UNIVERSITY OF TURKU

A Cost-Effective Zero-Trust Approach for Cloud Computing: Experimental Evaluation on AWS Platform

Cyber Security

Master's Degree Programme in Information and Communication Technology

Department of Computing, Faculty of Technology

Master of Science in Technology Thesis

Author:

Aref Mowloughi

Supervisors:

Tahir Mohammad

Antti Hakkala

June 2024

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Cyber Security
Programme: Master's Degree Programme in Information and Communication Technology
Author: Aref Mowloughi
Title: A Cost-Effective Zero-Trust Approach for Cloud Computing: Experimental Evaluation on AWS Platform
Number of pages: 67 pages
Date: June 2024

Abstract

As the volume of data continues to expand and the intricacy of managing conventional on-premise data centers becomes increasingly burdensome, numerous organizations are transitioning their infrastructure to cloud-based solutions. A pivotal aspect of this investigation is the integration of Zero Trust principles into cloud environments, with a particular emphasis on the widely recognized and globally utilized Amazon Web Services (AWS) platform. The thesis examines the limitations of traditional location-based security measures and highlights scenarios where they prove ineffective. Furthermore, it will elucidate how adopting a zero-trust approach can address these shortcomings and offer more robust security solutions in an automated, cost-effective approach. Through examining case studies and adherence to AWS security best practices, the research provides insights into the practical implementation of ZTA on AWS. This encompasses considerations for identity and access management, network segmentation, and real-time monitoring to establish a comprehensive security posture.

Moreover, the study assesses the feasibility of providing automated solutions for monitoring and threat remediation to alleviate the burden on the security team and mitigate human errors with the minimum costs possible. The proposed model, known as the Cost-Effective Zero-Trust (CEZT), demonstrated an enhancement in the security score of a cloud infrastructure on the AWS platform across five different security standards through a zero-trust approach. For the CIS AWS Foundations Benchmark v1.2.0, the security score improved from 23% prior to implementing CEZT to 26% post-implementation, marking the smallest improvement. Conversely, the most notable improvement was observed with the AWS Foundational Security Best Practices v1.0.0 standard, where the security score increased from 44% to 77%. Additionally, the CEZT model contributes to cost reduction for organizations by utilizing free or less expensive security tools. Furthermore, it also provides automation by enforcing security measures on resources automatically, thereby alleviating the burden on the security team and minimizing human errors. Through the use of zero-trust principles, CEZT also considers the possibility of insider threats, which is one thing the traditional perimetery-based security approaches lack.

Keywords: Security, Automation, AWS, Cloud, ZT, Zero Trust, Cost, Insider Attack, SDN, Micro-Segmentation, Least Privilege Principle, Misconfiguration, Certificate Authority, Intrusion, IDS, Trust

List of Abbreviations

CIS	Centre for Internet Security
POODL	Padding Oracle on Downgraded Legacy Encryption
HTTPS	Hypertext Transfer Protocol Secure
DDOS	Distributed Denial of Service
GRC	Governance, Risk, and Compliance
VDI	Virtual Desktop Infrastructure
MITC	Man in the Cloud
FTP	File Transfer Protocol
SSM	(AWS) Systems Manager
AWS	Amazon Web Services
XSS	Cross-Site Scripting Attack
MITM	Man in the Middle
HTTP	Hypertext Transfer Protocol
EKS	Elastic Kubernetes Service
ZTA	Zero Trust Architecture
PE	Policy Engine
CVE	Common Vulnerabilities and Exposures
JSSE	Java Secure Socket Extension
DNS	Domain Name System
APN	AWS partner network
SDN	Software-Defined Networking
SUBA	Security User Behaviour Analytics
GCP	Google Cloud Platform
SSL	Secure Sockets Layer
ID	Identification
SCP	Situational Crime Prevention
SSH	Secure Shell
AES	Advanced Encryption Standard
PCI	Payment Card Industry
S3	(AWS) Simple Storage Service
CRIME	Compression Ratio Info Leak Mass Exploitatio
MTTR	Mean Time to Remediate
EC2	(AWS) Elastic Compute Cloud
RDS	(AWS) Rational Database Service

VPN	Virtual Private Network
NGFW	Next Generation Firewall
KMS	(AWS) Key Management Service
IDPS	Intrusion Detection and Prevention System
NIST	National Institute of Standards and Technology
IAM	Identity and Access Management
ASG	Auto-Scaling Group
SNS	(AWS) Simple Notification Service
IEEE	Institute of Electrical and Electronics Engineers
FIN	Finnish (TCP flag)
ZTX	(Forrester) Zero Trust Extended
WPA	Wi-Fi Protected Access
RSA	Rivest-Shamir-Adleman
SPDY	Speedy (a Google Protocol)
SDP	Software-Defined Perimeter
API	Application Programming Interface
CEZT	Cost-Effective Zero-Trust
PRNGS	Pseudo Random Number Generator Sequences
TA	Trust Algorithm
CLI	Command Line Interface
PA	Policy Administrator
VPC	(AWS) Virtual Private Cloud
CA	Certificate Authority
IPS	Intrusion Prevention System
NAT	Network Address Translation
EFB	Electronic Flight Bad
FREAK	Factoring RSA Export keys Attack
AZ	Availability Zone
ASR	(AWS) Automated Security Response
IDS	Intrusion Detection System
TLS	Transport Layer Security
PII	Personally Identifiable Information
RFC	Request for Comments
JSON	JavaScript Object Notation
ASFF	AWS Security Finding Format
3DES	Triple Data Encryption Standard

VM	Virtual Machine
ECS	Elastic Container Service
VLAN	Virtual Local Area Network
ARN	Amazon Resource Name
AMI	Amazon Machine Image
WAF	Web Application Firewall
PEP	Policy Enforcement Points
ACM	AWS Certificate Manager
RC4	Rivest Cipher 4
CBC	Cipher Block Chaining
BREACH	Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext
TBAC	Trust-Based Access Control
MCAP	Microcore and Perimeter
IG	Internet Gateway
SIEM	Security Information and Event Management
OSI	Open Systems Interconnection
SOAR	Security Orchestration, Automation, and Response
MFA	Multi-factor Authentication
DB	Database
ML	Machine Learning
SSRF	Server-Side Request Forgery

List of Figures

Figure 1. Device Agent/Gateway Model (Rose et al., 2020, Figure 3). 19
Figure 2. Enclave Gateway Model (Rose et al., 2020, Figure 4). 20
Figure 3. Resource Portal Model (Rose et al., 2020, Figure 5). 21
Figure 4. Beyondcorp Components and Access Flow (Ward & Beyer, 2014, Figure 1). 22
Figure 5. Zero Trust Network Architecture Segmentation (Kindervag, 2010, Figure 10). 23
Figure 6. The Extended ZT's Components (Cunningham, 2018, Figure 1). 25
Figure 7. ZT Model Using NSX (K. Kumar, 2017, Figure 1). 26
Figure 8. Workflow of the Architecture of SDP (Cloud Security Alliance, 2013, Figure 1). 27
Figure 9. CEZT Basic Architecture. Diagram Created by AWS Cloudformation Designer 38
Figure 10. Sample Code for Creating Subnets in AWS CLI 40
Figure 11. Generating an Elastic/Static IP 41
Figure 12. Creating a NAT Gateway for the Public Subnet through AWS CLI. 41
Figure 13. Associating a NAT Gateway to the Private Subnet through AWS CLI 42
Figure 14. SSM Runbook Diagram. Diagram Generated by AWS SSM. 46
Figure 15. Diagram of Custom SSM Runbook For S3 Buckets. Figure Generated by AWS SSM 49
Figure 16. Diagram of SSM Runbook for RDS ZT Enforcement. Figure Generated By AWS SSM 51
Figure 17. Runbook Diagram for ZT Enforcement on Users. Figure Generated By AWS SSM. 53
Figure 18. Overall Security Score Measured by Security Hub with the CEZT Disabled. 56
Figure 19. All the Risks Grouped by Severity, Measured by Security Hub Before Enabling the CEZT. 57
Figure 20. Most Common Threat Before Enabling the CEZT. 57
Figure 21. Security Scores Measured by Security Hub with the CEZT in Effect. 58
Figure 22. Threats Grouped by Severity with the CEZT in Effect. 58

List of Tables

Table 1. SSL/TLS Attack Prevention 14
Table 2. Comparison of Security Scores Before and After Enabling the CEZT. 59
Table 3. Comparison of Security Threats Findings Grouped by Severity Before and After Enabling the CEZT. 59

Table of Contents

List of Abbreviations **3**

List of Figures **6**

List of Tables **7**

Acknowledgements **11**

1. Introduction **1**

1.1 Research Problem **1**

1.2 Research Questions **2**

1.3 Research Objectives **3**

1.4 Organization of Thesis **3**

2 Zero Trust in Cloud Computing: Enhancing Modern Infrastructure Security **5**

2.1 TLS/SSL Protocols **5**

2.2 Cloud Computing **6**

2.3 Cloud Security **7**

2.4 Security Attack Types **9**

2.4.1 SSL/TLS attacks 9

2.4.2 DoS and DDoS 11

2.4.3 MITM & MITC Attacks 12

2.4.4 Application Layer Attacks 13

2.4.5 Brute-Force Attacks 14

2.5 Some Prevention Methods **14**

2.6 Insider Threat **15**

2.7 Zero-Trust Architecture **16**

2.7.1 More on Trust Algorithm 17

2.7.2 NIST ZT Models 18

2.7.3 Some Real-World ZT Solutions 21

2.8 Discussion **27**

3 Methodology and Design **29**

3.1 Partner Solutions **29**

3.2 AWS Services/Solutions 29
3.2.1 Storage and Server Services 29
3.2.2 Tools used for Automation 30
3.2.3 Security Services 30
4 Zero trust Architecture Design for Cloud Computing 35
4.1 Segmentation of the Network 39
4.1.1 Setting up the Required Policy-Enforcing Resources in the Public Subnet 43
4.2 ZT Policies Enforcement on EC2 44
4.3 ZT Policies Enforcement on S3 48
4.4 ZT Policies Enforcement on RDS 50
4.5 ZT Policies Enforcement on Users 52
5 Analysis and Further Discussion 55
6 Conclusion 61
6.1 Evaluation of Research Questions and Their Resolutions 61
6.2 Limitations and Future Work 62
References 63

Acknowledgements

I extend my heartfelt gratitude to my esteemed supervisors, Tahir Mohammad and Antti Hakkala, for their invaluable guidance and unwavering support throughout the duration of this thesis. Their insightful comments and engaging discussions have significantly contributed to the enhancement of my writing, the incorporation of crucial information, and the refinement of my work. Their mentorship has been instrumental in the successful completion of this research endeavor.

Furthermore, I would like to express my sincere appreciation to the University of Turku and all the teachers who have played a pivotal role in enriching my knowledge and honing my skills during my master's program. Their dedication to education has not only shaped me into a more knowledgeable and skillful individual but has also contributed to my personal growth and development.

1. Introduction

The landscape of information technology has undergone a significant transformation with the widespread adoption of cloud computing. As organizations increasingly migrate their operations to cloud environments, there is an escalating demand for robust security measures. Several approaches have been proposed to enhance security in the rapidly evolving digital landscape, with cloud computing being one of the main concerns (Yan & Wang, 2020). Zero Trust is a promising solution, emphasizing its foundation on the principle of least privilege. This approach treats all resources as untrusted, regardless of their location within or outside the network, and incorporates robust logging practices to facilitate forensic investigations. This thesis investigates the application of Zero Trust Architecture (ZTA) as a strategic framework to enhance security in cloud environments.

Zero Trust Architecture challenges the conventional perimeter-based network security model by postulating that threats may already exist within the network, encompassing both insider threats and external threats. This paradigm necessitates the continuous verification of every user and device, irrespective of their location or network connection. The work presented in this thesis comprehensively explores the principles and components of ZTA, elucidating its capacity to reinforce cloud infrastructures against dynamic cyber threats through automated processes.

1.1 Research Problem

Security is one of the main concerns of organizations due to increased digitalization. With businesses increasing there online presence to offer a wide array of digital services, the on-premise solution proves costly and does not scale. Due to this, cloud computing has increasingly become a major part of various businesses to host their service and applications. Migration to the cloud has changed security practices, and due to this change and newly-invented attack types, new security approaches should be considered. The vastness of cloud infrastructure and the sheer number of resources often make it difficult for security teams to safeguard all components effectively. Repetitive tasks further increase the risk of human error. Cloud service providers, such as AWS, offer numerous security solutions; however, these are typically expensive, as they are already-managed services. Managed services are those directly administered by AWS, whereas unmanaged services require user management. Although managed services offer benefits like ease of use by eliminating complex configurations, they also have some disadvantages. These managed services are designed to meet the needs of the majority, making customization difficult when specific requirements differ. Managed services, under the purview of AWS, are directly administered by the cloud provider, whereas

unmanaged services necessitate user management. While the utilization of managed services confers certain advantages, such as streamlining complex configurations, it also entails certain drawbacks:

Lack of Flexibility: Managed services are pre-configured to suit prevalent use cases and may lack the adaptability required to address unique or industry-specific security requisites. For instance, a healthcare organization might necessitate specific compliance controls distinct from those of a financial institution. Conversely, an enterprise may need to adhere to its own security standards rather than industry-defined protocols. Consider, for example, a gaming company facing a higher susceptibility to DDoS attacks as opposed to phishing attempts. As such, prioritizing protection against DDoS incidents becomes paramount. Conversely, this priority might be inverted for another entity, such as a credit reporting firm, where safeguarding customers' sensitive data, such as Social Security Numbers, supersedes temporary service unavailability.

Integration Challenges: The transition from on-premise systems to cloud-based infrastructure may pose challenges for certain organizations, particularly if their existing configurations are incompatible with the pre-established services provided by cloud vendors.

Cost Implications: Managed security services often entail exorbitant costs, especially for small to medium-sized enterprises (SMEs) or start-ups, as cloud providers may levy charges not only for resource allocation but also for management services. Consequently, while these organizations may necessitate robust security measures, the steep expenses associated with comprehensive managed services may be beyond their financial means.

This thesis aims to enhance cloud security by leveraging automation, ensuring cost-effectiveness, and implementing stringent security policies while limiting the use of managed services, suitable for critical infrastructures and start-ups. To maximize the security and avoid insider threats emphasized in section 2, the proposed model is based on a zero trust approach.

1.2 Research Questions

The primary focus of this investigation revolves around the following research questions:

1. How can enterprises enhance the security of their cloud infrastructure to meet the stringent demands of critical infrastructure while also integrating automated security protocols to alleviate pressure on security teams and minimize the risk of human error?
2. How can these security strategies be deployed in a manner that is both cost-effective and adaptable to the diverse needs and resources of organizations?

3. Does adopting a zero-trust approach significantly improve security or impose unnecessary burdens by enforcing overly strict policies without providing additional security?

1.3 Research Objectives

To address these challenges and achieve our research objectives, we propose the creation of Cost-Effective Zero Trust (CEZT), a tailored automated security solution designed specifically for Amazon Web Services (AWS) environments. CEZT is designed to merge the principles of ZTA with sophisticated automation techniques to establish a robust and adaptable security framework. This solution aims to bolster the security stance of enterprises operating on AWS, safeguarding critical infrastructure assets while simplifying security management processes and optimizing resource allocation. The decision to choose AWS over other cloud platforms is based on its status as one of the most widely used cloud platforms globally. Additionally, AWS aims to expand its market presence in Europe, which is currently dominated by Microsoft Azure, with the launch of The AWS European Sovereign Cloud in Germany by the end of 2025 (Peterson, 2024).

This thesis outlines the research objectives and methodology behind the development and deployment of CEZT. In particular, the objectives of the thesis are as follows:

To explore the rationale for adopting a Zero Trust approach and highlight the benefits of automation in enhancing security effectiveness and operational efficiency.

To examine cost optimization strategies within cloud security, emphasizing the need to strike a balance between security imperatives and financial constraints.

To contribute to advancing cloud security practices by providing practical insights and solutions, address the evolving threat landscape, and empower organizations to safeguard their critical assets in an increasingly interconnected and digital environment.

1.4 Organization of Thesis

The thesis has been structured into separate chapters as follows. Chapter 2, the Introduction, serves to acquaint readers with pertinent terminology, including contemporary research on cloud computing and its security implications, relevant protocols, and diverse attack vectors targeted for mitigation. Real-world instances of these attacks are outlined, alongside an exploration of how adopting a zero-trust approach can forestall such threats. Additionally, this section delves into zero-trust architecture and foundational models. Chapter 3 will delve into the technologies underpinning our proposed

methodology. Chapter 4 elucidates the methodology and design of the CEZT framework. It explains the practical implementation of CEZT on the AWS platform and its efficacy in mitigating security breaches. Chapter 5, the Analysis and Discussion section, will assess the CEZT's effectiveness through testing on an intentionally weakened AWS infrastructure, revealing notable reductions in security vulnerabilities post-implementation. In the final substantive section, Chapter 6, the research culminates in a conclusive summary.

2 Zero Trust in Cloud Computing: Enhancing Modern Infrastructure Security

In this Chapter, the current state-of-the-art research about Zero Trust Architecture will be discussed, and different frameworks proposed by various organizations will be evaluated, followed by a comparison of their advantages and disadvantages and choosing one of them to deploy on AWS. Furthermore, the current research on ZTA deployments in cloud environments will also be studied. The principle of ZTA is that threats originate from inside the network in addition to external threats. ZTA can be replaced with traditional location-based network security policies that only consider external threats. The problem with these security frameworks lies in the fact that once the attacker has access to the network through various ways, such as a backdoor or a zero-day vulnerability, they no longer need to go through the process of identification. It makes the attack much easier once the attacker is inside the network.

Another problem is malicious employees, also referred to as insider attackers. According to the National Institute of Standards and Technology (NIST), an "insider" refers to an individual or a collective entity possessing authorized access to an organization's information systems, facilities, or data (NIST, 2020). This category encompasses employees, contractors, or other individuals who have been officially granted legitimate access privileges. Based on a survey from over 326 cybersecurity professionals (Gurucul, 2023), 53% of the survey respondents believed that detecting insider attacks is more difficult on cloud platforms than on-premises environments. Amongst the respondents, 74% reported an increasing frequency of insider threats, raising concern over cloud solutions as more organizations migrate to the cloud environment.

2.1 TLS/SSL Protocols

Various attack types commonly employed against non-cloud infrastructures can also target cloud-based solutions. Despite the widespread use of TLS (Transport Layer Security) as a security defence mechanism in cloud environments, vulnerabilities may still exist that could compromise the integrity of the cloud infrastructure, which can also result in other forms of attacks, e.g. MITM attacks (Abusaimeh, 2020). One potential avenue for compromise is through misconfiguration of the Certificate Authority (CA) server, which can occur due to a variety of factors. TLS stands as an evolution of the Secure Sockets Layer (SSL), conceived as its successor to address inherent vulnerabilities. According to the RFC (Rescorla, 2018), the TLS handshake unfolds through the following steps:

Step 1: The client initiates communication by dispatching a hello message to the server, encapsulating pertinent details such as supported TLS versions, cipher suites, and additional parameters.

Step 2: In response, the server provides a reply inclusive of the highest TLS version and cipher suites compatible with the client's system.

Step 3: Subsequently, the server crafts and authenticates a certificate containing its public key, which is then transmitted to the client for authentication purposes.

Step 4: Upon receipt, the client generates a premaster secret, encrypts it utilizing the server's public key, and forwards it to the server.

Step 5: Following this exchange, both parties derive a master secret from the premaster secret, serving as a foundation for generating encryption keys.

Step 6: Upon completion of the key exchange, both entities transmit a concluding message to signify the completion of the handshake process. With the handshake successfully concluded, subsequent data transmission between the two parties proceeds in an encrypted format, facilitated by the keys generated during the 4^{th} step.

