1) What are we building and what are our assets?

We are building a deployable web app that is to be used for training the Red Team. It's a basic architecture that should allow for secure connection for our team and allow for them to train against a vulnerable web application. Though the application is vulnerable by design, the focus is to ensure our architecture is secure enough that if the public facing app is exposed, that our cloud infrastructure has the security in place to prevent any sort of breach.

ASSETS:
1 Load Balancer
3 VMs
1 Network Security Group
1 Virtual Network
Docker Containers with the DVWA Application
1 Availability Set
Web App Portal
Ubuntu 18.04
php 7.0
MySQL DB

2) What can go wrong?

Security Design Principles:

In reviewing this question, We believe we have some *fail-safe* defaults. The first of which is we have a jumpbox allowing remote access into our virtual system(s). If this jumpbox is compromised our system fails. From this context, I am approaching further investigation of Threats and Security Properties:

Spoofing

The infrastructure is vulnerable to DNS spoofing and malicious users can spoof the training app the Red Team is using.

- Since we have a public IP address within our Azure network, our network is vulnerable to I*P Address spoofing* and *DNS spoofing*.
- Our public IP address has SSH access to the Jumpbox server. If an attacker is able to spoof our public IP address, they can gain the ability to SSH to the Jumpbox server.
- An attacker can also redirect traffic to an alternate website by hijacking our DNS entry. This is most commonly carried out by cache poisoning (e.g. when an attacker injects a forged entry into the DNS server).
    - If the appropriate DNS spoofing detection software is not implemented, such as XArp, thus making our website vulnerable to this sort of attack.
    - We should ensure our web app uses end-to-end encryption which will help decrease the chances of being compromised.
    - We should also use digitally signed DNS records to help determine data authenticity.

Tampering

Since the DVWA is a web application, it is generally vulnerable to tampering attacks such as *cross site scripting (XSS)* and *SQL injection*.

Our web application has several forms and so we must be careful to deny JavaScript, HTML, Flash, or any other type of code that the browser may be able to execute. We should perform a security review of the code and search for all places where input from an HTTP request could possibly make its way into the HTML output.

Using SQL Injection, an attacker can gain access to the most valuable asset - user and application data. This is a very serious threat that can cause repudiation issues such as voiding transactions or changing balances, or allowing the complete disclosure of all data on the system. Our web app uses PHP, which is especially vulnerable to such attacks due to the prevalence of older functional interfaces.

Information Disclosure

From this compromised terminal, much accessible private data can breach the Elk box system. Passwords which are core to the key remote entry point can be compromised. From this point, further exploitation of the system can occur. While establishing our Elk Stack is foundational, we should organizationally consider securing our passwords for our terminals we access into our jump boxes. Our current access control and passwords consist of :
--Terminal: non encrypted with simple password to unlock terminal

Compromising our terminal will compromise our entire system as this is a key access point outside of our Azure environment.

Next, as we access our system, we are utilizing a jumpbox:
--Jump box: With encryption with single passphrase without hierarchical administrative password control to force passphrase resets.

We will not have any way of knowing if this access point is compromised.

Finally, our 2 VM Boxes are wide open
--2 VM Boxes: With white listed IP Addresses from Terminal and Jump Box

Here if the first 2 are compromised, we will grant complete access to our assets and data.

Denial of Service

Azure has some advantages related to preventing Dos attacks due to its sheer size and scale regarding Absorption, Detection, and Mitigation. Absorption happens before detection, and detection happens before mitigation. Absorption is the best defense against DoS attacks. If the attack can't be detected, it can't be mitigated. But if even the smallest DoS attack can't be absorbed, then services aren't going to survive long enough for the attack to be detected. In this context, we have some issues regarding our rate of Absorption:
--One Load Balancer
--Two (2) VMs with containers

A coordinated Dos Attack will quickly overwhelm our capacity of absorption of an aggressive attack. To counteract our limited budget regarding our infrastructure budget, we should consider time-to-detection decreasing through increased monitoring at key points in our system:
--One Load Balancer
--Jump Box

A TechRepublic study found that misconfigured databases and services have resulted in the exposure of more than 3.2 billion records so far in 2019[1]. Additionally, there is a need to decrease our attack surface by reducing the number of open ports. To this point, Currently, we have the following open ports:
Jumpbox

Jumpbox NSG
(Ports 22: 80)

VM1
VM1 NSG
Inbound

VM2
VM2 NSG
Inbound
Destination

Additionally, with the above, we should settle on optimal NSG rules for each of the above ports which we currently do not have implemented. The plan to monitor to decrease our time-to-detection would be determined by consensus of above.

