CAPSTONE PROJECT REPORT
JUNE 27th 2023

Authors: Adeel Awan, Keegan Dasilva Barbosa, Robin Gaudreau

1 Introduction

This report outlines the design decisions and cyber security considerations that come with the implementation of a highly available cloud computing platform for an organization running a web application with a public-facing page, a computation layer, and an online database. We opted to view this project from a car dealership's perspective. Our reasoning is two fold. One, we could give context to the project and better understand what assets we were protecting (customer PI) and further lean into Zero Trust with this in mind. Two, auto dealerships are a high priority target for cyber criminals due to their lax security. We wanted to showcase that, despite being cyber-security newbies, we are prepared for industry level security problems.

Throughout this report, we use the list from the Security and Privacy by Design Principles (S|P) as defined by Secure Control Framework Council to analyze the security of our architecture. We were not able to implement all the principles due to the limitations of this project, mainly the lack of actual employees, an actual hybrid environment, and a limited time frame. Despite this, we were able to liberally apply the Zero Trust framework throughout our architecture. See Appendix A for a comprehensive list of inapplicable (S|P) design principles.

The remaining principles are mentioned as footnotes throughout the report as they are implemented. This report is designed as follows. The body of the report will consist of four sections. The introduction will continue below with our network design. Section 2 will focus on the implementation of a comprehensive Zero Trust architecture, highlighting how each element of our design was purposefully built to align with the principles of Zero Trust. Section 3 will contain our Tenable scan reports, and our analysis of the results. Section 4 will be a reflection on what we could have and would have done differently, having now completed the project.

1.1 Network Design

The information technology resources of our organization are divided into three sectors. The first one is the cloud-based production environment, consisting of our cloud hosted web application. The second is the cloud-based non-production environment, consisting of a collection of virtual machines that are used by remote employees. The third sector is the on-premises computing resources that are used by the programming team, and for the purposes of this project, are simulated by a single EC2 instance. Throughout the report, we will refer to this diagram (alternatively see Appendix B) which serves as the blueprint for our design.

1.2 Infrastructure as Code (IaC)

We implemented IaC at every instance of deployment and testing by liberally implementing Terraform.[1] In particular, Terraform was responsible for creation of VPCs, EC2 instances, security groups, customer gateways, transit gateways, load balancers[2], VPN connections, ACLs and routing. Terraform was not used for IAM (given we are assuming the role of an SMB, simpler through console), key generation (as this is less secure and harder to configure than through the AWS console), and account management (for the same reasoning as IAM).

All but the virtual assets disposition is handled securely. AWS does not guarantee the secure destruction of data until their physical devices are decommissioned.[3] Unfortunately, we could not truly implement crypto-shredding, because the AWS backup system does not guarantee that backups are saved at a consistent location.

[1] **AAT:** Artificial Intelligence and Autonomous Technology
[2] **CAP:** Capacity and Performance Planning
[3] **AST:** Asset Management

2 Zero Trust Architecture

Zero Trust is a security framework that assumes no inherent trust between users, devices, or networks, requiring strict authentication, authorization, and continuous verification for all access attempts. The incessant use of the term, especially by marketers, might suggest Zero Trust is more of a marketing gimmick than a tried and true framework. This simply is not the case, as noted by both Forrester and NIST research reports. President Biden even made an executive order to push the United States governments towards having a fully Zero Trust architecture for the purpose of national security. Given the United States Federal government is postured as one of the strongest cyber security institutions in the world, it is undeniable that authentication based security is the modern standard of excellence. In this section, we emphasize how the Zero Trust ethos arises in each and every aspect of our design.[4]

2.1 A Zero Trust Network Infrastructure

Zero Trust has three core principles. Least privilege, always verify, and assume breach. A true Zero Trust architecture ought to have these principles exercised. Given that the entire network is partitioned into three segments, we will explain exactly how the core principles arise across each fragment. [5]

2.1.1 The Production VPC

The production environment is a three-tier web application deployed over two availability zones. Given we were tasked with making the presentation layer double as a jump box, we only allowed SSH access to the jump box from the on-premises device. We do this with all devices in the environment, ensuring SSH is only whitelisted by at most one device on the network. We also allow internet access to the front facing layer, as it needs to be publicly accessible. Unfortunately, we needed to leave port 80 open as a requirement for the project. Our design ensures bare minimum access for our devices to function, liberally applying the principle of least privilege.