2.2 Cloud Computing

As the amount of data grows and the complexity of managing traditional on-premise data centers becomes more challenging, many organizations are opting to migrate their infrastructure to cloud-based solutions. By transitioning from on-premise to cloud infrastructure, organizations and users can access a broad array of managed resources, leading to potential cost and time savings (Sunyaev & Sunyaev, 2020). This approach allows users to concentrate primarily on developing their services, with less concern for the underlying infrastructure, as it is provided and maintained by the cloud provider. Additionally, depending on the type of cloud environment, managed and/or unmanaged services are available. The degree of control that users can exert over the underlying resources varies depending on the service model, which generally falls into the following four main categories.

Software as a Service (SaaS): Rani & Ranjan (2014) describe this model as entailing software utilization via the cloud, accessible to clients without the necessity of installation. Users can procure and utilize the software directly via internet connectivity, typically through web browsers. Each client retains the ability to configure settings and allocate resources according to their specific requirements.

However, in this paradigm, users are not granted access to the underlying infrastructure; their interaction is confined solely to the software application, e.g. Google Drive.

Platform as a Service (PaaS): In contrast to Software as a Service (SaaS), PaaS affords users expanded capabilities by enabling them to develop custom applications within the cloud environment. Importantly, users are spared the necessity of detailed knowledge about the underlying infrastructure supporting their applications. Essentially, in this model, users craft and deploy their applications utilizing the cloud provider's infrastructure resources (Dillon et al., 2010), e.g., Azure Blob Storage.

Infrastructure as a Service (IaaS): Analogous to Platform as a Service (PaaS), this model empowers users to develop their applications (Mohammed & Zeebaree, 2021). Nevertheless, users enjoy heightened access to the underlying infrastructure. They possess the capability to virtually and occasionally physically reserve and utilize certain resources provided by the cloud service provider. Moreover, users are permitted to modify the configuration of the infrastructure to a certain extent, granting them greater flexibility compared to SaaS and PaaS models. For instance, within AWS, users can virtually reserve a portion of the infrastructure as a virtual private cloud (VPC).

Container as a Service (CaaS): Alouffi et al. (2021) characterize this model as leveraging containerization techniques to render applications independent of environment specifications, exemplified by platforms such as Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS).

2.3 Cloud Security

In the field of information security, cloud security has become a prominent area of discussion and concern. In contrast to established security domains, such as network or application security, cloud security represents a comparatively developing area of research. Its significance has garnered increasing attention from both academic and industrial quarters. A comprehensive analysis of scholarly literature spanning the decade from 2010 to 2020, conducted by Alouffi et al. (2021) identified seven principal security threats pertinent to cloud computing. Notably, among the topics extensively deliberated upon in the reviewed literature are data tampering and leakage. Additionally, the study underscored the pivotal role of intrusion detection systems (IDS) within cloud environments. For this purpose our proposed model, the CEZT, will use Snort as the IDS.

Another scholar posited that security is a primary factor influencing the apprehension surrounding the adoption of cloud services. Vidal & Choo (2018) approached cloud security threats through the lens of criminology, offering a conceptual model rooted in the principles of situational crime

prevention (SCP). This model aims to diminish the appeal of cloud-related crimes and dissuade potential offenders.

In terms of services, cloud models are classified into 4 main categories, as discussed in the previous section. The possibility of enhancing security is maximum in the IaaS model amongst all the models as the user has the most access to the underlying infrastructure. This can also act as a double-edged sword, which can be a disadvantage to the user if they do not have a suitable security background and skills. This thesis aims to develop a ZT model for IaaS, specifically AWS. According to the shared responsibility model (Amazon Web Services, 2006), AWS is responsible for the security of the cloud, while the user is responsible for the security in the cloud.

Numerous cyberattacks have targeted cloud-based infrastructures. in this section three recent attacks on cloud enviromnets have been emphasized. The following attacks happened due to insider threat and human error.

Case 1 – Capital One 2019 Attack

One notable example is the 2019 Capital One breach. In this incident, the sensitive data of approximately 106 million users of Capital One, a major U.S. bank, was compromised due to a misconfiguration leading to an insider attack. The cloud-based nature of the infrastructure raised concerns about the security and reliability of cloud environments.

Paige Thompson, a former AWS employee, exploited a misconfigured Web Application Firewall (WAF) within the AWS environment. This misconfiguration allowed Thompson to perform a Server-Side Request Forgery (SSRF) attack on the metadata service, enabling her to obtain security credentials. She then used these credentials to access an AWS S3 bucket belonging to Capital One, which contained sensitive user information, including names, credit scores, addresses, and social security numbers (U.S. Department of Justice, 2022).

After Capital One was informed about the incident, the company sought assistance from professional cybersecurity experts both within and outside the organization, as well as law enforcement, to assess the severity of the attack and the extent of the compromised resources. Notably, the misconfiguration originated from AWS and not from Capital One, classifying this breach as an insider attack. Traditional perimeter-based security controls are ineffective against insider attacks since, once inside the network, attackers are considered trusted entities. A zero trust approach could prevent such attack by considering all the users, even AWS employees, untrusted entities.

Case 2 – Equifax 2017 Data Breach

In 2017, the sensitive data of approximately 147 million users of Equifax, a credit reporting firm, was exposed due to a vulnerability in Apache Struts. This breach serves as an example of human error and negligence, as a security update to patch the vulnerability was already available but was ignored by the company. The use of cloud-based data infrastructure by Equifax raised concerns about the security of cloud environments (U.S. Federal Trade Commission, 2022).

In this instance, the responsibility lay with the company rather than the cloud provider. The attack highlighted the critical importance of regular patching and updates, emphasizing the role of human error in such incidents and suggesting how automation could prevent such disasters. Automated update could avoid such disaster.

Case 3 – Pegasus 2022 Data Breach

A recent incident is the data breach of Pegasus Airlines caused due to misconfiguration of the cloud platform of the company by a privileged user, leading to exposure of around 6.5 terabytes of sensitive data, encompassing staff's PII, flight data and source code from Electronic Flight Bad (EFB) software (SafetyDetectives Cybersecurity Team, 2022). The data was leaked from a compromised AWS S3 bucket, raising concerns about the reliability of cloud environments. Automated configuration could avoid this incident.

2.4 Security Attack Types

In this section, various cyber attack types and how they can be a threat to cloud environments will be discussed. Depending on the network topology and architecture, these attacks can be a concern for cloud.

2.4.1 SSL/TLS attacks

Despite the age of TLS attacks and mitigation strategies, they continue to occur, often due to misconceptions among users who believe that the mere use of TLS certificates ensures the security of the transport layer. However, this assumption is flawed, particularly in instances where the CA server is misconfigured.

FREAK: Factoring RSA Export keys (FREAK) also known as Server Spoof Attack against Browsers, this attack is possible due to the use of weak export cipher suites used by TLS, implemented within several TLS libraries including but not limited to LibReSSL, IBM JSSE, and OpenSSL (the

problematic TLS libraries were discovered in 2015, and updated library versions have patched this vulnerability to an extent, but not completely (Durumeric et al., 2015). Through MITM attacks, the attackers can make the user's browser to use weak export keys. This is possible with RSA keys with a length less than 512 bits (Extreme Networks, 2015).

Bar Mitzvah: An attack in which the assailant endeavours to uncover vulnerable keys by focusing on the initial 100 bytes of encrypted data, within which 36 bytes pertain to the SSL/TLS finished message. Since the finished message contains highly predictable content, plain finished messages are subjected to an XOR operation with encrypted finished messages to extract segments of Pseudo Random Number Generator Sequences (PRNGS). Subsequently, PRNGS that deviate from the pattern of weakly generated keys are discarded, and the remaining keys from the selected PRNGS are employed to decipher the ciphertext intercepted by the attacker, utilizing the RC4 algorithm (Roos, 1995) (Mantin, 2015).

BEAST: Found in the first version of TLS, it is a brute force attack in which the attacker tries to guess the cipher key through taking advantages of symmetric encryption and cipher block chaining (CBC) method (Newman, 2014).

CRIME: Compression Ratio Info Leak Mass Exploitation (CRIME) is a cryptographic attack that exploits vulnerabilities in the compression algorithms used in TLS (Transport Layer Security) and SPDY, a protocol developed by Google for enhancing web browsing speed. Both TLS and SPDY employ the DEFLATE compression algorithm, which removes duplicate strings during compression and encryption processes. The vulnerability arises from the way compression interacts with encryption. By sending multiple encrypted, compressed requests to a legitimate website and observing variations in the size of the HTTP responses, the attacker can deduce patterns in the encrypted data. If the size of the response is smaller than expected, it suggests that certain characters were removed through compression. Through repeated attempts, the attacker can eventually deduce the encryption key, compromising the security of the system (Sarkar & Fitzgerald, 2014) (Shaji & Subramanian, 2021).

SSL Renegotiation: This vulnerability enables attackers to inject commands into the client's requests, downgrade HTTPS connections to HTTP, inject customized responses, or execute denial-of-service (DoS) attacks (Shaji & Subramanian, 2021) (Zoller, 2011).

POODLE: The vulnerability known as Padding Oracle on Downgraded Legacy Encryption (POODLE) manifests when the browser of the target system transitions from the more secure TLS

protocol to the less secure SSL 3.0. This transition permits attackers to exploit weaknesses in the encryption scheme, potentially intercepting plaintext data. Consequently, sensitive information such as credit card details or login credentials may be compromised (Shaji & Subramanian, 2021) (Möller et al., 2014).

TLS Truncation: Smyth & Pironti (2013) explain an attack in which the attacker makes the log-out request performed by user invalid to the server through injecting RESET or TCP FIN messages prior to the request due to abnormal termination of connection, keeping the session alive without the user knowing.

BREACH: This is an extension of the CRIME attack targeting HTTPS, where the attacker leverages HTTP compression to make guesses about characters. These guesses are then reflected in the response body. To execute this attack, the attacker and victim need to be on the same network, highlighting the principle of "no one is trusted, even internal entities" in Zero Trust Architecture (Satapathy & Livingston, 2016) (Gluck et al., 2013).

RC4 BIASES: AlFardan et al. (2013) discovered the use of RC4-128 can be problematic if it uses 128-bit long keys to generate random keys, followed by generating the keys and XORing them with the plaintext to produce the ciphertext. Mantin (2015) further explains that the problem is because the keys are not random enough, resulting in the possibility of recovering some parts of plaintexts, particularly when the same plaintext is encrypted using multiple different cipher keys. Since there is a possibility of some parts of those ciphertexts to match, attackers can use statical analysis of individual locations of the ciphertexts.

In case of using a CA server to provide certificates in the cloud infrastructure, the cloud-based infrastructure can be compromised if the CA server is misconfigured. Users are advised to carefully consider the mentioned attacks and their mitigation methods, which will be explained in the next subsection, during its configuration process.

2.4.2 DoS and DDoS

Denial of Service (DoS) attacks involve the deliberate interruption of services over an extended period by obstructing legitimate traffic by inundating the target with a significant volume of malicious traffic. Distributed Denial of Service (DDoS) attacks occur when a network of compromised systems, known as a botnet, is orchestrated via a command and control (C&C) server to execute a more potent form of DoS utilizing multiple systems simultaneously. Presently, such attacks can escalate to a magnitude of approximately 1 Terabit per second of malicious traffic (Zhijun et al., 2020) (S. Kumar

& Chandavarkar, 2021). DoS and DDoS are two of the main concerns of this research. This is because this thesis approach is inspired by the concept of Software Defined Networking (SDN), and both DoS and DDoS can be used to flood either the control plane, data plane or the communication channel between them (Eliyan & Di Pietro, 2021). In a paper by Douligeris & Mitrokotsa (2004) it was stated that DDoS consisted 14% of attacks on cloud environments during the corresponding year.

Gupta et al. (2023) classify DDoS attack types into five distinct categories:

Spoofed Attacks: These attacks involve the use of modified packets with fake IP addresses, making it more challenging to trace the attacker.

Amplification Attacks: In this type, attackers use intermediary servers to send malicious traffic through different channels with increased intensity. This makes it more difficult to trace and mitigate the attack due to the heightened traffic volume.

Application Layer DDoS Attacks: These attacks target specific applications by exploiting vulnerabilities, such as HTTP flooding attacks.

Volumetric Attacks: These are high-volume flooding attacks, such as ICMP attacks, that overwhelm the network with a massive amount of traffic.

Protocol Attacks: Attackers exploit flaws in protocols, such as by initiating multiple incomplete TCP handshakes, causing sockets to become flooded as they wait for the handshake process to complete.

Exploiting DoS/DDoS attacks, resources hosted on cloud environments can become inaccessible for a period, disrupting services and damaging the company's reputation. For instance, AWS EC2 instances and S3 buckets are potential targets of such attacks, especially if adequate security controls are not implemented, such as unnecessarily allowing traffic from all sources.

2.4.3 MITM & MITC Attacks

Al-Shareeda et al. (2020) describe Man-in-the-Middle Attack (MITM) as a form of intrusion where a malicious actor positions themselves between two victims, potentially intercepting packets containing sensitive information. This attack manifests in two distinct forms; passive and active. In the passive variant, known as sniffing, the attacker solely eavesdrops on the target network without actively altering data. Conversely, the active iteration involves the attacker engaging in manipulative actions including delaying, dropping, or tampering with messages transmitted between the victims.

According to a research conducted by Goyal et al. (2016) an element contributing to the severity of this threat lies in the abundance of accessible tools, such as Mirage, BTLeJuice, and GATTack, alongside the possibility of performing the attack using available inexpensive hardware.

Numerous scenarios of MITM attacks can occur in cloud environments. For example, when connecting to an AWS EC2 instance through the HTTPS protocol, an attacker can employ SSL Stripping attacks, such as the previously mentioned POODLE attack, to downgrade the connection from HTTPS to HTTP, thereby exploiting the unencrypted data.

The Man in the Cloud (MITC) attack, inspired by the MITM concept, involves exploiting cloud file synchronization services such as AWS Cloud Sync, AWS DataSync, and AWS S3 Sync. According to Popli (2019) this can happen due to the predominant utilization of synchronization tokens for application authentication, rendering them ineffective in distinguishing malicious traffic from normal activity.

2.4.4 Application Layer Attacks

The application layer, positioned as the final layer in the Open Systems Interconnection (OSI) model, directly interfaces with end users, and thus application layer threats are one of the main concerns when securing the cloud (Tewari & Gupta, 2020). Within this layer, a spectrum of attacks exists. Mishra & Pandya (2021) categorize these attacks into five primary classifications: sniffing attacks, access control attacks, service interruption attacks, malicious code injection attacks, reprogram attacks, and cross-site scripting attacks. Considering the attack types, different authors show different opinions regarding the level of importance. For example, while Ahlawat et al. (2020) identify access control attacks as a major concern, particularly due to the allocation of unique privileges to certain users, granting them access to a substantial portion, if not the entirety, of the enterprise infrastructure, leading to these privileges becoming susceptible to compromise and potentially resulting in unauthorized access to sensitive systems and data, on the other hand, Prabadevi's & Jeyanthi's concern (2018) is mainly sniffing attacks due to the availability of various tools used for the attack and one of the main reasons of DoS/DDoS attacks.

Various factors can lead to successful application attacks on cloud environments, including accepting raw input from users without proper filtration. This oversight can result in multiple attack possibilities, such as injection attacks.

2.4.5 Brute-Force Attacks

A brute-force attack is characterized by the assailant's systematic attempt to uncover sensitive information, including credentials, by exhaustively testing all conceivable combinations. This form of attack, deemed one of the earliest methods employed by malicious actors, can be executed across diverse services, such as SSH and FTP, through various available automated tools, including but not limited to Hydra, John the Ripper, and Aircrack-ng, as noted by Hossain et al. (2020).

Misconfiguration is a significant factor that can render cloud-based resources vulnerable to brute force attacks. For example, if an attacker is allowed to send a large volume of traffic to guess login credentials without any protective policies in place, such as enforcing a delay between login attempts or limiting the number of invalid credential inputs, it can compromise the security of the cloud environment.

2.5 Some Prevention Methods

To protect the enterprise against attacks on SSL/TLS, Satapathy & Livingston (2016) propose the following mitigation methods:

Table 1. SSL/TLS Attack Prevention

Attack Method	Mitigation
FREAK	Utalizing higher versions of cipher and TLS, Disabling the export cipher suite in browsers to avoid Logjam attack (Kerner, 2015).
Bar Mitzvah	RC4 needs to be disabled.
BEAST	opt using AES 256, 3DES or RC4 (using of RC4 can avoid BEAST attack but it is still prone to Bar Mitzvah and RC4 BIASES attacks).
CRIME	TLS Compression needs to be disabled.
SSL Renegotiation	The mitigation strategy involves either disabling renegotiation at the server level or implementing prior handshake verification by both the server and client.
POODLE	SSL version 3 needs to be disabled.
TLS Truncation	opt using a centralized authentication method and chain sign outs.
BREACH	HTTP compression needs to be disabled.
RC4 BIASES	RC4 needs to be disabled.

Saharan & Gupta (2021) categorize DDoS protection methods into detection, mitigation, and prevention strategies. Furthermore, within the prevention category, they distinguish between ideal prevention, where the network is theoretically never compromised, and true prevention, which involves the network recovering after attacks within acceptable intervals.

For brute force mitigarion, Nadeem et al. (2021) conclude that employing an IDS can significantly enhance defense against brute-force attacks on cloud platforms. They emphasize that the effectiveness of this approach hinges on the capability of the detection algorithm to accurately identify and respond to such threats.

The effectiveness of zero trust architecture in mitigating insider threats and the ineffectiveness of traditional premitery-based security approaches will be discussed in the following subsection.

2.6 Insider Threat

Previously a number of real-world cloud-based attacks were mentioned. In this subsection the importance of insider attack prevention is illustrated, which is one of the main reasons that the focus of this research is on ZTA, as primetry-based security approaches do not consider the risk of insider threats. First two recent attacks which could not be prevented by primetry-based security approaches will be mentioned, and then the importance of ZTA and how effective it can be to prevent such attacks will be discussed.

In May 2022, a senior research scientist employed at Yahoo, Qian Sang, undertook the downloading of approximately 570,000 pages of crucial information pertaining to Yahoo, subsequent to receiving a job offer from a rival company. This compromised data encompassed significant details, including source code, backend advertising architecture, ad-placement controlling algorithms, and pertinent earnings tracking information related to the Yahoo Demand Side Platform. Additionally, the acquired information included insights into the strategic aspects underlying Yahoo's backend advertising technology (Barnett, 2022). Another example is the 2023 data breach of Mailchimp caused by the compromise of credential of at least one employee through phishing attacks, resulting in the compromise of more than 132 Mailchimp user accounts (Rowe, 2023).

Additionally, according to two reports from Ponemon Institute, (Ponemon Institute, 2020) (Ponemon Institute, 2022), the total average cost of insider-originated incidents surged from $11.45 million in 2019 to $15.38 million in 2021 and the frequency of incidents spiked by 47% from 2018 to 2020. In the 2022 report, Ponemon classifies insiders into three categories: Negligent insiders (e.g. the

mentioned Pegasus Airline's incident), criminal or malicious insiders (e.g. the mentioned Yahoo's incident), and credential thieves (e.g. the mentioned Mailchimp's incident).

He et al. (2022) emphasize the importance of ZTA due to the downsides of perimeter-based security architecture. The authors describe perimeter-based security architecture as dividing the network into internal and external parts, separated with firewalls, IDSs and IPSs. An entity in a perimeter-based security architecture is either trusted or untrusted based on their location. If the entity is inside the internal network, then it is trusted, otherwise it is considered untrusted and needs to authenticate to reach the network. The authors explain this kind of architecture can be problematic because if a malicious entity gains access to the internal network, they can keep attacking the network for a long time since in a perimeter-based security architecture when an untrusted entity authenticates and becomes trusted, they can keep the session.