Escalation of Services

Our current architecture allows the greatest flexibility to meet demands for our organization. In the short span of a single week, we are already shifting into yet another architecture solution. Despite this ease of adaptability, we do have an issue with our issue with gaining access to our system and escalation privileges/services. We currently do not have a tiered restriction in place for:

Tier 0 Administrators who manage the identity store
Tier 1 Administrators, who manage enterprise servers, servers, and applications
Tier 2 Administrators, who manage devices like desktop, laptops, and printers

We essentially have a system if compromise immediately leads to entire ownership by the attacker. Additionally, another threat has recently emerged which does not have a patch issued.

DLL side-loading, as it's labeled in the MITRE ATT&CK framework, can happen when programs "improperly or vaguely specify a required DLL." As a result, they may be open to a vulnerability in which an unintended DLL is loaded into the program. Attackers can take advantage of legitimate programs vulnerable to side-loading to load a malicious DLL and mask any malicious actions they take under the guise of a trusted system or process.

To run RDP, users access the MSTSC in Windows to take control of a remote computer or virtual machine using a network connection, the researchers explain in a blog post. MSTSC relies on a DLL file (mstscax.dll) as one of its resources. Researchers learned MSTSC performs delay-loading of mstscax.dll with a behavior that can lead to attackers slipping past security controls. The executable loads "mstscax.dll" with no integrity checks to validate the library's code, they say.

An attacker could replace the DLL mstscax.dll in the folder c:\windows\system32, which requires local administrative privileges. In another scenario, an attacker could copy mstsc.exe to an external folder, placing the DLL in the same folder and running mstsc from there. This does not require admin privileges, Microsoft says the mstsc should not be used outside the folder c:\windows\system32; however, this is not enforced.

Both scenarios let an attacker bypass security controls because malicious code runs under the context of mstsc.exe, which is a Microsoft-signed executable. The first scenario will allow attackers to persist, he adds, because mstsc.exe will run the malicious code every time it's used[2].

f. Elevation of Privileges

Since we are reliant on Azure as our cloud provider, this is a big weak point in our network security model. If attackers are able to bypass Azure security, then by default, they will gain elevated privileges to our network.

In a research performed by CheckPoint Research, analysts concluded Azure Stack can be compromised by exploiting minor oversight in their software design:

- Azure does not require authentication for some requests
- Azure does not validate some requests.

https://research.checkpoint.com/2020/remote-cloud-execution-critical-vulnerabilities-in-azure-cloud-infrastructure-part-i/

[1] Isc2-cloud-security-risks.pdf

[2] https://www.darkreading.com/endpoint/researchers-use-microsoft-terminal-services-client-in-new-attack-method/d/d-id/1337614?_mc=bib&itc=bib

Single load balancer failure could open system up to flooded requests

Single availability set running both VMs.

3) What are we going to do about the issues we identified?

1) SSH security: We currently have Port 22 as our default ssh port. We can mitigate by changing our default SSH port to a random port of our choosing. This will require Administrative action of letting our team know about this change, as well as ensuring that we set the proper inbound rules over that port. We can config this in the sshd_config file and restart sshd via systemctl. We can also set our sshd_config file to only allow for root users to change it.

2) Single point of failure for load balancing: We have redundancy for our machines behind the load balancer, however we only have a single load balancer. We could add an additional load balancer in front of our jumpbox with a static ip, however this would increase costs.