Securing the database layer is crucial to safeguarding sensitive data and ensuring the integrity of the application. This is where we apply the principle of assume breach to this environment. We isolated the database layer in a private subnet within the VPC to limit exposure to external threats. We encrypted the data at rest with AES-256 encryption algorithm for the database storage volumes using AWS Key Management Service (KMS) to protect data in case of unauthorized access or data breaches. We ensure the database is encrypted in transit by ensuring HTTPS communication with the AWS API. Since AWS provides encryption on L1 and L3, it suffices to ensure L7 encryption using TLS.

[4] **NET:** Network Security
[5] **SEA:** Secure Engineering and Architecture

2.1.2 The On Premises VPC

Our on premises environment is highly restrictive, and where we believe we excelled at implementing Zero Trust. Not much happens in the public subnet, as it only hosts a customer gateway, and a NAT for secure and private internet access for our on premises instance. We only allow the on premises device to SSH to the other environments. Our crowning achievement was closing incoming SSH traffic from our on premises device. Access to the device can only be achieved by sessioning in, which requires the correct IAM credentials. Like this, we are applying both the principle of least privilege, and always verify. Moreover, our IAM policy is designed to forward logs to CloudTrail. This ensures the on-premises is always monitored while active, applying the principle of assume breach. Our on-premises environment meets the Zero Trust trifecta, and is maximally optimized for security purposes.

2.1.3 The Non-Production VPC

The non-production environment consists of 5 EC2 instances, all isolated into separate private subnets to limit possible lateral movement of an attacker (inline with assume breach). The principle of least privilege is applied here as well, with sessioning and SSH from on-premises being the only means of accessing the devices. We setup monitoring for these devices using CloudTrail.[6] [7]

2.1.4 The Transit Gateway

The current architecture relies on a single transit gateway connection between the different VPCs. Communication between production and non-production environments is disallowed by the gateway in accordance with the principle of least privilege. We tested the connection between the on-premises environment and the non-production and production environments using the reachability analyzer. We implemented site-to-site VPN encryption to guarantee the confidentiality of cross VPC communication.[8]

2.2 IAM

It is imperative that an organization establishes well defined roles for employees. For one, this allows for the segregation of duties and least privilege access.[9] Two, narrowing down access makes incident response easier, ensuring that the right individuals with the necessary expertise and authority are responsible for addressing security breaches or vulnerabilities. For the purposes of this project, we tried to enforce strict authorization

[6] **THR:** Threat Management
[7] **MON:** Continuous Monitoring
[8] **CFG:** Configuration Management
[9] **CHG**: Change Management

rules for how our AWS accounts are accessible. We did this by choosing the *correct* type of MFA, and securing the root user.

2.2.1 MFA

Authentication of employees is managed by AWS IAM and uses MFA for all members of the organization. The authentication factors are a password and a physical MFA device for employees. We opted to use Duo for MFA, as it is vastly more secure than text verification. This ensured the integrity of our sign-ons, whereas any SMS MFA system could not guarantee this, as SMS is sent in open airwaves through plaintext.

2.2.2 Securing the Root

The root user for the AWS account is secured and is only shared with two members of the team. We chose a strong password to prevent brute force attacks from potential adversaries. After initializing accounts for our team, no task was implemented from the root user account. Access keys for the root user were not created, hence that account does not have programmatic access. We secured the AWS accounts by disallowing account access with root user credentials, as per AWS best practice.