2.7 Zero-Trust Architecture

To tackle the security issues mentioned in the previous section, Kindervag (2010) proposed a new type of security architecture different from perimeter-based architecture. Based on what the author proposed, ZTA assumes an entity is always considered untrusted; thus, authentication and authorization are required for an entity to establish a connection with enterprise resources. Later various companies and organizations proposed their own ZTA models. According to Rose et al. (2020), in ZTA a trusted status is granted to an entity based on the following factors: identity authentication (used to realize the identification of an entity of ZTA), access control (for maintaining secure and efficient access of ZTA entities to the enterprise resources), trust evaluation algorithms (to facilitate the determination of the trust level associated with the ZTA entity, serving as the primary credential for both identity authentication and access control purposes), and micro-segmentation of the network and the segment where the entity is located.

The authors further describe the main components in their ZT approach:

Policy Engine (PE): This component assumes the pivotal role of making the final determination regarding access authorization for a specific subject seeking access to a particular resource. It utilizes enterprise policies in conjunction with input from external sources, employing these inputs as parameters for a trust algorithm.

Trust Algorithm (TA): PE's logic on deciding whether to grant access to a resources or not is based on an algorithm known as TA.

Policy Administrator (PA): This module assumes the role of initiating or terminating the communication pathway linking a subject with a resource by issuing commands to pertinent Policy Enforcement Points (PEPs). Additionally, it is responsible for generating any session-specific authentication tokens or credentials essential for a client to access an enterprise resource. Intricately interconnected with the PE component, the PA relies on the decisions made by PE to ultimately authorize or reject a session. Upon successful authentication or denial of a session, or if a prior approval is revoked, the PA sends a signal to the PEP to either establish or terminate the connection between the client and the designated resource.

Policy enforcement point (PEP): This system is tasked with facilitating, overseeing, and ultimately discontinuing connections between a client and an enterprise resource. The PEP engages in communication with the Policy Administration (PA), forwarding requests, and/or receiving policy updates from the PA. While constituting a single logical component within the ZTA, it may be decomposed into two distinct elements: one operating on the client side (e.g., an agent on a laptop) and the other on the resource side (e.g., data resources behind a gateway). Alternatively, a singular portal component may be employed, assuming the role of a gatekeeper for communication pathways. Beyond the PEP lies the trust zone housing the enterprise resource.

2.7.1 More on Trust Algorithm

There are various proposed techniques for designing a TA proposed by different authors. For example, Brosso et al. (2010) proposed a solution for continuous authentication, leveraging user behaviour and employing a neuro-fuzzy logic technique. This method entails the ongoing update of user behaviour data within a database, facilitating more accurate measurement of trust levels over time. Later, Chen et al. (2020) presented a methodology akin to the least privilege principle, employing a behaviour-based anomaly detection technique. This method scrutinizes user and device identities, terminal security status, and both user and system behaviours to evaluate trust levels. In accordance with this proposed approach, access authorization is contingent upon the attainment of a specific trust threshold. Likewise, Yao et al. (2020) introduce a dynamic Trust-Based Access Control (TBAC) system, where user or resource access is permitted solely upon the fulfilment of a designated behaviour trust (BT) level. This determination is informed by diverse factors, including login-related metrics such as login time and method, operational behaviour such as resource interactions or access records, and network behaviour metrics such as TCP or UDP traffic volume. Lastly, an additional scholar introduces FURZE, a framework designed by Vanickis et al. (2018) designed for risk-adaptive access control (RAdAC). This framework enables the system to enforce ZT policies by precisely and

dynamically granting or rejecting access based on operational requirements and security threats. There exist commonalities across all the three methodologies. Each of the proposed solutions regulates access contingent upon a trust level, determined through the evaluation of behaviour-based anomaly detection factors, such as login pattern, user conduct, network activities, and operational patterns.

Furthermore, the trust models within these proposed solutions draw inspiration from the principle of least privilege. Continuous authentication and monitoring should be designed to minimize inconvenience to the user, such as avoiding the need for repeated password entry for every action. A recommended approach involves initially employing Multi-Factor Authentication (MFA) during the login process, followed by continuous analysis of user behaviour over time to dynamically establish and maintain trust levels within the system (Teerakanok et al., 2021) (Embrey, 2020).

2.7.2 NIST ZT Models

Rose et al. (2020) also delineated three distinct categories of ZT deployment, each differentiated by the extent of integration severity with organizational resources:

Device Agent/Gateway-Based Deployment: In this conceptual framework, the enforcement of policies is intricately linked to both endpoints involved: the user's endpoint device, representing the enterprise system, and the targeted data resources. A specialized agent is incorporated into the enterprise system, while the data resources are positioned beyond a gateway. In accordance with established policies and identity parameters, the agent exercises discretion to either permit or deny the enterprise system's access to the designated data resource. This model's advantage lies in the potential for comprehensive control deployment based on specified policies. However, its drawback stems from the requisite high level of integration between the enterprise system and the agent, as well as the necessity for the data resources to interact extensively or be fully integrated with the gateway, thereby rendering it a challenging model to implement.

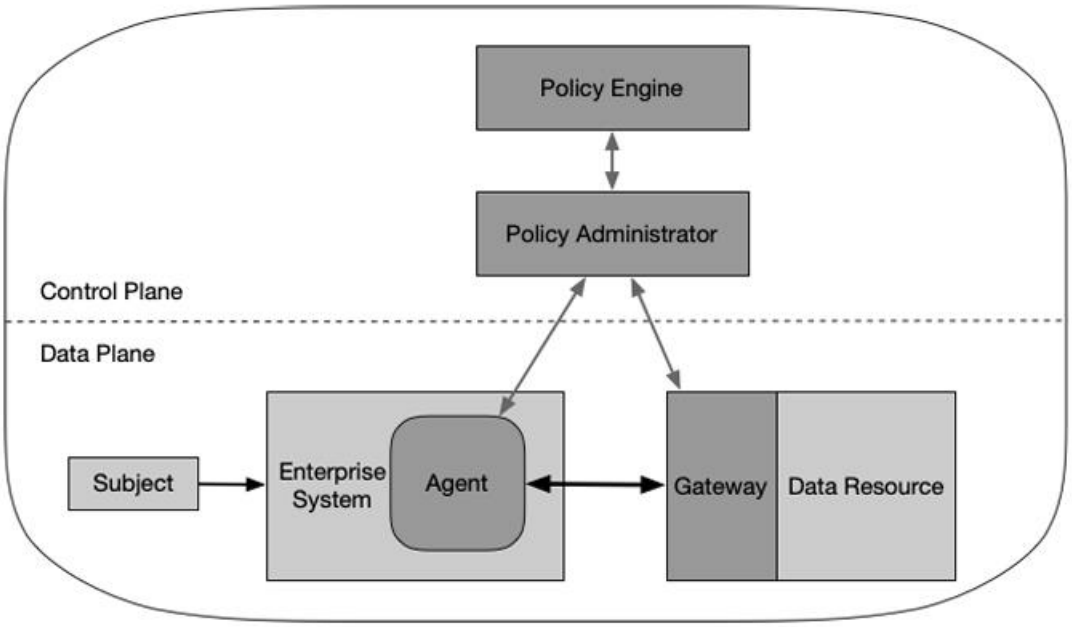

Figure 1. Device Agent/Gateway Model (Rose et al., 2020, Figure 3).

Enclave-Based Deployment: This model bears resemblance to the initial model, differing in the placement of policy enforcement, which is positioned anterior to a resource or an enclave of resources, serving as a gateway. Consequently, the necessity for extensive integration between the gateway and resources is obviated, rendering this model comparatively simpler to deploy than its precursor. However, a drawback of this model emerges from the existence of an implicit trust zone between the gateway and resources. In this configuration, the gateway lacks precise information about the specific application or resource the user intends to access, resulting in a limitation on the extent to which policy enforcement can be effectively implemented.

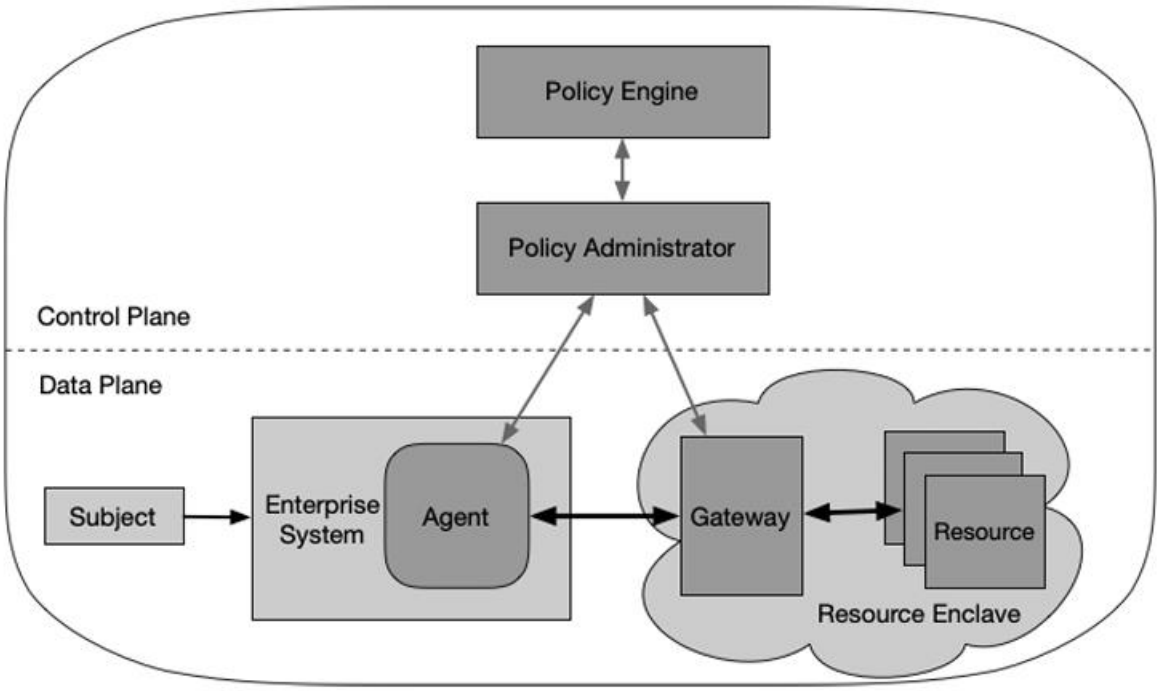

Figure 2. Enclave Gateway Model (Rose et al., 2020, Figure 4).

Resource Portal-Based Deployment: This model shares similarities with the second model, differing in that it does not necessitate the installation of an agent on endpoint devices, rendering it a more straightforward deployment. The Policy Enforcement Point (PEP) assumes a central position, functioning as a gateway. Behind this gateway, there may be one or multiple resources, such as a unit or section of an organization. A limitation of this model is its vulnerability to Distributed Denial of Service (DoS) attacks, as the PEP portal may become a potential bottleneck. Additionally, the absence of an agent on end devices implies that no information about the device is available prior to the establishment of the connection between the device and the PEP portal.

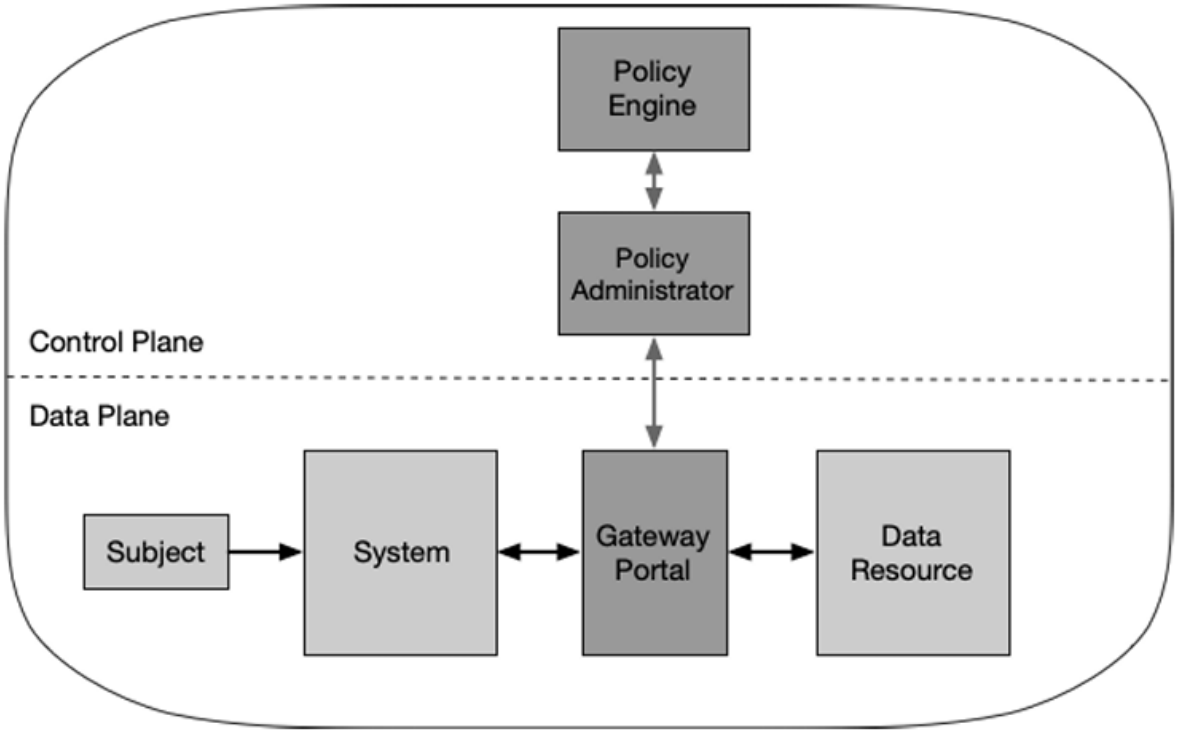

Figure 3. Resource Portal Model (Rose et al., 2020, Figure 5).

2.7.3 Some Real-World ZT Solutions

The NIST ZTA is theoretical and describes the foundations of ZTA. For real-world scenarios, there are models for ZTA developed by various organizations based on the three mentioned NIST ZT models, including:

Google's BeyondCorp model: Introduced by Ward & Beyer (2014), based on this model an access to enterprise resources is granted to an entity only by taking two steps: identity authentication followed by authorization through the access control engine inspired by the Device Agent/Gateway-based Deployment NIST model. According to this model facilitating user access to services should not be contingent upon a specific network connection. Instead, access to services is granted based on identification of the user and the device. Access to services is permissible solely when authentication, authorization, and encryption measures are concurrently in place.

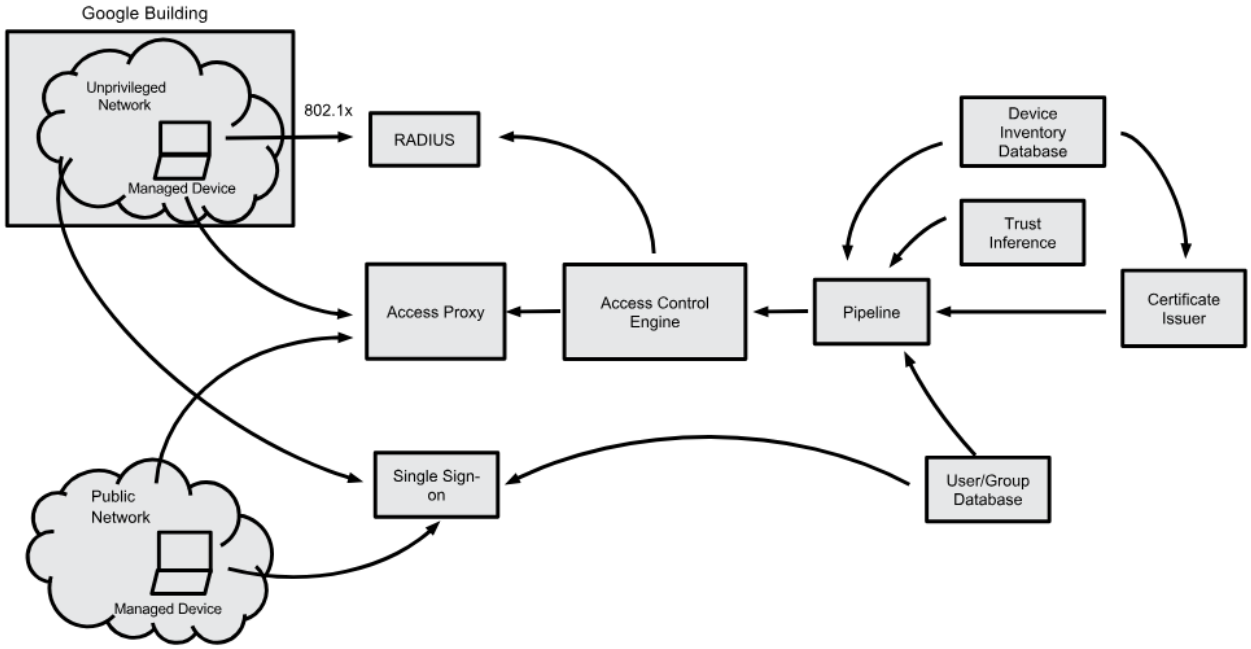

Figure 4. BeyondCorp Components and Access Flow (Ward & Beyer, 2014, Figure 1).

Next-Generation Firewall (NGFW)/Forrester Zero Trust Extended (ZTX) Model: Proposed by Kindervag (2010), within this model, enterprise resources are isolated on Very Large Area Networks (VLANs), separating the enterprise network into multiple "microcore and perimeter" (MCAP) segments. A cutting-edge firewall designated as a next-generation firewall assumes a central role within the network, functioning as a segmentation engine and conforming to the "Resource Portal" NIST deployment model previously explained.

Uttecht (2020) believes the Forrester ZT model is possibly one of the most cost-effective and easily implementable models, as this approach requires minimal adjustments to resources within the enterprise network. The primary modifications are confined to the resources within the central network that necessitate replacement or updating. Nonetheless, a drawback of this model stems from its reliance on network stack enforcement, limiting its capacity to enforce trust solely based on the protocol information available in the data packets. This granularity is notably lower compared to architectures that establish more intimate integration with endpoints and applications. Since users do not authenticate directly with the segmentation engine, policy enforcement based on user or device attributes is inherently constrained. Policy enforcement, therefore, relies on IP addresses as identity, unless the segmentation engine is integrated with additional authentication technologies. Mitigating this limitation involves enhancing integration between the firewall and other technologies such as VPNs, Identity and Access Management (IAM), or device management systems.

