Interview Questions

Cloud Security

Cloud Access Control

How would you control access to a cloud network?

Access to any network must be controlled to prevent unwanted compromise. While intrusion or exfiltration of data are examples of deliberate malicious action taken against a network, accidents caused by a legitimate user who has too much access to network resources can similarly disrupt network services and delivery of goods.

Provide a Concrete Example Scenario

The first Cyber Security Bootcamp project the University of Oregon required had us create and deploy a completed cloud-based network developed in Microsoft’s Azure Lab Services - Cloud Service environment. When completed the full network consisted of 6 Azure Services Virtual Machines (VM) running the Ubuntu 18.04 version of the Linux - Debian-build as their OS. The primary VM was running an Ansible container configured as a gateway jumpbox. The secondary VM managed a Docker container configured as an ELK server. The final four Virtual Machines in this deployment manage the Docker DVWA containers configured as webservers. The webservers and ELKStack containers are configured and deployed from the jumpbox container by running a YAML scripted Ansible Playbook I configured for this deployment.

To provide security to the network I employed a ground-up security design building a secure environment from its initial design. The gateway jumpbox was the first and most important piece of my secure from the ground up implementation. Beginning with controlling access to the frontend by restricting ports and services and limiting access into the Jumpbox to ssh via port 22. Additionally, bypassing the vulnerable user login and instead, employing a sha-256 public-private ssh-key login protocol. The ssh-key login is further restricted to the sysadmin IP only. The key pair used for this stage of login is generated by Sysadmin machine where the private key resides and is sent from sysadmin IP to jumpbox for system/network login. IPs can be spoofed easily enough, but the ssh keypair is far more robust. It would be very difficult to spoof the IP AND provide the correct private ssh-key login credentials. Together they provide the first layers of the security onion to prevent unwanted network intrusion.

Network Security requires constant monitoring and vigilance. To this end I deployed a VM with dual CPUs, 8 GB of RAM and the ELK Docker container, ELKStash, as the network IDS/IPS defense system. It integrates data collection of system logs with various analytical tools including Elasticsearch, Logstash and the Kabana (ELK) web interface to display the data for analysis. ELKStack operates on port 5601, all log and metric data from the webservers, data to Kabana, and login from the Jumpbox occur on port 5601. To harden port 5601, and protect the ELK IDS/IPS system, NSG inbound rules restrict all inbound traffic to have originated within the V-Net. Only the jumpbox, webservers and ELKStack can access port:5601. ELKStack port:9200 is open to HTTP traffic from sysadmin IP for monitoring purposes.

Securing the Webservers began with Jumpbox restricting network access and ELKStack monitoring network security and continues with restricting the network access options potentially available for infiltration. On the backend, as all 4 Webservers are headless, without public IPs, only Jumpbox has the correct private ssh-key required to control them through the Virtual Network via ssh on port 22. ELKStack can receive system logs and metrics sent from webservers via port 5601 but can’t directly request any. The webservers have ELK modules known as “Beats” modules. Included in the deployment, they send data to ELKStack. Filebeat obtains and forwards log data to ELK and Metricbeat similarly obtains and forwards system metrics to ELK. Metrics such as CPU and memory usage to name a couple of the many available for monitoring. HTTP traffic from Load Balancer’s IP is routed through port 80 to its backpool, the webservers. All other ports and IPs are blocked from access to the webservers.

The Jumpbox / Virtual Network also share a ssh-key pair. Jumpbox maintains the private key and provides the public key to the 5 other VMs. This allows sysadmin can login to all of them, so they can all talk to Jumpbox, but not each other. The system administrator IP can log into the jumpbox using the SSH-keypair. Once in Jumpbox, sysadmin can provide the ssh-key from the Ansible container to any of the five servers in the network. In this manner, sysadmin can access all the systems on the network but must go through Jumpbox to do it. This “bottleneck” provides yet another layer of the security onion by blocking all intrusive traffic.

Securing the Webservers on the frontend from internet intrusion begins with a load balancer to distribute network HTTP traffic to the four DVWA webservers. With all web traffic going through the load balancer, it filters the incoming HTTP by being the public IP of the webservers. The webservers are restricted through NSG rule to only permit HTTP traffic on port 80 from the Load Balancer’s IP, adding more layers to the security onion enveloping this network.

The entire Virtual Network is 2 VNets. Both Jumpbox and ELKStack are on the West US regional network while the 4 webservers and load balancer are in the East US region. A peering between the two networks permits smooth reliable data transfer across and between both networks. To further harden the webservers against disaster, natural or otherwise, the 4 webservers are divided into 3 availability groups. Availability groups refer to the actual physical servers supporting the cloud environment. Dividing one availability group into 3 different groups, can maintain some webserver presence even if a fire or flood physically destroy one or even two server farms.

When completed, this webserver network was hardened against deliberate or accidental intrusion, and natural or man-made disaster. It has built-in redundancy, IDS/IPS monitoring and protection, and a load balancer to filter and distribute incoming HTTP traffic. Users are restricted to sysadmin only or internet via HTTP on the frontend to minimize human error or skullduggery as a source of network failure. Additionally due to ansible playbook protocol, this network can be scaled up, updated, upgraded or completely torn down and rebuilt in a few minutes by simply executing a script.