2.3 Key Management

Key management is crucial to Zero Trust architecture, as keys determine who has access to what device. Our current setup has a total of 9 RSA keys.[10] A backup on premises key that was mostly used for troubleshooting (would be deleted in preference of sessioning), one key for each non-production VPC, a presentation layer key, an application layer key, and a key to access the database. AWS manages the keys to ensure there is no bias in key generation, and to outsource general key management. Access to keys is controlled by IAM policies. In a real setting, we would opt against keys being used at all by non-production, and only allow access to those computers via sessioning. Alternatively, we would allow even more granular access to keys, though sessioning is a purely identity based method of access and more in line with the Zero Trust ethos.

3 Reports

In an industry setting, there would be dedicated cybersecurity professionals with the task of periodically monitoring the environment for cyber threats and vulnerability, via a variety of scans. These range from malware detection to unpatched software. For automated cloud security scanning, Tenable is the industry standard, hence this was mandated for the project. We decided that it was best to go beyond just Tenable. Given the environment is hosted by AWS, it only seems natural to utilize AWS native scanning tools. Hence, we did

[10] **CRY:** Cryptographic Protections

in-house scans with AWS Trusted Advisor, AWS Inspector, and AWS Security Hub. Given Terraform was integral to production, it only seemed natural that we implement Policy as Code (PaC). Using a free Bridgecrew trial, we were able to scan our Terraform code to determine where our infrastructure was most vulnerable.

3.1 Tenable Reports

3.2.1 Internal Agent Scan

Using agents installed on the EC2 instances in the production environment yields a rather tame report which recommends that software and operating systems be updated and patched. It also showed that, using port 22, a variety of properties could be enumerated. This suggests that, despite its restrictive access, a compromised jump server could allow attackers to move laterally into our application layer. Note that access to port 22 of the jump server is only accessible by the on-premises environment, so attackers would have to be really clever to exploit this.

3.2.2 External Scanner Scan

A Tenable scanner was downloaded onto the on-prem device and used to scan the non-production environment. We took a scientific approach to verifying the strength of our design. During the first basic scan, every device has all ports open. This served as a "control group" to be tested against. Our second scan was an advanced scan and had one device with its security setup as described in Section 2, and the other devices had the control group settings. Unintuitively, both scans are nearly identical. The key observation we took from this report was that SHA-1 HMAC was enabled, which could be exploited by an attacker to ruin the integrity of our SSH connections. The device with the oldest operating system had weaker key exchange algorithms enabled, which the scan reported as low severity. We were unable to see from the external scan that our devices were intentionally left unpatched, which we address in the next assessment.

3.2 AWS in-house Security Analysis

Due to the sparsity of information provided by the Tenable reports, we elected to use the AWS security analysis services.[11] These included AWS Config[12] (required for the operation of Security Hub), Trusted Advisor, Inspector, Security Hub, and GuardDuty. The latter monitors ongoing activity and automates security response[13]. Since the environment was only running for a short time, GuardDuty provided no insight into its security.

[11] **PRM:** Project and resource management
[12] **TDA:** Technology Development and Acquisition
[13] **OPS:** Security operations

3.2.1 Trusted Advisor

Trusted Advisor recommended restricting which IP addresses can connect to the open ports. It also found that we were properly using MFA on the root user, IAM for all users, and that none of our storage or computing resources were at over 80% capacity.

3.2.2 AWS Inspector

Inspector [page1, page2] mostly recommended that we update our operating systems, and gave us a low risk warning for ports 80 and 443 being open. What was most astonishing about the report however, is 80% of the data comes from a single Ubuntu machine. Two machines in non-production could not be scanned by AWS due to compatibility reasons. The remaining two machines in the environment, that were not running Ubuntu, ran on AWS Linux OS. Most vulnerabilities were about patching, and the lack of patching that was done (we did not patch machines in hopes that the scan would recognize this, unlike tenable).[14] The report also recommends updating certificates inside the Ubuntu machine, as they are out of date.