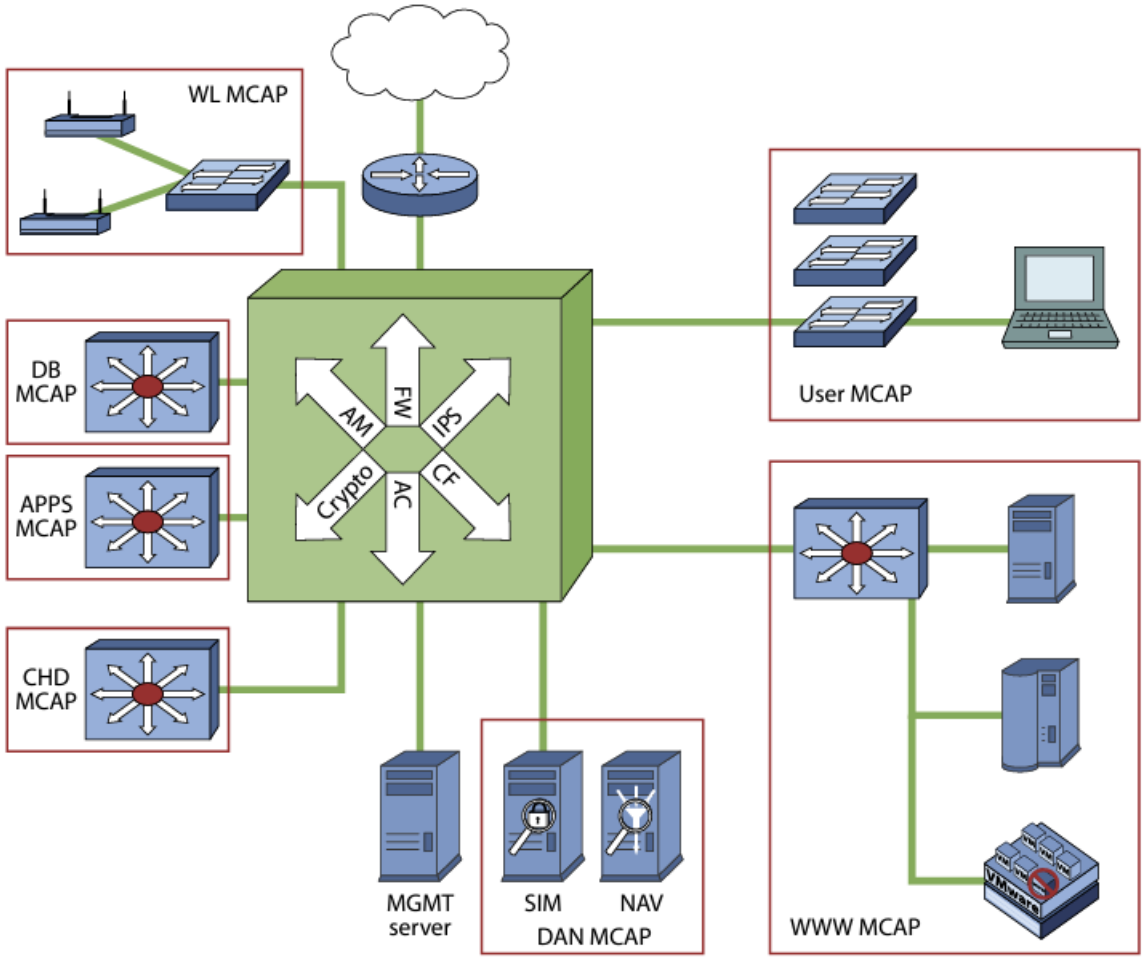

Figure 5. Zero Trust Network Architecture Segmentation (Kindervag, 2010, Figure 10).

Later, the ZT strategy delineated by Cunningham (2018) extended the previous 2010 paper by comprising seven pivotal pillars aimed at optimizing network security when appropriately implemented. These pillars encompass:

Zero Trust Data: Recognizing the paramount role of data in a ZT strategy, the author recommends comprehensive encryption for data in both transit and at rest. This safeguards data in the event of unauthorized access. Additionally, the author advocates for meticulous data management, categorization, and classification within appropriate schemas.

Zero Trust People: A fundamental pillar involves the consideration of individuals. Within a ZT framework, the author proposes restricting user access and securing their connections and interactions through authentication solutions, continuous monitoring, and governance of data/resource access and privileges. The core tenet in ZT is minimizing user access to the utmost extent and permitting access only to essentials required for organizational functions.

Zero Trust Network: To enhance network security and enable effective monitoring and governance of data, resources, and individuals, the standard ZT models advocate segmenting the network into microsegments, each isolated from others. Forrester's model employs a firewall as a gateway between resources, strictly enforcing security policies across the network.

Zero Trust Device: Another integral aspect of Forrester's ZT strategy pertains to network devices. The security team is tasked with classifying all IoT and network-enabled devices as untrusted entities, necessitating their isolation, continuous monitoring, and governance.

Zero Trust Workloads: Particularly pertinent for enterprises on public cloud platforms, this pillar emphasizes the classification of workloads, spanning the entire application stack including all front-end and back-end systems and processes used to run the organization and serve the customers, as untrusted entities. This entails stringent monitoring and governance aligned with ZT policies, mirroring the treatment of people and devices.

Visibility and Analytics: The research focal point centres on comprehending and defending against threats and attacks. Utilizing tools like security user behaviour analytics (SUBA), traditional security information management (SIM), and advanced security analytics platforms is advocated. The research underscores the employment of various tools from both AWS and external services for heightened security and automation.

Automation and Orchestration: The conclusive pillar underscores the significance of automation and orchestration across an enterprise. Various tools and technologies, including those mentioned in the previous pillar, are posited as instrumental in achieving this goal within the extended ZT strategy proposed by Forrester. Below you can find a photo illustrating the components in the extended ZT strategy:

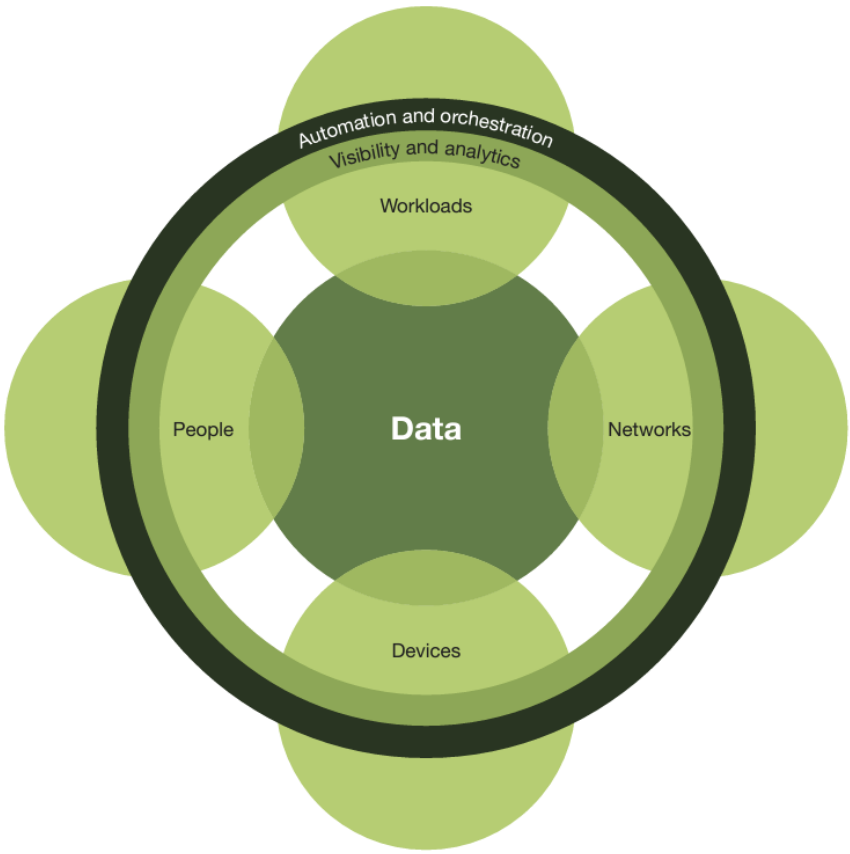

Figure 6. The Extended ZT's Components (Cunningham, 2018, Figure 1).

VMWare/NSX: Tailored for enterprises employing a virtual desktop infrastructure (VDI) and predominantly relying on virtualized systems, this model aligns with the NIST Device-agent/gateway deployment paradigm. In essence, all resources and applications in this model are hosted on virtual servers, and users gain remote access to them through successful authentication to the VDI server. All virtual desktops and servers adhere to policies defined by VMware's network and security virtualization software, NSX, functioning as a firewall.

According to Uttecht (2020), an advantageous feature of this model lies in the administrator's ability to finely segment the network, a concept termed micro-segmentation. Additionally, the model facilitates the creation, maintenance, and dynamic scaling of virtual desktops in an auto-scaling mode, enabling administrators to add or remove virtual desktops as needed. This capability proves beneficial in preventing permanent access for attackers who may have gained access to a virtual desktop (most commonly through phishing attacks). However, the model has inherent drawbacks, including potential cost implications for organizations transitioning to VDI-based architectures, particularly if their enterprise is not already utilizing such technology. Furthermore, it may not be a suitable

approach for enterprises housing numerous resources incompatible with virtualization, such as Internet of Things (IoT) devices.

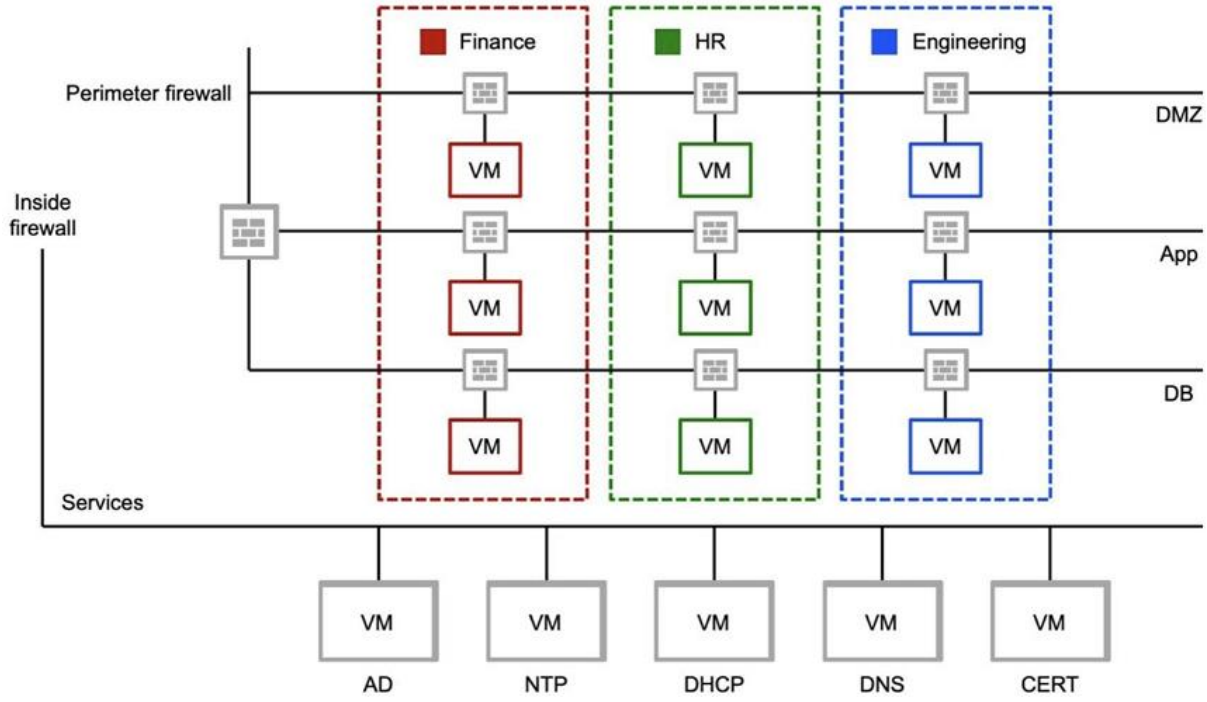

Figure 7. ZT Model Using NSX (K. Kumar, 2017, Figure 1).

Software-Defined Perimeter: proposed by Cloud Security Alliance (2013), Software-Defined Perimeter (SDP) uses two agents, one at the end device and the other at the application without being integrated with it, falling under the NIST Enclave-based deployment model, it basically acts as a gateway by creating another network on top of the existing network. To access the enterprise resources, the user needs to authenticate to the SDP server, then the SDP server creates a VPN tunnel between the user and the requested resource/application. However, unlike the traditional VPN, SDP VPN does not grant full access to all the enterprise resources to authenticated users, instead it gives access to specific resources based on various factors including attributes, roles, and/or user trust.

Uttecht (2020) suggests that this model is generally well-suited for a broad spectrum of federated solutions, as it does not necessitate integration with applications/resources, hosting on external cloud providers, or substantial modifications to users' working environments. Furthermore, it represents a cost-effective solution and proves applicable to enterprises with a substantial quantity of Internet of Things (IoT) resources by employing an SDP server to serve as a gateway for these resources.

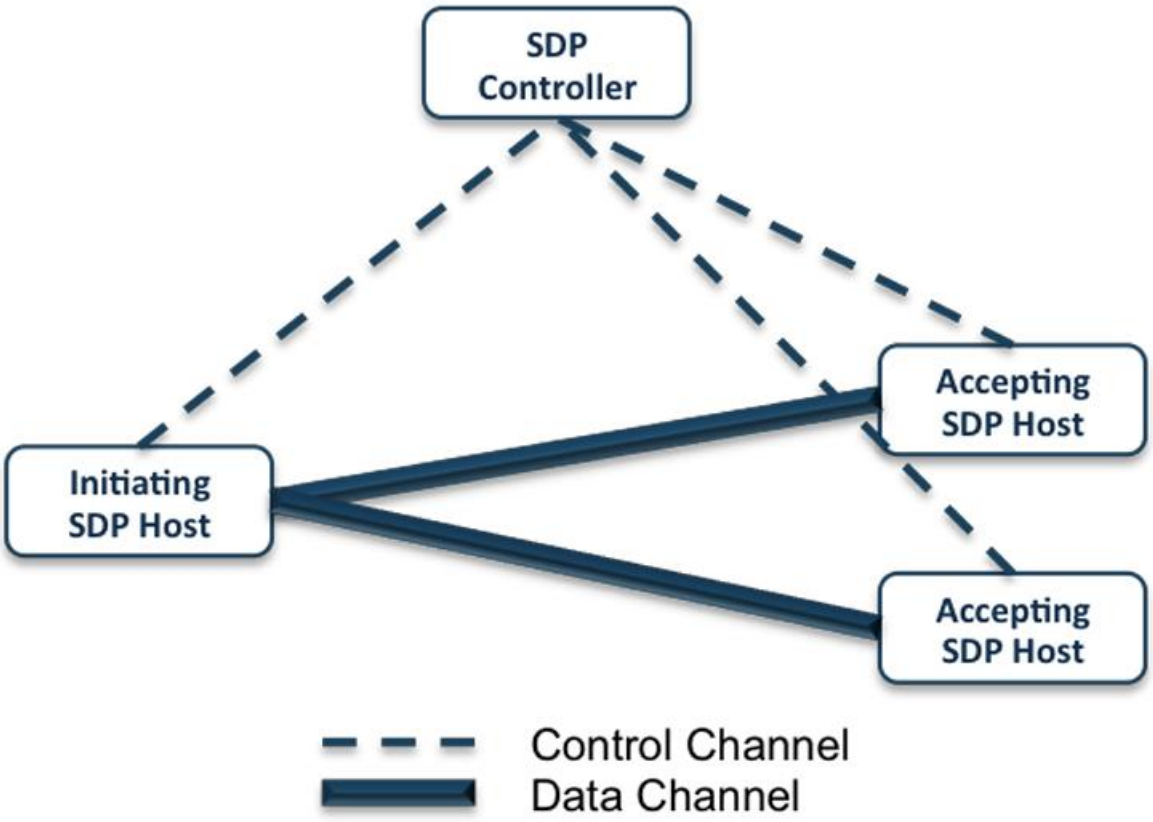

Figure 8. Workflow of the architecture of SDP (Cloud Security Alliance, 2013, Figure 1).

The author further recommends each organization to select a ZTA model based on their requirements and situation. They state for a governmental entity primarily engaged in public interaction, any of the aforementioned architectural frameworks would suffice, with the selection contingent upon the pre-existing infrastructure. Optimal advantages would be derived from adopting a virtualized architecture akin to VMWare/NSX, despite the potential complexities associated with migration. Conversely, a public safety-focused agency, characterized by numerous field operatives with intermittent internet connectivity and reliance on diverse sensors and Internet of Things (IoT) devices, would find a SDP-based solution particularly advantageous. In instances where a broader organizational structure encompasses sub-entities akin to the previously delineated exemplars, a federated architectural model is recommended to adeptly accommodate the array of distinct requirements.

2.8 Discussion

For this research, based on the various ZT models covered so far, a federated ZT solution designed particularly for AWS platform will be proposed. Due to time constraints and budget issues, only a very small model will be proposed considering the main factors in each model. According to the

author, there are various factors we should consider when deciding whether or not to upgrade the organization to a ZT architecture. We should ask ourselves:

What level of integration do we need? ZT can be used to provide maximum security, but possibly at the expense of more complicated resource management.

What level of policy enforcement do we need? Not all policies are compatible with ZT. Security policies and standards should be chosen based on our organization's needs and values.

Is there a whole single network across our organization or multiple sub networks with different policies?

Are our applications and devices fully supported by the chosen ZT architecture? Sometimes the already-established protocols make it difficult to migrate to a ZT model.

How can we identify the diversity of missions across our organization as a whole and as various units?

Should we start implementing ZTA from scratch (impossible in many cases), or in an incremental solution? Do we have the budget for it?

Although ZTA has the potential to enhance the security of our organization when appropriately integrated with identity and access management, continuous monitoring, adherence to security best practices, and compliance with policies and guidance, it is susceptible to threats primarily linked to ZT. Some of these threats encompass:

Threat 1: PE, PA, and PEP can act as bottlenecks and limit access or even completely disrupt services if configured improperly. In addition to limiting access, if compromised they can also allow malicious traffic into the internal network and directed to the resources.

Threat 2: DDoS/DoS attacks are one of the main concerns of ZTA. This is because PE, PA and PEP can be targeted by attackers as critical connectors.

Threat 3: Since in ZT trust is based on policies and not based on location, attackers need to gain access to accounts through various methods such as phishing, social engineering or even insider attackers. The accounts with high privilege become the targets, and if compromised, it can lead to severe consequences. High level administrators should respect security as their account have the highest privilege across the organization.

3 Methodology and Design

AWS users can use services and solutions offered both by AWS and other companies supported by AWS known as Partner Solutions. In this section, various AWS services and external services (partner solutions) which will be used in the implementation of the CEZT will be introduced to facilitate the understanding of this research and the corresponding implementation.

The reason behind choosing AWS over other alternatives is the global reach of AWS as it is recognized and widely used across all over the world.

3.1 Partner Solutions

OpenVPN: a free open-source VPN service which can be used to route traffic both on-premise and in the cloud (such as AWS) providing various security services including but not limited to encryption, advanced IP routing, DNS-based content filtering and enforcing strict identity-based policies suitable for ZT strategies thanks to integrating with Cyber Shield, a built-in DNS-based intrusion detection and prevention (IDPS) content filtering feature. Another reason for choosing OpenVPN is its cost-free nature, aligning with our cost-effective strategy.

Pfsense: a firewall service which offers several features, such as Snort, VPN and NAT. The rationale for selecting pfSense is its provision of multiple services essential for CEZT. The ability to utilize all its features, such as the CA server, NAT, firewall, and built-in IDS, simplifies management and can be more cost-effective compared to relying solely on AWS managed services. For instance, instead of using a NAT gateway, PfSense's NAT service can be employed, along with its other features, for a reasonable subscription fee.

3.2 AWS Services/Solutions

In this section various AWS services used during this research are listed.

3.2.1 Storage and Server Services

EC2: Amazon Elastic Compute Cloud (Amazon EC2) is a cloud-based web service offered by Amazon Web Services (AWS) designed to offer secure and scalable computing resources.

Amazon S3: S3 is a cloud-based object storage service offered by AWS. It facilitates the storage and safeguarding of diverse volumes of data for various purposes, including data lakes, websites, cloud-

native applications, backups, archives, machine learning, and analytics. Security measures such as encryption and adherence to security policies are employed to ensure the protection of stored data.

3.2.2 Tools used for Automation

AWS Lambda: AWS Lambda is a serverless service provided by AWS enabling the execution of scripts in response to events occurring on cloud resources and facilitates their management.

AWS SSM Runbooks: Runbooks offered by AWS Systems Manager (SSM) can be used to perform repeated tasks on well-defined events with the least data required.

