies that blockchains utilize. Blockchains act as decentralized ledger of information. Essentially, participants in the blockchain contain a piece of the blockchain that "links" with the rest of the chain. The blockchain manages the distributed ledger by imposing a consensus algorithm that serves a mechanism for verifying existing and new data on the ledger. As a result tampering with existing data on the ledger requires circumventing the consensus algorithm, which is possible for some algorithms but generally very difficult.

Vulnerabilities & Threats

Blockchain offers a great deal of improvement to the security of services that are in place as of right now. This has resulted in the increasing use of blockchain in various sectors, but blockchain is not without its vulnerabilities and threats. As established, one of the key strengths of blockchain is the distribution in managing and adding data. In most consensus algorithms, the majority of nodes or blocks in the blockchain have to reach a consensus over the addition of data to the blockchain, which can result in a threat vector. Furthermore, there are various other threat vectors that are present in blockchain applications that can vary based on the features and capabilities present in the blockchain.

The 51% attack is one of the first conceived threat vectors that affect blockchain networks. A 51% attack occurs when a threat actor gains a simple majority of the network hashrate. When this occurs the threat actor can create a copy of new blocks in the blockchain network, which can pose significant problems especially for cryptocurrencies. Due to the nature of blockchains and consensus algorithms, the bigger a network of nodes, the less likely that a blockchain is susceptible to a 51% attack. Once a blockchain network grows large enough, the likelihood that anyone would have the majority of the blockchain hashrate decreases.

Our Approach

Similar to network emulators, our team researched various blockchain solutions that could be implemented for the purpose of demonstrating the relevant capabilities of our project; however, various blockchain solutions for identity management require significant infrastructure and data. Initially the team considered using the Hyperledger Fabric platform to develop a blockchain application; however, the capabilities of Hyperledger Fabric didn't align properly with the capabilities provided by our emulated network.

The developed blockchain solution utilizes a Flask web server that contains the functionality of a simple decentralized application. The initial blockchain application the team used was an open source blockchain project. The original purpose of this blockchain project was to host a decentralized content sharing application, essentially a blog. The blockchain application contains a consensus algorithm that is based on the Proof-Of-Work algorithm, where each new block in the blockchain has to be hashed and verified before being linked with the rest of the blockchain. Furthermore, this consensus algorithm allowed for multiple peers containing pieces of the ledger to be added as long as they satisfied the consensus algorithm. The team used this project as a foundation to create a certificate authority.

Certificate Authority

Certificate authorities (CA) are an integral part of authentication for the internet. CAs handle the distribution and management of digital certificates. Digital certificates are essentially digital documents that are used to authenticate one's identity. A standard format for digital certificates is X.509. The X.509 standard is commonly used for SSL/TLS, which is what the HTTPS protocol uses to establish secure network connections.

For the purpose of our project the blockchain application acts as a certificate authority, where it creates, signs, and manages digital certificates with a root certificate. The frontend generates certificates to be used for the single sign-on by creating and sending a certificate signing request (CSR) to the blockchain application. The blockchain certificate authority generates a digital certificate from the CSR, adds it to the ledger, and returns it to the frontend. The certificates generated are verified by the Ryu single sign-on before network communication can begin.

Demonstration

Ultimately in order to demonstrate the capabilities of the features created, the team needed a graphical user interface that could interface with the project components. This interface needs to simulate the before and after demonstration of zero trust and blockchain. The decision was made to use the Python web framework Flask, which is also used for the blockchain solution, since the majority of the project already used Python.

The frontend user interface acts as a hub to connect the other relevant systems of the project. The firewall, single sign-on, and certificate authority systems are all configured via the user interface. On the other hand, the network connectivity is demonstrated via the Mininet command line interface. The frontend also can view the certificates that are configured by the certificate authority, and the frontend hosts this report in the about section.

Before Simulation

The before simulation involves a topology with three hosts connected to a switch and an implementation of a firewall with static rules. Each connection between two hosts requires two rules for the firewall, which results in a total of 6 rules needed to enable communication between 3 hosts. Each rule contains the direction of one source to a destination, which is why two hosts require 2 rules to communicate. Overall the before simulation is simple due to the focus on demonstrating the necessary capabilities in the after simulation.