3.2.3 AWS Security Hub

AWS Security Hub provides a security score for an account comparing it to a variety of security standards. Those include AWS Foundational Security Best Practices v1.0.0, CIS AWS Foundations Benchmark v1.2.0, and NIST Special Publication 800-53 Revision 5. Security Hub also recommends ongoing logging and monitoring of changes to the deployed environment.

3.3 Bridgecrew Report (Bonus)

We decided to have our Terraform code scanned by Bridgecrew, given we had access to a free trial. Our findings were interesting. According to this report, if we did not have to leave port 22 open for the on prem environment (only did so for the purpose of Tenable scan), hard coded logging into our terraform code, and put implicit denials in our VPC security, we would have met a NIST benchmark.[15] The full report suggests we would have been completely NIST compliant had we closed port 80 on all devices, another project related restriction.[16]

[14] **VPM:** Vulnerability and Patch Management
[15] **RSK:** Risk Management
[16] **CPL:** Compliance

4 Conclusion

One of the largest flaws of our design is the allowing of http traffic. Due to the restrictions of this project, we could not acquire a domain nor a security certificate, which would have allowed for us to explicitly require https traffic via port 443 to all our internet facing instances. Amazon Route 53 could also be utilized to assign a domain name to the public-facing web page. Our architecture was also flawed from the start, by having our presentation layer enforced to also act as a jump server. This is totally contrary to the ethos of Zero Trust. In our final section, we outline the risks associated with our architecture, and explain what we would have done instead. However, we believe that overall, we have created a very secure minimum viable product that would be feasible for an SMB.

4.1 The Jump Server

The architecture of the on-prem environment, with its use of the jump server, had been mandated for the project. However, it is clear from the research that we have done [citation1, citation2, citation3] that this is a vulnerability flaw. Not only is the use of a jump server a somewhat antiquated model, which means that attacks on it are well-known, but it is at its core not compatible with the Zero Trust ethos. We even saw from our Tenable report that a lot of enumeration could be achieved using port 22 being open, despite how restrictive access to the port was.

A key part of Zero Trust is also the principle of least privilege. Taken to its extreme, this includes least privilege for attackers i.e minimizing attack surface. A publicly facing EC2 instance with free internet access that sits directly between our private servers is a blaring design flaw in the eyes of Zero Trust and least privilege.

4.1.1 A Static Bucket

A solution that would reduce the vulnerability of the production environment would be to have a static website hosted in an S3 bucket. Since the bucket is not a computing instance, it has less privileges, and cannot be used to SSH into the application layer. Instead, that access could be done securely from on-premises. S3 buckets are also where Amazon got its start, so the features are way more advanced than with EC2 instances. Buckets are much more available, cheaper, and more secure. This is not a perfect solution however. Buckets cannot host dynamic websites, and we still need to ensure malicious code could not be inserted into the web service.

4.1.2 A Load Balancer

Similar to the bucket solution, we could replace the front facing presentation layer with a load balancer. This serves a similar line of defense in that a load balancer is “dumber” than an EC2 instance. Ofcourse, we would now have to have the application layer serve as both a web layer, and an application layer. We'd rather maintain 3 tiers for segregation of duties, as is the Zero Trust way. Moreover, load balancers are still susceptible to man in the middle attacks and session forwarding when placed on the public facing end. Hence, we'd lean towards the bucket before this choice.

4.1.3 Go Serverless

We also have the option of going serverless. Such an architecture can massively reduce the standard attack surface as the underlying infrastructure becomes abstracted. Moreover, this could be a more cost-effective option than using EC2 instances depending on the traffic. It also allows for the hosting of dynamic websites, unlike an S3 bucket. However, this would restrict access to logging and monitoring, and require stronger access controls and code review. Moreover, malicious actors would no longer have to attack us with malware or stealing data. Instead, sending an influx of traffic would send our costs through the roof and cause a serious financial debt. This is something Amazon learned the hard way with the serverless model, when standard user traffic cost them 10 times more than they could have spent on a simpler model.