3.2.3 Security Services

Security Hub: A cloud security posture management service offered by Amazon which automates the security checks against AWS resources. There are many benefits for security hub, including but not limited to:

> **Benefit 1:** Consolidated findings of all security checks from AWS security services and AWS partner network products (APN).
>
> **Benefit 2:** The findings from all these services are automatically normalized by AWS based on the AWS Security Finding Format (ASFF), which in turn reduces the Mean Time to Remediate (MTTR) and managing the findings.
>
> **Benefit 3:** Security Hub can be integrated with many external security services from APN, such as Security Information and Event Management (SIEM), Security Orchestration, Automation, and Response (SOAR) and Governance, Risk, and Compliance (GRC) tools.
>
> **Benefit 4:** Automated, continues security monitoring and threat detection based on AWS security best practices and other industry or custom compliance frameworks.
>
> **Benefit 5:** Automated remediation based on the findings.
>
> **Benefit 6:** Single account, multi-account and cross-region aggregation monitoring and remediation.

Amazon Inspector: a tool offered by AWS for automated security monitoring and vulnerability remediation based on AWS security best practices and other security standards. Amazon Inspector continuously scans the target resources through agents installed on the resources, alerts the

administrator in case of vulnerability findings, and offers remediation steps. The findings include: Name of assessment target and template, assessment details including start and end times and status, name of rule package, details about the finding including name, severity, and its description, remediation guidance. Rule Packages Amazon Inspector works with include:

Centre of Internet Security (CIS) benchmark, Common Vulnerabilities and Exposures (CVEs), security best practices defined by AWS security experts, and runtime behaviour analysis; the process of continuous monitoring and analysis of systems and applications in their runtime to detect anomalies, deviations and unusual behaviours and vulnerabilities during the runtime phase. Amazon Inspector monitors resources at both network level and application level.

Amazon GuardDuty: A managed regional-based security service offered by AWS for which can be used to automate monitoring one's AWS environment for malicious and anomalous activities. It also provides malware detection and protection, and forensics investigations. Threat detection by Amazon GuardDuty is done by continuously analysing data from AWS CloudTrail, Amazon Elastic Kubernetes Service (EKS) audit logs, DNS logs, Amazon Virtual Private Cloud (VPC) Flow Logs, and Amazon Simple Storage Service (S3) events. Amazon GuardDuty submits the security findings to both CloudWatch Events and GuardDuty console in detail, which makes it possible to actionize alerts and integrate them into existing event management or workflow systems, providing automatic remediation. Threats are intelligently detected by AWS, and you do not need to write the rule sets or detection logics as they are created and continuously updated and maintained by AWS. Amazon GuardDuty employs multiple detection methods, including Heuristics, machine learning (ML), anomaly detection, and fused threat intelligence, to detect potential signs of compromise. Automated remediation is possible by integrating Amazon GuardDuty with other services (both AWS services and external security information and event management (SIEM) and other security tools), such as Amazon EventBridge, AWS Lambda, and AWS Security Hub. Forensics investigations for finding the root cause of the security issues is also possible thanks to tight integration of Amazon GuardDuty and Amazon Detective. Some other benefits to use Amazon GuardDuty include: first, the possibility of using it cross-accounts, providing security across multiple AWS accounts through AWS Organization; second, the ability to send alerts to security staff through email or SMS, informing them about the threats right away; and third, clear fees as they are only based on the analysed events without any upfront payment.

in addition, Amazon GuardDuty functions autonomously from your resources, ensuring no compromise to the performance or accessibility of your workloads. The detections can be categorized into 5 different groups:

Group 1 - Reconnaissance: Indications hinting at reconnaissance by an assailant encompass atypical API behavior, internal VPC port scanning, irregular sequences of unsuccessful logic requests, or unhindered probing of ports originating from a recognized malicious IP address.

Group 2 - Instance Compromise: Signs signalling a potential instance compromise encompassing various indicators such as cryptocurrency mining, an upsurge in network traffic, utilization of unfamiliar network protocols, outgoing denial-of-service (DoS) activities, and communication between an EC2 instance and a malicious IP address.

Group 3 - Account Compromise: Frequent indications suggesting potential account compromise involve patterns such as API requests originating from an atypical geolocation through anonymizing proxies, endeavours to deactivate AWS CloudTrail logging, abnormal launches of instances or infrastructure, deployment of infrastructure in unconventional regions, and API activities originating from a malicious IP address.

Group 4 - Bucket Compromise: Indicators signalling a potential compromise of a bucket including irregular data access patterns suggesting potential credential misuse, uncommon S3 API behaviour originating from a distant host, unauthorized access to S3 from recognized malicious IPs, and API requests to retrieve data from an S3 bucket by a user lacking a history of accessing the bucket or initiated from an unusual location.

Group 5 - Amazon EKS Cluster Compromise: signs signalling an Amazon EKS cluster compromise including access by unauthorized actors or from Tor network, API calls originating from anonymous users indicating misconfiguration or patterns of known privileged-escalation methods.

AWS CloudTrail: AWS CloudTrail is a multi-region multi-account service which serves as a pivotal security service within the AWS ecosystem, meticulously documenting activities, and all API calls, auditing events classified into management events (control plane operations such as configuring security groups, launching instances, and setting up logging within one's AWS account), data events (data plane operations executed on or within resources, encompassing object-level actions on S3 buckets and objects, Amazon DynamoDB activity at the object level within tables, AWS Lambda function operations, and other supported AWS services that contribute to data-related activities), and

CloudTrail Insights events (events based on anomalies when substantial variances in API usage patterns diverge from the account's established norms, triggered after establishing a baseline of routine AWS activity), enabling operational and risk auditing, governance, and compliance across an individual's AWS infrastructure in addition to creating insights based on the logs to identify anomalies. It meticulously records the identity of the entity effecting a change, the specific resources subjected to modification, and the intricate details of the alteration itself. This service empowers users to view, audit, analyse, and respond to account activities comprehensively. CloudTrail's comprehensive logs provide a chronological sequence detailing the initiator, timestamp, nature, and description of each alteration, fostering enhanced accountability, and facilitating forensic analysis within the AWS environment. The benefits of CloudTrail include:

Benefit 1: Ability to archive all audited events during the last 90 days and search through them based on time, API operation, username, or resources.

Benefit 2: The capability to seamlessly integrate with numerous AWS services and a wide array of partner integrations across various operational and security solutions.

Benefit 3: CloudTrail Insights expedites the detection and handling of unconventional operational behaviour, promptly identifying anomalies like sudden increases in resource provisioning, surges in IAM activities, and interruptions in routine maintenance cycles. Automating the analysis of API calls and usage patterns allows for the swift detection of any atypical behaviour, including unusual error rates, enhancing the ability to pinpoint irregularities within your AWS account.

Benefit 4: Trails for CloudTrail adeptly gather and persist both management and data events spanning multi-Region and multi-account setups. Tailoring log retention periods enables adherence to regulatory requirements. Additionally, integrating trails with CloudWatch Logs or CloudTrail Lake facilitates the storage of activities, while employing advanced event selectors streamlines the filtering process for data events, ensuring a more refined and seamless data handling mechanism.

Benefit 5: CloudTrail Lake serves as a purpose-built managed multi-region multi-account data lake, consolidating, securely housing, and facilitating query access to CloudTrail events. Tailored for auditing, security investigations, and operational issue resolution, it ensures complete immutability of stored events. Its architecture optimizes event data storage for a default 7-year period, allowing efficient querying through SQL-based searches. Operating

within your AWS Organization, CloudTrail Lake conveniently centralizes event management from diverse accounts and Regions. Advanced event selectors further streamline event filtration, enhancing its utility for seamless event handling.

Amazon Detective: A managed security tool offered by AWS which facilitates the process of forensics investigations through machine learning, statistical analysis, and graph theory after aggregating and analysing security findings ingested from various sources including AWS CloudTrail, Amazon VPC Flow Logs, Amazon Elastic Kubernetes Service (Amazon EKS) audit logs, and GuardDuty findings by maintaining a maximum of a one-year period of historical data through a set of visualizations. Some of the benefits Amazon Detective provides include: detecting anomalies suggesting security issues over time, finding resources affected and linked to those anomalies, followed by analysing the anomalies to find the root cause of security issues.

One of the security common practices to use CloudTrail is to store CloudTrail logs in a dedicated S3 bucket in an administration account and possibly use S3 Object Lock to prevent attackers from changing or deleting the logs, should any credentials be compromised. However, due to budget constraints, for this research I will use all the services including CloudTrail inside one AWS account. Another best practice is to use AWS Organizations to divide each team of the organization into different groups with different accounts. Then the security tools are managed by, and security findings are stored to the security team account. However, again due to budget constraints for this research no organization will be created, and all the lab entities will be provisioned in the same AWS account.

Next, the factors considered for developing a tailored ZT model suitable for deployment on the AWS platform are delineated, drawing inspiration from the literature review and various ZTA approaches.

When confronted with the decision of selecting from different NIST ZT models, it is acknowledged that each model possesses merits, and the optimal choice hinges on organizational considerations.

4 Zero trust Architecture Design for Cloud Computing

The preceding Chapter outlined pertinent questions that should be deliberated before opting for a specific ZT approach, which are, how to provide robust ZT security policies for infrastructure deployed on AWS platform in an automatic and cost-effective way.

Among the available options, namely Device Agent/Gateway-Based Deployment, Enclave-Based Deployment, and Resource Portal-Based Deployment, the latter has been chosen for this research endeavor. The rationale behind this decision is rooted in the ability to forego the installation of an agent on end-point devices, aligning with the imperative of expediency imposed by time constraints and the limitations inherent in being a solo researcher. In addition, SDN was chosen to be the base of this model, as it simplifies network and security management through decoupling the control plane from the data plane (Eliyan & Di Pietro, 2021).

In the context of Resource Portal-Based Deployment, the Policy Enforcement Point (PEP) portal is strategically positioned as a gateway. The policy engine (PE), policy administrator (PA), and the policy enforcement point (PEP) can be seamlessly integrated into a singular component. To operationalize this configuration, Pfsense, a well-regarded open-source firewall, emerges as a viable option due to its widespread recognition and functionality.

As policy engine (PE): Pfsense can enforce various policies as a firewall to either grant access, refuse or terminate an existing connection.

As policy administrator (PA): Pfsense can allow/disallow access according to the rules set by the administrator.

As policy enforcement point (PEP): Pfsense can use Network Address Translation (NAT) to direct traffic to the target resource according to the policies.

For a ZT approach, there are some similarities between all ZT models that we can take into account:

Common Rule 1: All unused ports must be closed.

Common Rule 2: connections need to be encrypted, preferably through VPN.

Common Rule 3: Secure Contingency Planning, to minimize access of both regular users and administrators through proper privilege assignment.

Common Rule 4: A whitelist approach to allow connections only from trusted users.

Common Rule 5: Automation of manual processes for managing policies as much as possible to decrease burden and human error.

Common Rule 6: Continuous monitoring for security threats, both internal and external using various services such as IDS.

In establishing secure communication between users and AWS, OpenVPN emerges as a highly advantageous option due to its status as a freely available VPN service with widespread recognition and usage. An alternative to OpenVPN is AWS VPN, a managed service provided by Amazon Web Services. While opting for AWS VPN simplifies management procedures given its status as a fully managed service by AWS, it should be noted that this convenience comes at a cost, as AWS VPN is not provided as a free solution. In contrast, the implementation of OpenVPN demands a more intricate configuration process and poses challenges in deployment. However, its cost-effectiveness makes it a viable choice within our financial considerations. It is essential to acknowledge that while OpenVPN itself is free, the deployment necessitates an Amazon Elastic Compute Cloud (EC2) instance, the usage of which incurs costs. Despite this cost implication, opting for an EC2 instance remains a more economical alternative compared to the expenses associated with AWS VPN. Thus, the decision to prioritize cost efficiency involves a trade-off, where the complexity of OpenVPN deployment is accepted for the sake of reduced expenditure. An additional advantage of choosing OpenVPN over AWS VPN lies in the potential incorporation of "Cyber Shield," serving as a supplementary security layer for our enterprise. This feature functions as an intrusion detection and prevention system, enhancing the overall security posture of the VPN deployment. OpenVPN, being an open-source solution, offers the flexibility to integrate third-party security solutions like Cyber Shield, allowing for a comprehensive approach to threat detection and prevention. This capability positions OpenVPN as a versatile choice for enterprises seeking to fortify their network infrastructure with advanced security measures beyond the standard VPN functionality provided by AWS VPN.

Within the realm of ZT strategies explored in the literature review, the decision has been strategy's comprehensive approach, delineated through various pillars, rendering it more intricate and detailed compared to the NIST ZT models. While NIST ZT models tend to adopt a more generic stance, the Forrester extended ZT strategy stands out due to its nuanced and detailed framework, providing a more granular and specialized perspective on implementing Zero Trust principles. This strategic focus aligns with the aim of pursuing a ZT approach that offers a more elaborate and tailored guidance for enhancing cybersecurity within the organizational context.

In the context of implementing identity-based policies within a ZT model on our AWS platform, the chosen approach involves leveraging AWS Lambda. The rationale behind this decision lies in the serverless nature of AWS Lambda, which facilitates the execution of custom scripts without the necessity for explicit server configuration. AWS Lambda provides a seamless and resource-efficient environment for running code in response to events, making it an optimal solution for enforcing identity-based policies within a ZT framework. By utilizing Lambda, custom scripts can be developed and executed in a serverless manner, contributing to operational simplicity and flexibility. The serverless architecture of AWS Lambda aligns well with the principles of Zero Trust, as it minimizes the attack surface and reduces the need for persistent infrastructure. This approach not only enhances security by enforcing identity-based policies but also ensures efficiency and ease of maintenance in the dynamic and evolving landscape of a ZT model. Figure 9 presents the basic architecture of CEZT.

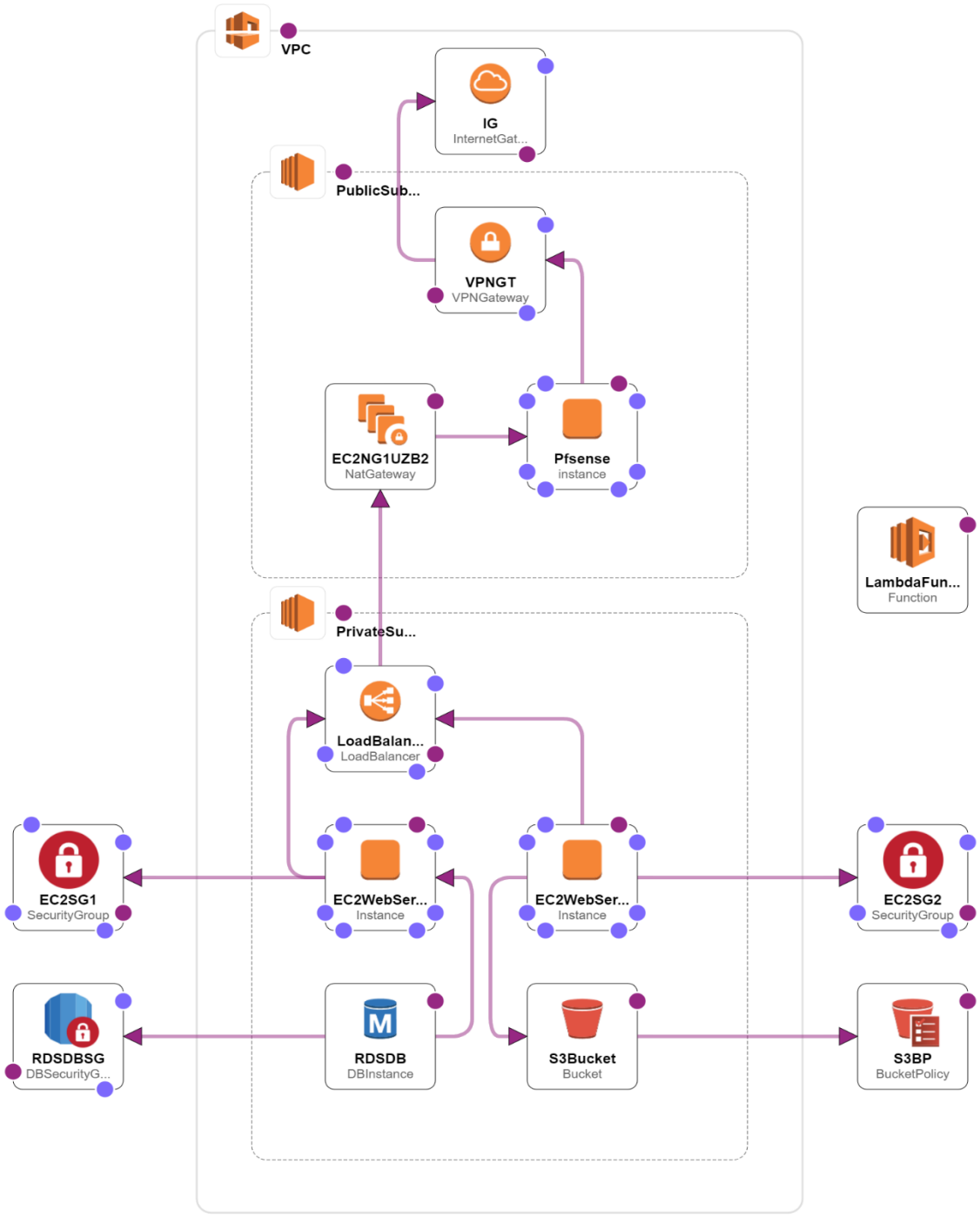

Figure 9. CEZT Basic Architecture. Diagram created by AWS CloudFormation Designer

In the delineated model, as depicted in the Figure 9, the Virtual Private Network (VPC) is partitioned into a public subnet and a private subnet. The former is linked to the internet via an Internet Gateway (IG), while the latter is deliberately isolated from direct internet access, yet possesses an indirect route through adherence to policies specified by Pfsense. This conceptual framework advocates for the concentration of all resources within the private subnet, leveraging security groups for EC2 and RDS, and bucket policies for S3 buckets to establish isolation. A security group for an EC2 instance and an

RDS database, as well as a bucket policy for an S3 bucket, act as dedicated firewalls for each, enforcing policies based on pre-defined rules. As an illustration, EC2Webserver1 interfaces with RDSDB (database) while maintaining isolation from other services. Corresponding security groups enforce the isolation policies on both entities.

Conversely, EC2Webserver2 connects to an S3 bucket based on policies dictated by the respective security group for EC2 and the bucket policy for S3, similarly isolated from other services. Both EC2 servers within the private subnet are linked to a load balancer, which, in turn, interfaces with a Network Address Translation (NAT) gateway. The NAT gateway establishes a connection to Pfsense, which is interlinked with a VPN gateway (OpenVPN), ultimately affording the VPN gateway internet access through the Internet Gateway. Consequently, resources in the private subnet retain internet accessibility in accordance with stringent policies enforced by Pfsense and the VPN server located in the public subnet, aligning with a zero-trust strategy. The Lambda function diligently monitors the entire VPC, systematically implementing the CEZT policies on both resources and users, thereby offering an automated approach.

4.1 Segmentation of the Network

Initially, segmentation of the Virtual Private Cloud (VPC) into distinct subnets is imperative, delineating a public and private subnets. The public subnet facilitates direct connectivity to the internet and serves as the locus for implementing zero trust policies, whereas the private subnet affords only indirect internet access through the intermediary of the public subnet. The establishment of subnets can be achieved through the utilization of a code akin to that depicted in Figure 10, employing the AWS Command Line Interface (CLI). It is pertinent to acknowledge that sensitive data, such as IP addresses, have been redacted as "****" to uphold security measures.

```
subnet_id=$(aws ec2 create-subnet \
    --vpc-id **** \
    --cidr-block **** \
    --availability-zone **** \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=****}]' \
    --query 'Subnet.SubnetId' \
    --output text)
```

Figure 10. Sample Code for Creating Subnets in AWS CLI

This code facilitates the creation of a subnet endowed with direct internet access via a routing table. To transform the subnet into a private entity, the routing table can be eliminated utilizing the AWS CLI, as elucidated in Figure 11.