3) Ubuntu configuration: We have a relatively default configuration of ubuntu on our systems for 18.04. This allows logged in users to use sudo, which can increase the risk of escalation of privilege at the Jumpbox level. There are several mitigating factors we can use in this one, but here are the most crucial for hardening.
    a) Harden Sudoer file (provide examples, report write up will show more technical details)
    b) Install and configure SELinux (I don't know how to configure this, but installing it should be enough of a good step. Explain why)
    c) Harden access to crucial files/processes (passwd/shadow/group/cron)

4) Add a Firewall with Threat Intelligence (Azure Feature) to track inbound threats, but also monitor any outbound threats.

5) Don't allow for Root User login.

6) Add a root user password that is only given to team leads via a Password Manager. Rotate this every 90 days.

7) We need to keep port 80 open (and we should also open 443), but we can whitelist a set of IP addresses and provide VPNs to the employees to ensure they are the only ones able to access the training apps.

4) Was our work enough (did we do good enough)?

Executive Summary

XCorp should do more to protect and secure their assessments. XCorp infrastructure uses Microsoft Azure as a platform to deploy their infrastructure and provide web services.

XCorp needs to understand that cloud uses Shared Security model; By that, it means Cloud Service Providers protect the datacenter but RedTeam is responsible for safeguarding its own data.

XCorp needs to understand that Cloud Service Providers' responsibility is concerned with keeping up with vulnerabilities and data exploits behind the interfaces and services that are exposed and consumed by their customers.

However, it is XCorp’s responsibility to understand cloud architecture and is responsible for securing XCorp cloud infrastructure and securing the data.
It is XCorp’s responsibility to protect and secure the environment in order to fence off security threats that imperil customer data in XCorp environment.

The goals are to do enough to strengthen XCorp infrastructure:
For this, CloudTeam recommends the followings:

Rating: High
Description: XCorp does not have a Cloud environment management function.
Impact: XCorp does not have a group of personnel or personnel who is tasked with responsibility for strategic planning, architecting infrastructure, and maintaining XCorp’s assets.
Remediation: Hire or delicate an Architect who will be leading the effort and be responsible for

- Architecting the XCorp cloud environment.
- Define, oversee and approve cloud setup prior to deployment.
- Layout plan for securing data, and assets in the cloud.
- Implement best practices as well as technology to monitor and safeguard data in the cloud.
- Conduct regular cyber risk assessments with help from cybersecurity specialists if needed.
- Monitor XCorp public facing interfaces and Network's inbound and outbound using Monitoring function tools (Network Security Monitoring tools), and automate monitoring software tools.

Rating: High
Description: Immediate stop using all default login credentials.
Impact: The web login credentials for customer facing is using default login credentials (admin/admin). It has been known to take only a second of time for Cybercriminals to again access that interface and start the exploits beyond that.
Remediation: Immediate conduct an assessment on all login credentials that are created and used in the environment and change all default web login credentials. Moreover, make it a policy that no default login credentials for services/interfaces are allowed. Automate setup script to detect such violations.

Rating: Medium
Description: Increase Robustness and recovery ability of XCorp’s cloud infrastructure.
Impact: Current environment only includes one Load Balancer.
There are no backup nodes/servers for the Load Balancer node. In the event any of these servers malfunction, there will be no recovery and XCopr’s cloud environment would be totally inoperable. On top of that, XCorp’s assets could be totally up to Cyber Criminals merciness.
Remediation: Create some backup nodes/servers that are ready to failover to ensure the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation.

Rating: Medium

Description:	Foster a cloud risk and awareness work environment
Impact:	Insufficient cyberthreat knowledge and insecure Interfaces and APIs as well as poorly APIs call create vulnerabilities and allow exploits. Cybercriminals can be posing as legitimate users, and operators or developers to exploit vulnerabilities.
Remediation: Conduct code review and design review for application/deployment that use Cloud provided APIs and services. Conduct cybersecurity Training for organization to rise/Change employee awareness and behavior. Foster a workplace environment that enables employees to acquire the skills needed to keep cyber-threats at bay.