4.1.4 Just Don't Jump

The final option we considered was simply not jumping. Keep the front facing presentation layer a public facing EC2. We could have our cake and eat it too, allowing for dynamic websites, trade variable costs for fixed secure rates, and remain highly available to users. Instead, access to application layers would be done either directly by SSH from on-premises, or by sessioning (our absolute favorite with respect to Zero Trust). We think this solution provides the cleanest and healthiest balance of the CIA triad. Moreover, it is the easiest to implement from our current Terraform script. Simply close port 22 on the presentation layer.

4.2 Better IAM

Due to time limitations, we did not segregate our duties nearly as well as we wished we could. Given Adeel was responsible for initial account access and root user controls, Robin was responsible for checking connectivity/logging, and Keegan was responsible for network deployment, it is not as though roles were not clearly defined within our group. However, we did not take privileges from our IAM account, and mostly worked with administrator-like privileges. Had this not been a month long project where we had to learn from the ground up, we would have administered these roles and removed privileges and access from the very start.

4.3 Excessive Extras

We debated on further implementing zero trust by applying AWS Lattice. This would allow us to work on VPCs separately, and connect them with Lattice. This is slightly outside the scope of the project, as Lattice does not need a transit gateway to establish the connection. However, this connection method is more inline with the Zero Trust ethos.

Also outside the scope of the project, we considered creating backups for different cloud platforms. Currently, we only rely on the use of different availability zones.[17] This past June 2023, AWS services were down for multiple companies in North America. In another incident, Google Cloud Services was heavily criticized for downsizing a customer with what the customer claimed was no notice. A backup on a different cloud provider would both serve as disaster control, and an instance of Zero Trust taken to the extreme. We should not presume our third party vendors will always remain reliable.

4.4 Final Thoughts

In conclusion, implementing a defense-in-depth strategy is crucial to safeguarding sensitive data and resources.[18] By leveraging security groups, NACLs, sessioning, and logging via Amazon CloudFront, we created multiple layers of protection against a wide range of security threats. We appropriately segmented the network for least privilege access, were responsible with key management, and created architecture that was *almost* NIST compliant according to Bridgecrew. While not perfect, we are proud of our Zero Trust architecture. We hope this report demonstrates to the reader our industry-ready professionalism in the field of cybersecurity by showcasing our comprehensive understanding and implementation of the Zero Trust framework.

[17] **BCD:** Business Continuity and Disaster Recovery
[18] **CLD:** Cloud Security

Appendix A: Unapplicable S|P

GOV: Cybersecurity and Privacy Governance (no humans)

DCH: Data Classification and Handling (treating everything as equally sensitive)

EMB: Embedded Technology (outsourced to AWS)

END: Endpoint Security (no premises)

HRS: Human Resources Security (no humans)

IRO: Incident Response (no humans)

IAO: Information Assurance (it is impossible for us to be impartial. This would need to be outsourced)

MNT: Maintenance (too short lived)

MDM: Mobile Device Management (no humans)