While the public subnet enjoys unfettered connectivity to the internet along with a public IP address, the private subnet necessitates indirect internet access via the public subnet. For this purpose, the private subnet leverages either a NAT gateway or a NAT instance, ensuring internet access through the intermediary of the public subnet. Both NAT gateway and NAT instance necessitate a static IP address to maintain persistent connectivity amidst infrastructure alterations. The initial step entails the creation of an elastic IP, followed by the establishment of a NAT gateway linked to the elastic IP. Figure 11 delineates the exemplar code for generating an elastic IP, alongside the corresponding response garnered from the AWS CLI.

```
# Allocate a new Static IP address
aws ec2 allocate-address --domain vpc
# Response from AWS CLI
{
  "PublicIp": "****",
  "AllocationId": "****",
  "PublicIpv4Pool": "amazon",
  "NetworkBorderGroup": "****",
  "Domain": "vpc"
}
```

Figure 11. Generating an Elastic/Static IP

The subsequent task involves associating the NAT gateway with the Allocation ID. Following this, the creation of a NAT gateway and its association with the Allocation ID is imperative. Please refer to Figure 12 for detailed instructions on executing this process via the AWS CLI. Ensure to utilize the ID of the public subnet for the Subnet ID parameter during the execution.

```
nat_gateway_id=$(aws ec2 create-nat-gateway \
  --subnet-id **** \
  --allocation-id **** \
  --output text \
  --query 'NatGateway.NatGatewayId')
```

Figure 12. Creating a NAT Gateway for the Public Subnet through AWS CLI.

To eliminate direct internet access from the private subnet, it is imperative to disassociate the associated routing table. However, given that the routing table is utilized across all subnets and the objective is to solely remove it from the private subnet, a sequential process is necessitated. Initially, a new routing table is created, followed by its association with the private subnet. Subsequently, the routing table is removed, ensuring exclusive removal of routes solely from the private subnet. Figure 13 elucidates the procedural steps involved in creating a routing table, associating it with the private subnet, and subsequently removing it through the AWS CLI.

```
# Create a new routing table
private_route_table_id=$(aws ec2 create-route-table --vpc-id **** --query
'RouteTable.RouteTableId' --output text)
# Associate the routing table to the private subnet
aws ec2 associate-route-table --subnet-id **** --route-table-id ****
# Add a new route to the routing table which directs all the outbound traffic to the NAT
Gateway
aws ec2 create-route \
  --route-table-id **** \
  --destination-cidr-block 0.0.0.0/0 \
  --gateway-id ****
# Remove any other routes except the default local route
aws ec2 delete-route --route-table-id **** --destination-cidr-block ****
```

Figure 13. Associating a NAT Gateway to the Private Subnet through AWS CLI

An alternative solution involves employing a NAT instance instead of a NAT Gateway. While both serve the same purpose of facilitating outbound internet connectivity for private subnets, a key distinction lies in their management approach. NAT Gateways are AWS-managed services, relieving users of maintenance and security concerns, whereas NAT instances necessitate continuous user oversight, including maintenance, updates, and security patching. Although NAT Gateways offer simplicity, NAT instances present a cost-effective option, as users only incur expenses for the EC2 instance hosting the NAT instance, which is typically less costly than a managed NAT Gateway. The selection between the two options hinges on organizational objectives and budgetary considerations. For instance, a nascent startup with limited financial resources may opt for a NAT instance to align with budget constraints during the initial stages of operation.

In this research endeavor, cost-effectiveness emerges as a primary consideration, thus motivating the utilization of a NAT instance over a NAT Gateway. Establishing a NAT instance entails provisioning an EC2 instance and deploying a NAT-supporting operating system or product. Among the widely recognized products compatible with AWS, PfSense emerges as a prominent choice for its versatility and NAT capabilities.

In the proposed implementation, a PfSense instance will be deployed within the public subnet to enable internet access through the internet gateway. Subsequent to installation, adjustments to NAT rules are imperative to facilitate internet access for resources residing in the private subnet via PfSense. This necessitates the removal of default NAT rules assigned by AWS and the incorporation

of additional rules tailored to the enterprise's architectural requirements, particularly those pertaining to IPSec for the public subnet.

Furthermore, configuring PfSense settings to capture all traffic, including traffic with a destination address other than that of Pfsense, is essential to prevent packet loss. By default, AWS configures PfSense to drop packets with destination addresses other than its own, necessitating adjustments to accommodate comprehensive traffic handling.

4.1.1 Setting up the Required Policy-Enforcing Resources in the Public Subnet

The public subnet is used to enforce most of the ZT policies on the resources placed in the private subnet. All the resources needed in the public subnet include:

Firewall: To enforce firewall regulations on inter-subnet traffic, PfSense will be leveraged. In line with Zero Trust principles, these regulations must adhere to company requirements and be structured in a whitelist format to strictly control permissible communication paths. However, for flexibility, the enterprise may opt for a grey-list approach based on specific needs. It's important to note that while a whitelist approach is recommended for safeguarding critical infrastructure due to its stringent control measures, a grey list offers a degree of flexibility by permitting communication unless explicitly restricted. Thus, the choice between whitelist and grey-list formats should be made considering the organization's security priorities and operational requirements.

IDS/IPS: To detect anomalies and known threat patterns, an IDS based on Snort, supported by PfSense, will be implemented. This system will be configured with a diverse array of Snort rules aimed at identifying and mitigating attacks. These rules encompass a broad spectrum of threats, including but not limited to SQL injection (SQLI) attacks, Cross-Site Scripting (XSS) attacks, various Denial of Service (DoS) attacks, and Spoofing attacks. The comprehensive set of rules will enable proactive identification and alerting of suspicious activity, empowering administrators to promptly respond to potential security breaches. In addition, the MITC attacks mentioned in the earlier sections can be mitigated to a great extent, as now all inbound and outbound traffic can be monitored by Snort to spot malicious traffic.

Internet Gateway: the internet gateway to bring internet access to the public subnet.

VPN Server: A VPN server serves dual purposes: firstly, to encrypt traffic between the cloud and external networks (such as the internet), and secondly, to distribute SSL/TLS certificates to users outside the cloud. This approach aligns with the Zero Trust principle of "trust no one," ensuring that

only users possessing certificates issued by the VPN server are authorized to transmit and receive packets with resources located in the private subnet. For the VPN server implementation, the VPN service provided by PfSense will be utilized in conjunction with OpenVPN, a free and open-source solution. This choice is consistent with the cost-effectiveness considerations inherent in the CEZT strategy. By leveraging OpenVPN's capabilities within the PfSense framework, the organization can establish a secure and scalable VPN infrastructure that effectively meets encryption and authentication requirements, thereby enhancing overall network security posture.

NAT Instance: for NAT, the NAT service offered by Pfsense will be used.

Elastic IP: Assigning Pfsense, which serves as the VPN server, IDS/IPS system, firewall, and NAT, with an elastic (static) IP is essential. The utilization of a static IP aims to ensure network stability, consistent performance, and minimized interruptions. Additionally, employing a static IP simplifies management tasks, streamlining administrative processes associated with network configuration and maintenance. By assigning a static IP to Pfsense, the organization can reliably access and manage critical network components, enhancing overall operational efficiency and facilitating seamless network operations.

4.2 ZT Policies Enforcement on EC2

AWS CloudWatch and AWS Lambda are integrated to create a Lambda function capable of responding to events triggered by resource creation. This Lambda function is designed with appropriate security groups and IAM roles to inspect newly created resources, specifically focusing on EC2 instances situated within the public subnet. Upon detection, the function runs a custom AWS SSM Runbook to enforce policies to relocate the instance to the private subnet while adjusting its security group and IAM role to restrict traffic exclusively to and from PfSense.

To provide flexibility and circumvent ZT policies when necessary, a tag-based approach is adopted. The Lambda function is configured to examine whether an EC2 instance possesses a tag labeled "Isolation" with a value of "False." Instances bearing this tag are exempted from ZT policies.

Furthermore, mitigation measures against Denial of Service (DoS)/Distributed Denial of Service (DDoS) attacks are implemented. If an instance is deemed vulnerable, its connections to PfSense are cut, and a more resilient infrastructure is deployed. This involves creating a load balancer and an Auto Scaling Group (ASG) within the private subnet. The EC2 instance is then associated with the ASG, which dynamically adjusts instance capacity based on traffic volume. The load balancer distributes

incoming traffic among instances, enhancing resilience against DoS/DDoS attacks by constant instance health monitoring and swiftly replacing compromised instances.

Additionally, to uphold the Zero Trust principle, all EC2 instances, regardless of their subnet placement, are treated as untrusted. PfSense is designated as a Certificate Authority (CA) to issue and distribute certificates. Only instances possessing a valid certificate signed by the CA are deemed trustworthy, thereby enforcing stringent security measures.

This multifaceted approach ensures robust security enforcement while offering flexibility and resilience to accommodate varying operational requirements and mitigate potential threats effectively.

Whenever an instance is moved from one subnet to another by Lambda and SSM, SSM triggers SNS to send an email to the administrator to inform them about the action. Server-side encryption should be enabled for the encryption of data at rest. Data at transit is automatically encrypted by SNS. In addition, to be in line with a ZT approach, all emails sent by SNS should be logged in Cloudwatch. for this, proper IAM rules need to be assigned to the SNS topic. For the SNS topic, a whitelist format of users who can receive the email should be added. It is suggested to use proper filtering to filter the message in terms of sensitive data to show to different priviledge users only the information they need to know. Figure 14 shows the diagram of the custom SSM runbook.

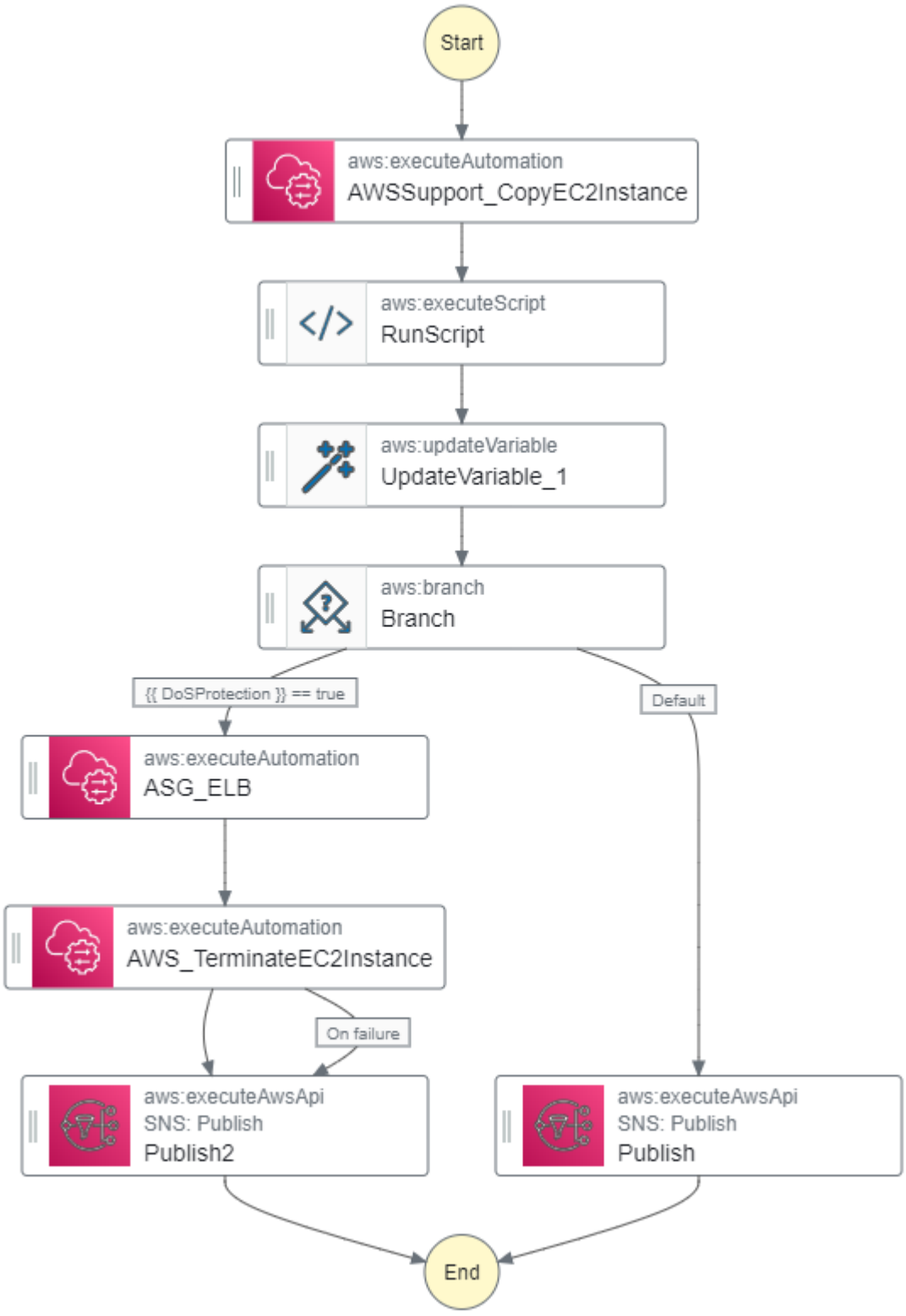

Figure 14. SSM RunBook Diagram. Diagram generated by AWS SSM.

This customized operational procedure utilizes a combination of AWS runbooks (both custom and AWS-offered), customized sub-runbooks, and AWS APIs associated with Simple Notification Service (SNS) and Elastic Compute Cloud (EC2). The runbook encompasses diverse inputs,

including the EC2 instance ID, randomly generated names for load balancer and Auto Scaling Group (if applicable), security group IDs, and the corresponding SNS Amazon Resource Name (ARN). The sequential steps involved are outlined as follows:

AWSSupport_CopyEC2Instance: This initial step leverages an AWS-provided runbook to address the limitation on AWS, where changing the subnet ID of an instance is not feasible. By employing this runbook, an Amazon Machine Image (AMI) of the instance is created, followed by launching a new instance based on the generated AMI within the private subnet. Subsequent actions involve terminating the original instance, essentially facilitating the relocation of an instance from one subnet to another.

Runscript and UpdateVariable_1: This phase encompasses the execution of a Python script designed to extract pertinent information from the payload of the preceding step, notably including the ID of the newly created instance.

Branch: This segment conducts a verification process to ascertain whether the DoS Protection tag is enabled. If affirmative, the workflow proceeds to step 5.

Email Notification via SNS: In the absence of DoS Protection tag activation, this step orchestrates the dispatch of an email notification via SNS, encompassing comprehensive details regarding the executed action. The whole process ends here.

ASG and Load Balancer Configuration: This stage involves the creation of an Auto Scaling Group (ASG) and a load balancer. Subsequently, the new instance is integrated into the ASG, followed by attaching the ASG to the load balancer. Furthermore, the load balancer is interconnected with Pfsense.

Termination of Initial EC2 Instance: Here, the original EC2 instance situated within the public subnet is terminated, ensuring the seamless transition of operations.

Email Notification via SNS: Analogous to step 4, this final step orchestrates the dispatch of an email notification via SNS, furnishing comprehensive details regarding the executed action.

This structured procedure encapsulates a series of meticulously orchestrated steps to facilitate the seamless management and migration of EC2 instances within the AWS infrastructure, leveraging a combination of runbooks, scripts, and AWS services.

4.3 ZT Policies Enforcement on S3

Unlike EC2 instances, which are assigned to VPCs and subnets, S3 buckets lack this capability. Instead, they are globally accessible within the AWS infrastructure. It's worth noting that EC2 instances can access S3 buckets through VPC endpoints, which enable the traffic to flow through the AWS network rather than being routed over the internet. For this purpose, an endpoint has been attached to the VPC. To enforce a zero-trust model, all S3 buckets must be encrypted and inaccessible to the public by default, while also implementing replication for DoS/DDoS attack mitigation. For forensics purposes, all the actions done on the bucket, such as deleting objects or any API calls related to the bucket, should be logged. A custom SSM runbook is created to automate these tasks. CloudWatch alarms are set to trigger an AWS Lambda function upon S3 bucket creation events. The Lambda function checks for an exception tag; if absent or disabled, it triggers the SSM runbook, which ensures logging, encryption, access control, and replication policies are applied appropriately. This comprehensive approach ensures consistent security posture across all S3 buckets, with automated responses to potential security threats.

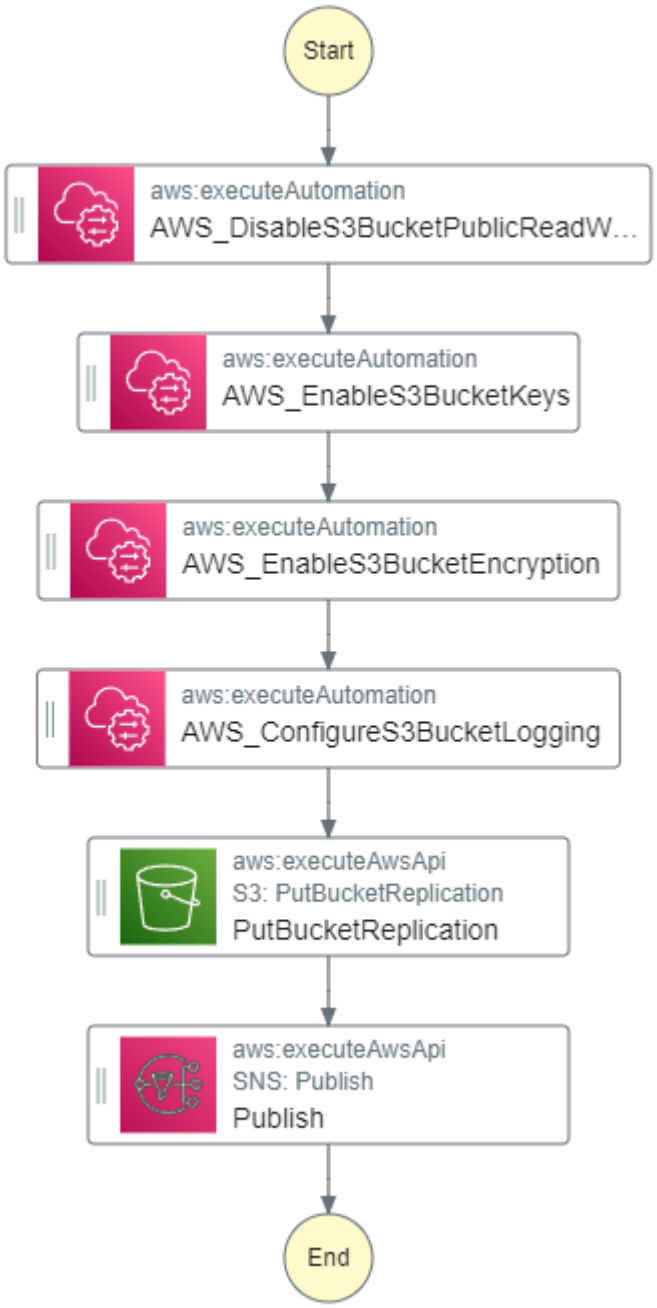

Figure 15. Diagram of Custom SSM Runbook for S3 Buckets. Figure Generated by AWS SSM

The runbook shown in Figure 15 consists of five key steps:

A**WS_DisableS3BucketPublicReadWrite:** This disables public accessibility for the S3 bucket, enhancing security by preventing unauthorized access.

AWS_EnableS3BucketKeys: Optionally assigns bucket keys to reduce the cost of server-side encryption by KMS, optimizing cost efficiency while maintaining security.

AWS_EnableS3BucketEncryption: Ensures encryption of the bucket and all objects at rest, bolstering data security, especially in scenarios where network intrusions may occur.

AWS_ConfigureS3BucketLogging: Enables logging of the bucket and stores the data in another bucket used for logging. It is recommended to place the target bucket in another account as the security account. It will later be used by Amazon Detective for forensics purposes.

PutBucketReplication: Utilizes the AWS API to replicate the bucket, providing protection against DoS/DDoS attacks and enabling efficient incident response in case of bucket unavailability or deletion. Together, these steps establish a robust security framework for S3 buckets, safeguarding data integrity and availability. Unlike the previous steps, this step requires a long JSON payload as input. A sample can be provided upon request.