After Simulation

The after simulation on the other hand consists of a topology with 5 hosts connected to a switch, a single sign-on implemented with Ryu, and a blockchain based certificate authority. As mentioned previously the implementation of single sign-on utilizes role-based access control with mandatory access control. The implementation of RBAC involves three roles: workers, admins, and servers. These roles are used to represent a straightforward traffic flow between users and sensitive information. Essentially, communication between the workers and servers is restricted through the administrator role. Furthermore, before communication can begin the administrator(s) must be authenticated via the certificate authority.

Avenues of Research & Development

As mentioned in the introduction, this project focuses on the broad capabilities of Zero Trust Architecture and Blockchain technology. As a result there are plenty of features that can be altered, added, or replaced in order to demonstrate the capabilities of a different security environment. There are many additional systems that can be implemented into a zero trust architecture, but the following systems are ones that were researched and considered for our project.

Intrusion Detection & Prevention

Intrusion detection (IDS) and prevention systems analyze network traffic in order to determine potential threats. As a result, intrusion detection and prevention requires inspecting network traffic and behavior in order to detect anomalies. Typically packets are routed to a “hidden” network component, the IDS, and they are inspected in order to create traffic logs.

There are two main types of intrusion detection systems, signature-based and anomaly-based. Signature-based IDS looks for specific “signatures” that are repeated in standard threat vectors. As a result, signature-based IDS relies on a central database to detect threats. These systems can be easily fooled and require constant updates. On the other hand anomaly-based IDS are far more flexible handling unknown threats. Typically a form of AI, such as machine learning, is used to train Anomaly-based IDS. Training an IDS using network data such as traffic logs enables the system to constantly monitor for anything out of the ordinary. The system can then be trained to quarantine potential threats.

Implementation of an intrusion detection system with Ryu was considered. Specifically the Snort IDS was considered. Snort monitors network packets as well as configures rules for packet detection. Implementation of AI based IDS was also considered, but there was a lack of training data. Ultimately,

Snort and AI based IDS were deemed out of the scope of our project as there was a lack of network services. They were also determined to be difficult to demonstrate.

Single Packet Authorization

Single packet authorization (SPA) is an effective method to protect networks from threat vectors such as port scanning. Port scanning is typically conducted during the reconnaissance phase of a threat vector, and it works by probing a network's ports to find an open one that can be accessed. As a result, SPA works by requiring a packet handshake before new connections can be established on a network port. Similar to intrusion detection systems, demonstration and development of SPA was not practical so the team decided to focus effort on the development of other systems.

Final Thoughts

The effects of zero and blockchain adoption are already being seen today. Most organizations have adopted improved security practices that move towards a trustless security environment, such as multi-factor authentication. Also, major institutions are starting to invest in cryptocurrencies, such as Bitcoin, but this is only the tip of the iceberg. As blockchain technology proliferates additional use cases will become more available, and as a result blockchain based applications will become more practical.

We hope that this project can serve as a foundation for future personal and educational projects that look to explore concepts relating to zero trust and blockchain technology in a virtual environment. The team is incredibly grateful for the opportunity provided by the University of Massachusetts Dartmouth and the Naval Undersea Warfare Center Division Newport to learn about cybersecurity and blockchain technology. We would like to specifically thank our Professor Iren Valova, teaching assistant Nathan Leblanc, and client Steven Roodbeen for their guidance and feedback throughout this process.

Further Reading & Resources

CORE Documentation

Mininet: An Instant Virtual Network

What is VMware NSX? | Network Security Virtualization Platform

Zero Trust Architecture, NIST Publication

Ryu SDN Framework

Blockchain Identity Management: The Definitive Guide

Blockchain Attack Vectors: Vulnerabilities of the Most Secure Technology

Flask Blockchain Application

Snort - Network Intrusion Detection & Prevention System

fwknop: Single Packet Authorization > Port Knocking