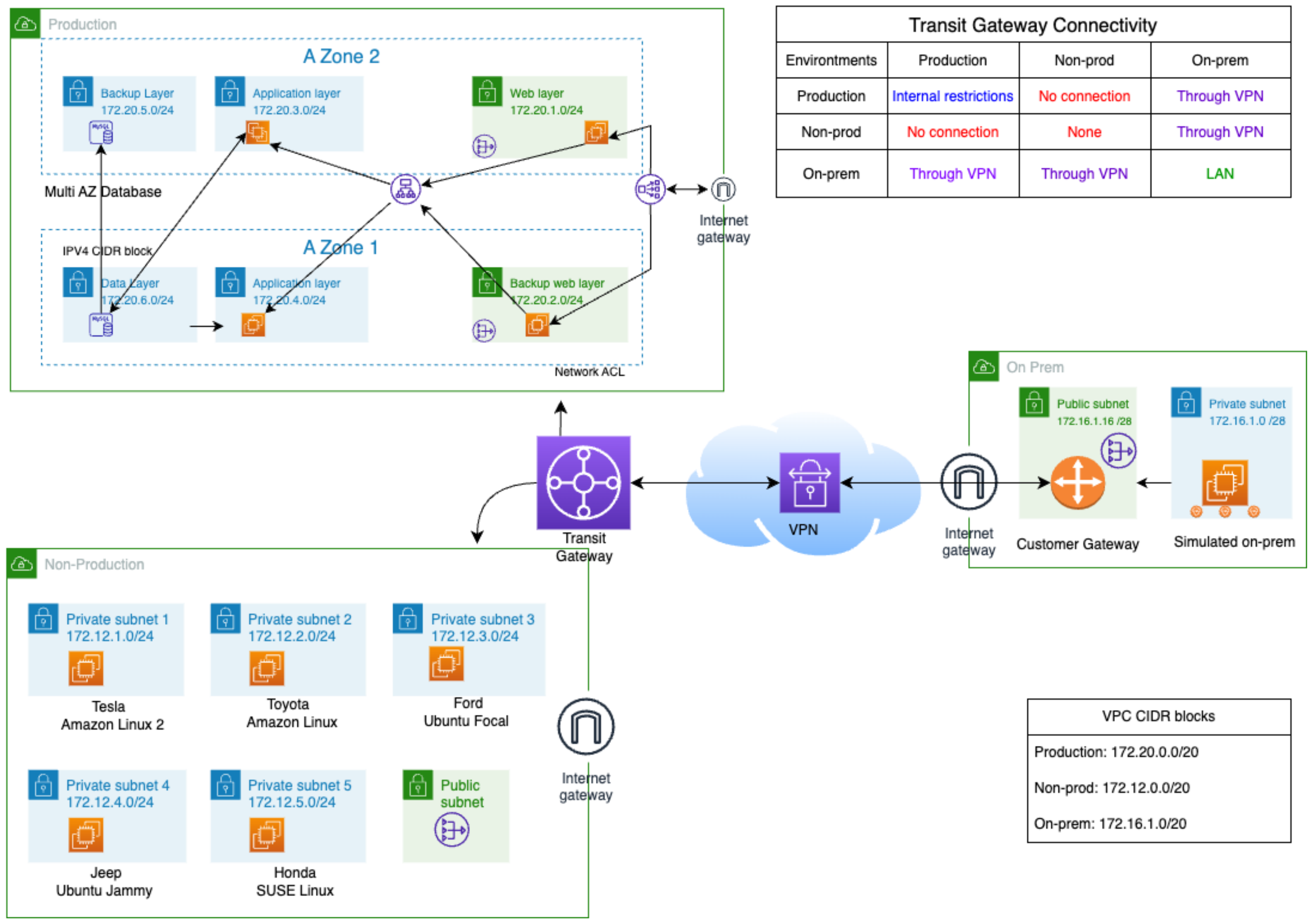
PES: Physical and Environmental Security (no premises)

PRI: Privacy (no humans)

SAT: Security Awareness and Training (no humans)

TPM: Third-party management (mandated by the project's description)

Appendix B: Diagram



Transit Gateway Connectivity			
Environtments	Production	Non-prod	On-prem
Production	Internal restrictions	No connection	Through VPN
Non-prod	No connection	None	Through VPN
On-prem	Through VPN	Through VPN	LAN

VPC CIDR blocks
Production: 172.20.0.0/20
Non-prod: 172.12.0.0/20
On-prem: 172.16.1.0/20