SNS Publish: Using the API, administrators are informed about the actions performed by SSM through an email sent by SNS.

4.4 ZT Policies Enforcement on RDS

In configuring a ZT model for Amazon Relational Database Service (RDS), it's imperative to adhere to several key principles. Firstly, according to the principle of least privilege policies, RDS instances should be shielded from public accessibility when not necessary. Secondly, measures must be implemented to fortify the infrastructure against potential denial-of-service (DoS/DDoS) attacks, ensuring resilience and availability. Thirdly, frequent backups should be considered for incidence response purposes. Finally, safeguards should be instituted to mitigate the risk of accidental or malicious deletion of critical data. To streamline the automation of these security protocols, AWS Lambda and Systems Manager (SSM) are harnessed. CloudWatch serves as the initial trigger, prompting a Lambda function to verify the presence and activation status of an exception tag upon the creation of an RDS instance. If this tag is not found or enabled, Lambda seamlessly communicates the pertinent details of the RDS instance to a bespoke SSM runbook, which orchestrates the subsequent security measures as shown in Figure 16.

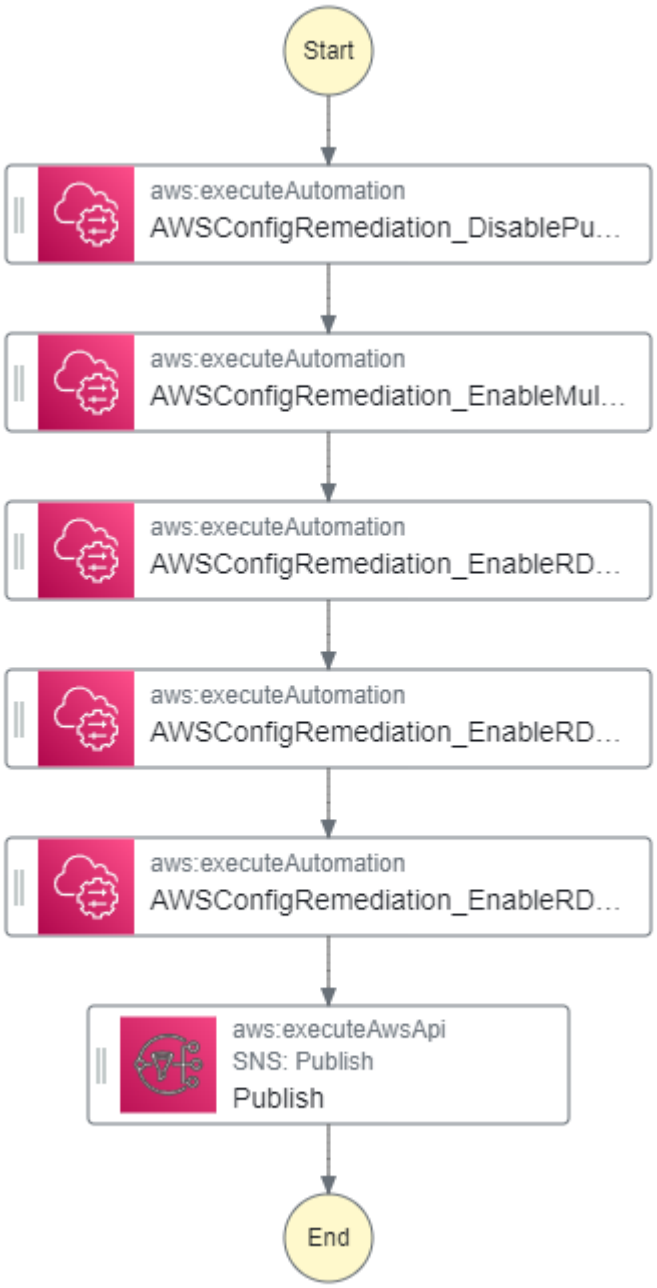

Figure 16. Diagram of SSM Runbook for RDS ZT Enforcement. Figure generated by AWS SSM

For successful operation of the custom runbook, Pfsense needs to be enabled as the NAT instance, and proper endpoints for the VPC should be configured, otherwise a timeout error is raised. The steps are as follows:

AWSConfigRemediation_DisablePublicAccessToRDSInstance: Public access to the RDS instance is disabled as a security precaution to restrict unauthorized access.

AWSConfigRemediation_EnableMultiAZOnRDSInstance: To mitigate the risk of denial-of-service (DoS/DDoS) attacks, the runbook configures the RDS instance to operate in multiple AZs. This ensures continued service availability even if one AZ is compromised.

AWSConfigRemediation_EnableRDSClusterDeletionProtection: Deletion protection is enabled at the cluster level to prevent accidental or malicious deletion of the entire cluster.

AWSConfigRemediation_EnableRDSInstanceDeletionProtection: Similar to the previous step, deletion protection is activated at the individual RDS instance level to safeguard against inadvertent deletion.

AWSConfigRemediation_EnableRDSInstanceBackup: Regular backups are enabled for the RDS instance to facilitate incident response and data recovery in the event of a security breach or data loss.

SNS Publish: Using the API, administrators are informed about the actions performed by SSM through an email sent by SNS.

These steps collectively establish a comprehensive security posture for RDS instances and clusters, aligning with the principles of zero-trust architecture and ensuring robust protection against potential threats and vulnerabilities.

4.5 ZT Policies Enforcement on Users

In adopting a zero-trust framework, several key considerations emerge. Initially, the enforcement of MFA for all users stands as a fundamental practice. Subsequently, users necessitate classification into distinct user pools to facilitate the allocation of privileges. Each pool should be associated with a dedicated certificate, subject to periodic renewal. Finally, a routine evaluation of IAM credentials is essential, wherein privileges are revoked should users demonstrate prolonged inactivity, thereby adhering to the principles of least privilege. The following figure shows a custom SSM runbook consisting of several scripts and sub-runbooks designed for automation of this purpose. The runbook is triggered by a Lambda function periodically. To make exceptions for the policies, assigning tags for each resource is one of the best practices of AWS.

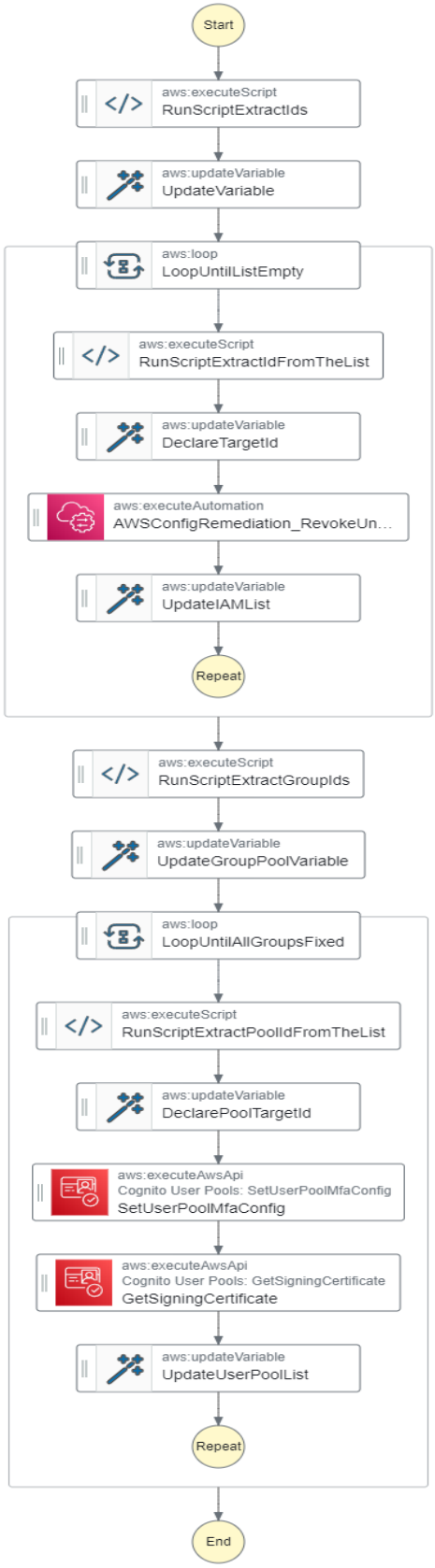

Figure 17. Runbook Diagram for ZT enforcement on Users. Figure generated by AWS SSM.

The comprehensive shown in Figure 17 outlines a series of steps within the runbook. To summarize, the initial phase involves the extraction of IAM resource identifiers, encompassing users, groups, roles, and policies. Subsequently, it proceeds to disable any inactive IAM credentials linked to these identifiers. Following this, the runbook retrieves user pool identifiers, ensures the activation of MFA for all users if previously disabled, and conducts certificate renewal procedures as necessary. All the scripts are written in Python. For successful operation of this runbook, AWS Cognito and AWS Config must be enabled.

5 Analysis and Further Discussion

This chapter presents the evaluation of ZT architecture design presented in Chapter 3. To evaluate the efficacy of our methodology in enhancing security measures, we will employ the following services: AWS GuardDuty, Amazon Detective, AWS Macie, AWS Inspector, and AWS Security Hub.

All the aforementioned services must be activated. Subsequently, Security Hub will be utilized to consolidate the results obtained from the aforementioned four services, as well as the findings derived from the following standards:

CIS AWS Foundations Benchmark v1.2.0 & v1.4.0: A set of AWS security best practices, consisting of various controls including but not limited to operating systems, network devices and cloud services (Amazon Web Services, 2024).

NIST Special Publication 800-53 Revision 5: A collection of various controls for providing security and privacy for information systems and enterprises developed by National Institute of Standards and Technology (NIST, 2020).

AWS Foundational Security Best Practices v1.0.0: A set of various controls for identifying non-compliant AWS resources and accounts with AWS security best practices defined by AWS experts (Amazon Web Services, 2024).

PCI DSS v3.2.1: A security standard suitable for organizations and enterprises who process cardholder data (PCI Security Standards Council, 2018).

Each standard caters to specific types of enterprises. For instance, PCI DSS v3.2.1 is tailored for organizations involved in transactions, such as banks, whereas the NIST framework is applicable for general purposes.

The test comprises two phases. In phase one, the framework is deactivated, and a vulnerable network is established. To ensure objectivity in the creation process, ChatGPT was utilized to propose a vulnerable network with diverse security risks. The vulnerable enterprise comprised two EC2 instances with enabled public access, situated in public subnets facing the internet, two S3 buckets with public access enabled, predominantly configured with default settings, and lacking encryption for both. Additionally, numerous unused security groups were present in the account, many of which granted public access to ports 22 and 3389. A MySQL database was also instantiated with default configurations. Alongside the vulnerable resources, various risks contradicting the principle of least

privilege of Zero Trust approach were present, and some resources inside the network were treated as trusted entities. Subsequently, CloudFormation served as the IaaS tool to instantiate the enterprise. Following this, AWS Config transmitted all configurations of the resources to Security Hub, which subsequently generated a security score for the enterprise, as depicted in Figure 18.

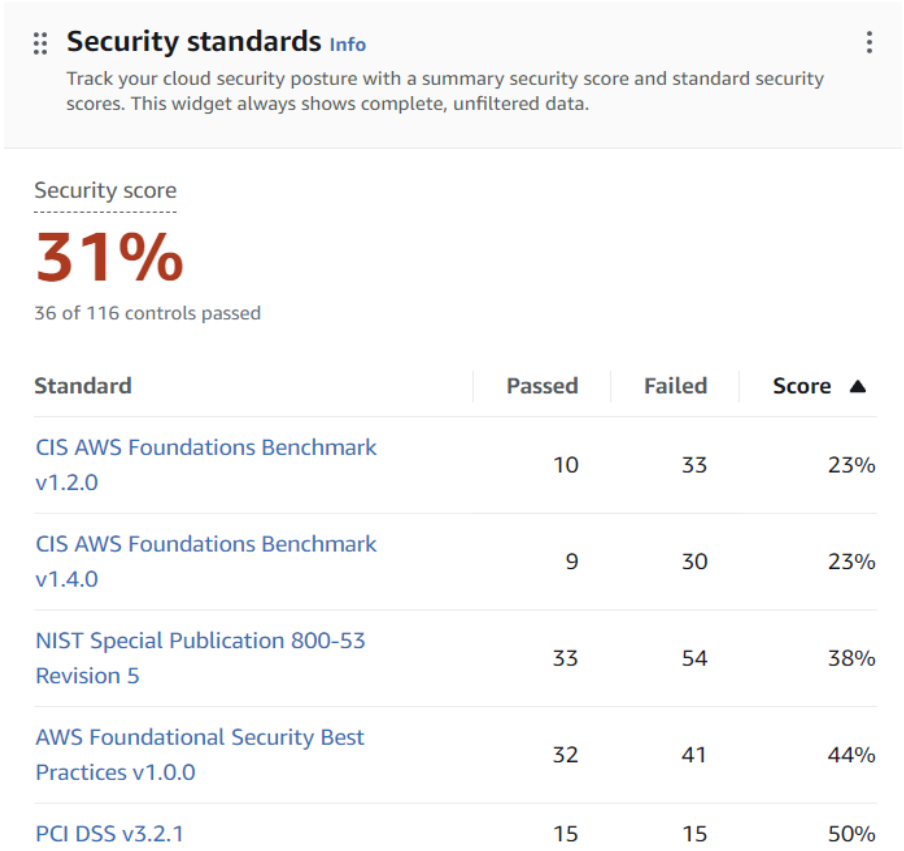

Figure 18. Overall Security Score Measured by Security Hub with the CEZT Disabled.

According to Security Hub metrics, the overall security level without enabling the CEZT stood at 31%. The lowest security scores were observed with the CIS AWS Foundations Benchmarks, each scoring only 23%, while the highest score was attained by PCI DSS v3.2.1, registering only 50%. Evidently, the enterprise faces substantial susceptibility to cyber-attacks. Figure 19 delineates all identified risks categorized by severity.

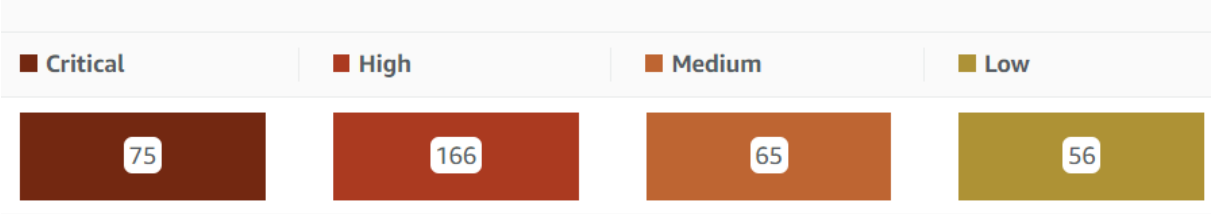

Figure 19. All the Risks Grouped by Severity, Measured by Security Hub Before Enabling the CEZT.

Furthermore, risks associated with compromised IAM rules were identified as the most prevalent threat type, as depicted in Figure 20.

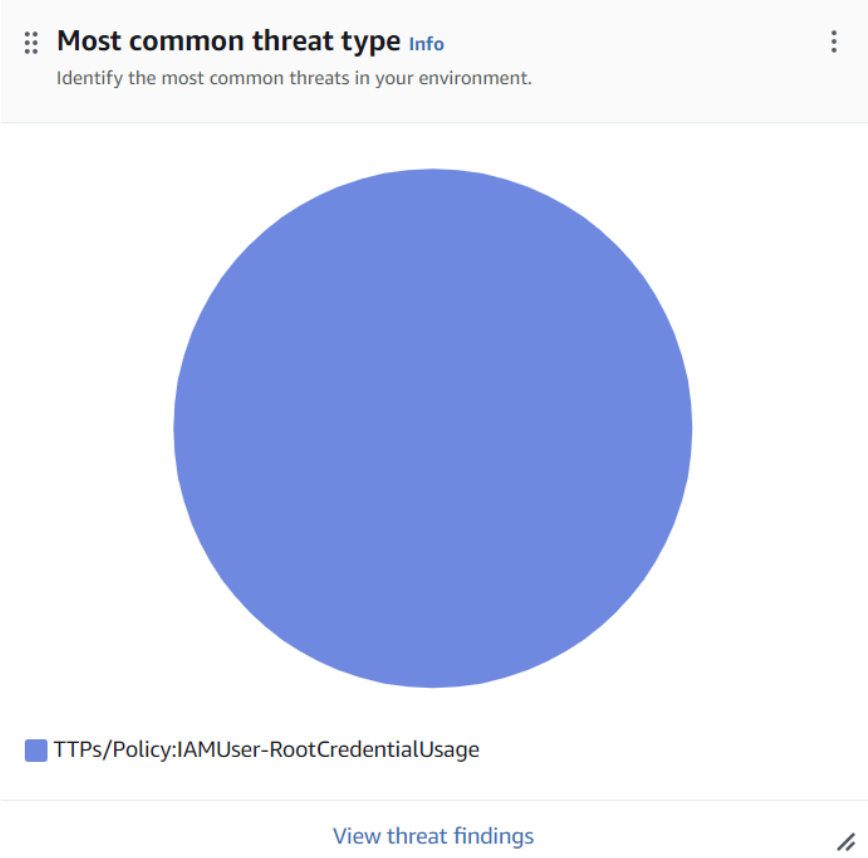

Figure 20. Most Common Threat Before Enabling the CEZT.

The second phase commenced with the removal of all resources instantiated by CloudFormation. Subsequently, the CEZT was activated to initiate enforcement measures. Following this, the identical stack was regenerated using CloudFormation. However, in this iteration, Lambda functions were employed to trigger various custom SSM runbooks outlined in the preceding section. These runbooks were designed to implement the policies delineated earlier onto the resources. Subsequent to this action, a 24-hour delay was observed to allow for the update of Security Hub. This delay was necessitated by the fact that Security Hub conducts security control checks based on the information relayed by AWS Config, which in turn may take up to 24 hours to transmit all configuration changes

to Security Hub. After this period elapsed, the security scores associated with all referenced security standards exhibited a notable increase, as depicted in Figure 21.

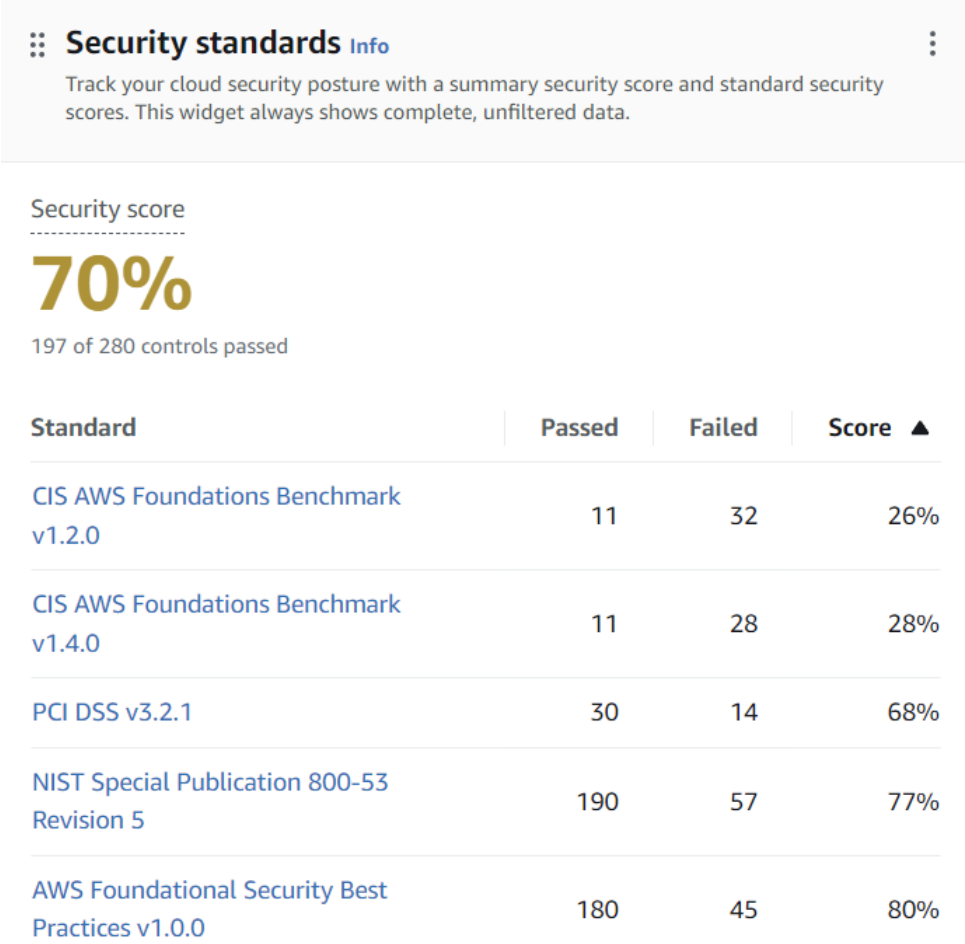

Figure 21. Security Scores measured by Security Hub with the CEZT in Effect.

The security scores for CIS AWS Foundations Benchmarks displayed the least enhancement, whereas significant improvements were observed across all other standards. Notably, NIST Special Publication 800-53 Rev. 5 and AWS Foundational Security Best Practices exhibited the most substantial improvements, achieving scores of 77% (up from 44%) and 80% (up from 50%), respectively. Figure 22 illustrates the threats categorized by severity with the CEZT activated.

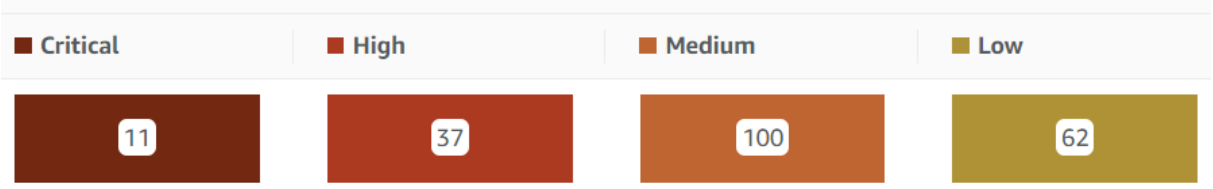

Figure 22. Threats grouped by severity with the CEZT in effect.

The quantity of threats categorized as critical or high risk witnessed a notable reduction; however, there was a contrasting increase in the number of threats classified as medium or low risk. Once more,

broken IAM configuration emerged as the most prevalent threat identified. Table 2 compares the number of security threats grouped by severity.

Table 2. Comparison of Security Scores Before and After Enabling the CEZT.

Security Standard	Security Score with the CEZT disabled (%)	Security Score with the CEZT enabled (%)	Change
CIS AWS Foundations Benchmark v1.2.0	23	26	+3
CIS AWS Foundations Benchmark v1.4.0	23	28	+5
NIST Special Publication 800-53 Rev. 5	38	68	+30
AWS Foundational Security Best Practices v1.0.0	44	77	+33
PCI DSS v3.2.1	50	80	+30
Overall Security	31	70	+39

Table 3. Comparison of Security Threats Findings Grouped by Severity Before and After Enabling the CEZT.

Severity Level	Number of threats with the CEZT disabled	Number of threats with the CEZT enabled	Change
Critical	75	11	-64
High	166	37	-129
Medium	65	100	+35
Low	56	62	+6

When assigning equal weight to all severity levels, the implementation of the CEZT resulted in a reduction of the total number of security threats by 152. However, when applying weights of 4, 3, 2, and 1 to critical, high, medium, and low severities respectively, the CEZT effectively mitigated 719 threats. This substantiates the significant enhancement in security achieved by our model within the enterprise.

While not free, the CEZT model is notably more cost-effective compared to solely relying on AWS managed services. Substituting tools like Pfsence instead of NAT Gateway, AWS Network Firewall,

AWS Certificate Manager (ACM), AWS WAF, and Amazon GuardDuty (utilized solely for the analysis section of the research, not active protection of the enterprise) not only reduces costs but also offers greater flexibility in configuring desired settings.

The semi-SDN micro-segmentation of the enterprise into control plane and data plane with having all the resources isolated from each other, simplifies the enforcement of ZT policies and provides security in case of existing of insider intruders and some resources under their control. In addition, combining the created runbooks with various Lambda functions and CloudWatch Events provides automation, which can in turn decrease the number of human errors. Continuous monitoring of traffic through Pfsense and its Snort-based IDS adds an additional customizable layer of security.

However, it's important to note that this research was limited to a simulated environment, and real-world results may vary. Additionally, the CEZT model currently focuses on a subset of AWS resource types (EC2, S3, RDS, LDR, ASG), potentially leaving other resource types without ZT enforcement.

Future research avenues could include extending the model to cover all major cloud platforms (e.g., AWS, Azure, GCP) and conducting tests on real-world enterprises to validate the model's efficacy. Access to source codes is available upon request.

6 Conclusion

In conclusion, this research positions Zero Trust Architecture as a strategic paradigm for enhancing the security posture of AWS-based infrastructures. Embracing the principles of continuous verification and monitoring and least privilege access empowers organizations to bolster their digital assets against an ever-evolving threat landscape. The findings derived from this study serve as a foundational resource for informed decision-making and the effective implementation of ZTA, contributing to the development of resilient and secure cloud infrastructures.

6.1 Evaluation of Research Questions and Their Resolutions

As a reminder the research questions are highlighted below.

1. How can enterprises enhance the security of their cloud infrastructure to meet the stringent demands of critical infrastructure while also integrating automated security protocols to alleviate pressure on security teams and minimize the risk of human error?

2. How can these security strategies be deployed in a manner that is both cost-effective and adaptable to the diverse needs and resources of organizations?

3. Does adopting a zero-trust approach significantly improve security or impose unnecessary burdens by enforcing overly strict policies without providing additional security?

Regarding the first question, the traditional perimeter-based security approaches, as discussed in Section 2, are ineffective against insider threats. Critical infrastructures can enhance their protection against insider threats by implementing zero trust principles. Automated policy enforcement reduces the burden on the security team and subsequently decreases the likelihood of human errors, since two factors of human error are the nature of forgetfulness or neglection, as discussed in section 2 in one of the recent incidents, which does not happen with machines.

Regarding the second question, the strateg proposed in this research is cost-effective due to the limited reliance on managed services. For example, rather than employing a NAT gateway, which is a managed NAT service offered by AWS, a NAT instance was utilized, which is significantly less expensive. To mitigate DDoS/DoS attacks, a micro-segmentation approach was adopted, incorporating one control plane (public subnet) and multiple data planes (private subnet), as well as auto-scaling groups (ASG) and load balancers, instead of using AWS Shield, a paid managed DDoS protection service from AWS. Since load balancers and auto-scaling groups are commonly employed

in AWS environments, leveraging them within a micro-segmentation framework under the governance of a control plane is unlikely to incur additional costs for the company. It is also customizable thanks to the use of SSM runbooks and custom scripts.

Regarding the third question, based on the analysis, CEZT effectively enhances cloud security across four key security standards by employing zero trust principles. In contrast to perimeter-based security approaches, CEZT acknowledges the potential for insider threats and, therefore, does not trust any entity by default, even those within the network. Entities are required to use certificates issued by the CA server and comply with the stringent zero trust (ZT) policies enforced by the control plane to interact with one another, regardless of their location within the same subnet. Across the four security standards used to measure the security score of the vulnerable network before and after enabling CEZT, including CIS AWS Foundations Benchmarks (v1.2.0 and v1.4.0), PCI DSS v3.2.1, NIST Special Publication 800-53 Rev. 5, and AWS Foundational Security Best Practices v1.0.0, the security score improved ranging from +3% for the former, CIS AWS Foundations Benchmark v1.2.0, to +33% for the latter, AWS Foundational Security Best Practices v1.0.0. In addition, the overal security increased by 39%. After enabling CEZT, although the number of low-level threats increased, +6 for low-level security threats and +35 for medium levels, the number of high-level security threats decreased, -64 for critical security threats and -129 for high-level security threats, which in turn, the network became more secure. The level of security was determined by the mentioned security standards.

6.2 Limitations and Future Work

CEZT is specifically tailored for the AWS platform. While AWS is a prominent and widely recognized cloud service provider, many organizations utilize alternative platforms like Azure. This platform-specific design limits the applicability of CEZT to companies that rely solely on AWS, potentially excluding those who use other cloud services.

Future research could explore the development of a similar model for other cloud platforms or, more ideally, create a universal model that is compatible with multiple platforms. Additionally, this study did not cover all AWS services, indicating that there is further potential to expand the scope of CEZT to include a more comprehensive range of AWS offerings.

References

Abusaimeh, H. (2020). Security attacks in cloud computing and corresponding defending mechanisims. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(3).

Ahlawat, B., Sangwan, A., & Sindhu, V. (2020). IoT system model, challenges and threats. *Int. J. Sci. Technol. Res*, *9*(3), 6771–6776.

AlFardan, N. J., Bernstein, D. J., Paterson, K. G., Poettering, B., & Schuldt, J. C. N. (2013). On the security of RC4 in TLS and WPA. *USENIX Security Symposium*, *173*.

Alouffi, B., Hasnain, M., Alharbi, A., Alosaimi, W., Alyami, H., & Ayaz, M. (2021). A systematic literature review on cloud computing security: threats and mitigation strategies. *IEEE Access*, *9*, 57792–57807.

Al-Shareeda, M. A., Anbar, M., Manickam, S., & Hasbullah, I. H. (2020). Review of prevention schemes for man-in-the-middle (MITM) attack in vehicular ad hoc networks. *International Journal of Engineering and Management Research*, *10*.

Amazon Web Services. (2006). *Shared Responsibility Model*. Https://Aws.Amazon.Com/Compliance/Shared-Responsibility-Model/.

Amazon Web Services. (2024). *fsbp-standard*. Https://Docs.Aws.Amazon.Com/Securityhub/Latest/Userguide/Fsbp-Standard.Html.

Amazon Web Services (AWS). (2024). *cis-aws-foundations-benchmark*. Https://Docs.Aws.Amazon.Com/Pdfs/Securityhub/Latest/Userguide/Securityhub.Pdf#cis-Aws-Foundations-Benchmark.

Barnett, K. (2022). *Yahoo lawsuit alleges employee stole trade secrets upon receiving Trade Desk job offer*. Https://Www.Thedrum.Com/News/2022/05/19/Yahoo-Lawsuit-Alleges-Employee-Stole-Trade-Secrets-upon-Receiving-Trade-Desk-Job.

Brosso, I., La Neve, A., Bressan, G., & Ruggiero, W. V. (2010). A continuous authentication system based on user behavior analysis. *2010 International Conference on Availability, Reliability and Security*, 380–385.

Chen, B., Qiao, S., Zhao, J., Liu, D., Shi, X., Lyu, M., Chen, H., Lu, H., & Zhai, Y. (2020). A security awareness and protection system for 5G smart healthcare based on zero-trust architecture. *IEEE Internet of Things Journal*, *8*(13), 10248–10263.

Cloud Security Alliance. (2013). *Software Defined Perimeter*. http://www.cloudsecurityalliance.org

Cunningham, C. (2018). The zero trust extended (ZTX) ecosystem. *Forrester, Cambridge, MA*.

Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: issues and challenges. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 27–33.

Douligeris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, *44*(5), 643–666.

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., & Halderman, J. A. (2015). *Tracking the FREAK Attack*. Https://Freakattack.Com.

Eliyan, L. F., & Di Pietro, R. (2021). DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges. *Future Generation Computer Systems*, *122*, 149–171.

Embrey, B. (2020). The top three factors driving zero trust adoption. *Computer Fraud & Security*, *2020*(9), 13–15.

Extreme Networks. (2015). *VN 2015 003 Freak - Factoring Attack on RSA-Export Keys*.

Gluck, Y., Harris, N., & Prado, A. (2013). BREACH: reviving the CRIME attack. *Unpublished Manuscript*.

Goyal, R., Dragoni, N., & Spognardi, A. (2016). Mind the tracker you wear: a security analysis of wearable health trackers. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 131–136.

Gupta, A., Sinha, S., Singh, H. K., & Bhushan, B. (2023). Vulnerability Assessment of Security Breach and Deadly Threat in Cloud Computing Environment. *2023 IEEE 15th International Conference on Computational Intelligence and Communication Networks (CICN)*, 433–442.

Gurucul. (2023). *Insider Threat Report*.

He, Y., Huang, D., Chen, L., Ni, Y., & Ma, X. (2022). A survey on zero trust architecture: Challenges and future trends. *Wireless Communications and Mobile Computing*, *2022*(1), 6476274.

Hossain, M. D., Ochiai, H., Doudou, F., & Kadobayashi, Y. (2020). Ssh and ftp brute-force attacks detection in computer networks: Lstm and machine learning approaches. *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, 491–497.

Kerner, S. M. (2015). *Logjam SSL/TLS Vulnerability Exposes Cryptographic Weakness Retrieved August 10, 2015*.

Kindervag, J. (2010). Build security into your network's dna: The zero trust network architecture. *Forrester Research Inc*, *27*, 1–16.

Kumar, K. (2017). *Micro-segmentation of Applications using Application Rule Manager*. Https://Blogs.Vmware.Com/Networkvirtualization/2017/04/Microsegmentation-Arm.Html/.

Kumar, S., & Chandavarkar, B. R. (2021). DDOS prevention in IoT. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–6.

Mantin, I. (2015). Bar-Mitzva Attack: Breaking SSL with 13-Year Old RC4 Weakness. *Black Hat Asia*.

Mishra, N., & Pandya, S. (2021). Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, *9*, 59353–59377.

Mohammed, C. M., & Zeebaree, S. R. M. (2021). Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review. *International Journal of Science and Business*, *5*(2), 17–30.

Möller, B., Duong, T., & Kotowicz, K. (2014). This POODLE bites: exploiting the SSL 3.0 fallback. *Security Advisory*, *21*, 34–58.

Nadeem, M., Arshad, A., Riaz, S., Band, S. S., & Mosavi, A. (2021). Intercept the cloud network from brute force and DDoS attacks via intrusion detection and prevention system. *IEEE Access*, *9*, 152300–152309.

Newman, R. (2014). *Taming the B.E.A.S.T.*

NIST. (2020). *Security and Privacy Controls for Information Systems and Organizations* [National Institute of Standards and Technology]. https://doi.org/10.6028/NIST.SP.800-53r5

PCI Security Standards Council. (2018). *PCI DSS Quick Reference Guide*. Https://Www.Pcisecuritystandards.Org/Pdfs/Pci_ssc_quick_guide.Pdf.

Peterson, M. (2024). *AWS Security Blog*. Https://Aws.Amazon.Com/Blogs/Security/Aws-Plans-to-Invest-E7-8b-into-the-Aws-European-Sovereign-Cloud-Set-to-Launch-by-the-End-of-2025/.

Ponemon Institute. (2020). *Cost of Insider Threats*. Https://Www.Proofpoint.Com/Cost-of-Insider-Threats/.

Ponemon Institute. (2022). *Cost of Insider Threats Global Report*. Https://Static.Poder360.Com.Br/2022/01/Pfpt-Us-Tr-the-Cost-of-Insider-Threats-Ponemon-Report.Pdf.

Popli, M. (2019). A survey on cloud security issues and challenges. *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, 230–235.

Prabadevi, B., & Jeyanthi, N. (2018). A review on various sniffing attacks and its mitigation techniques. *Indones. J. Electr. Eng. Comput. Sci*, *12*(3), 1117–1125.

Rani, D., & Ranjan, R. K. (2014). A comparative study of SaaS, PaaS and IaaS in cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *4*(6).

Rescorla, E. (2018). *The transport layer security (TLS) protocol version 1.3*.

Roos, A. (1995). Weak keys in RC4. *Sci. Crypt Post*, *22*.

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero Trust Architecture*. https://doi.org/10.6028/NIST.SP.800-207

Rowe, A. (2023). *Third MailChimp Data Breach Makes It Hard To "Rebuild Trust."* Https://Tech.Co/News/Mailchimp-Breach-Phishing-Trust.

SafetyDetectives Cybersecurity Team. (2022). *Turkish Based Airline's Sensitive EFB Data Leaked*. Https://Www.Safetydetectives.Com/News/Pegasus-Leak-Report/.

Saharan, S., & Gupta, V. (2021). DDoS prevention: review and issues. *Advances in Machine Learning and Computational Intelligence: Proceedings of ICMLCI 2019*, 579–586.

Sarkar, P. G., & Fitzgerald, S. (2014). Attacks on SSL a Comprehensive Study of BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 BIASES (2013). *Internet: Https://Www. Isecpartners. Com/Media/106031/Ssl_attacks_survey. Pdf*.

Satapathy, A., & Livingston, J. (2016). A Comprehensive Survey on SSL/TLS and their Vulnerabilities. *International Journal of Computer Applications*, *153*(5), 31–38.

Shaji, E., & Subramanian, N. (2021). Assessing Non-Intrusive Vulnerability Scanning Methodologies for Detecting Web Application Vulnerabilities on Large Scale. *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, 1–5.

Smyth, B., & Pironti, A. (2013). Truncating {TLS} Connections to Violate Beliefs in Web Applications. *7th USENIX Workshop on Offensive Technologies (WOOT 13)*.

Sunyaev, A., & Sunyaev, A. (2020). Cloud computing. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, 195–236.

Teerakanok, S., Uehara, T., & Inomata, A. (2021). Migrating to zero trust architecture: Reviews and challenges. *Security and Communication Networks*, *2021*(1), 9947347.

Tewari, A., & Gupta, B. B. (2020). Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework. *Future Generation Computer Systems*, *108*, 909–920.

U.S. Department of Justice. (2022). *United States v. Paige Thompson*. Https://Www.Justice.Gov/Usao-Wdwa/United-States-v-Paige-Thompson.

U.S. Federal Trade Commission. (2022). *Equifax Data Breach Settlement*. Https://Www.Ftc.Gov/Enforcement/Refunds/Equifax-Data-Breach-Settlement.

Uttecht, K. D. (2020). Zero Trust (ZT) concepts for federal government architectures. *Department of Homeland Security (DHS) Science and Technology Directorate (S&T), Lexington, Massachusetts*.

Vanickis, R., Jacob, P., Dehghanzadeh, S., & Lee, B. (2018). Access control policy enforcement for zero-trust-networking. *2018 29th Irish Signals and Systems Conference (ISSC)*, 1–6.

Vidal, C., & Choo, K.-K. R. (2018). Situational crime prevention and the mitigation of cloud computing threats. *Security and Privacy in Communication Networks: SecureComm 2017 International Workshops, ATCS and SePrIoT, Niagara Falls, ON, Canada, October 22–25, 2017, Proceedings 13*, 218–233.

Ward, R., & Beyer, B. (2014). Beyondcorp: A new approach to enterprise security. ; *; Login:: The Magazine of USENIX & SAGE*, *39*(6), 6–11.

Yan, X., & Wang, H. (2020). Survey on Zero-Trust Network Security. In X. Sun, J. Wang, & E. Bertino (Eds.), *Artificial Intelligence and Security* (pp. 50–60). Springer Singapore.

Yao, Q., Wang, Q., Zhang, X., & Fei, J. (2020). Dynamic access control and authorization system based on zero-trust architecture. *Proceedings of the 2020 1st International Conference on Control, Robotics and Intelligent System*, 123–127.

Zhijun, W., Wenjing, L., Liang, L., & Meng, Y. (2020). Low-rate DoS attacks, detection, defense, and challenges: a survey. *IEEE Access*, *8*, 43920–43943.

Zoller, T. (2011). TLS/SSLv3 renegotiation vulnerability explained. *G-Sec (University of Luxembourg)*.