Lab Report - Lab 14

Name: Tyler Burleson Date: 4/22/2024

Summary

The purpose of this lab was for us to launch and configure a second domain controller and member server. We were then tasked with having them join the Flux domain and connect to their corresponding subnets. Once each server was connected to their corresponding public or private subnet, we replicated a crash with the original domain controller and verified that the second domain controller was able to handle the DNS requests without the first domain controller.

We started by accessing our AWS EC2 dashboard and spinning up a new Windows Server 2022 to be our new domain controller. Once the instance was running and we had changed the machine name to match our AWS instance name we connected the network adapter to add two new DNS's so we could communicate with our first domain controller. Once this instance was able to reach the first domain controller, we added it to the domain and promoted the server to a DC after installing Active Directory.

Our next step was creating a private subnet in our VPC and launching a new member server using this subnet. This new instance wouldn't have access to the internet, but we would still be able to connect to it through our domain controller to configure it to be accessible through the network. Once we started our instance, we remoted in using our first domain controller and configured the member server's name and network adapter the same way we did the second domain controller. Once the server had restarted, we attempted to ping the first and second member servers by using their domain names. Once we realized we couldn't reach the machines through their domain we remoted into each one to tweak some settings.

Once we were inside each member server, we navigated to their Windows Defender Firewall and Advanced Security settings. We then located the inbound rules and typed "f" to find the "File and Printer

Sharing (Echo Request – Ipv4-In)" rule and enabled it. Once we had changed this rule, we navigated back to the domain controller and pinged the member servers using their domain names and we were able to reach them now.

After confirming our connection to the member server domains, we simulated a crash on DC01 by stopping the instance. We then RDP into our DC02 and pinged our member servers. We were still able to reach them, then we pinged our DC01, which returned request timeouts, but we were still able to see the IP address. We then checked our Active Directory Users and Computer and viewed our member servers and our domain controllers. Once this was complete, we knew our DC02 would be able to handle the DNS request in case DC01 crashed and our lab was complete.

Screen Shots

```
PS C:\Users\Administrator> ping fluxwindc01.flux.loc

Pinging fluxwindc01.flux.loc [172.30.1.10] with 32 bytes of data:
Reply from 172.30.1.10: bytes=32 time<1ms TTL=128
Reply from 172.30.1.10: bytes=32 time<1ms TTL=128
Reply from 172.30.1.10: bytes=32 time<1ms TTL=128
Reply from 172.30.1.10: bytes=32 time<1ms TTL=128

Ping statistics for 172.30.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\Users\Administrator>
```

Figure 1

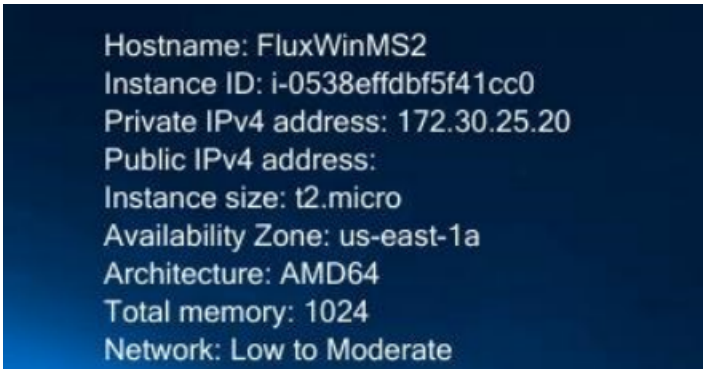

Figure 2

```
PS C:\Users\Administrator> ping google.com

Pinging google.com [172.253.63.113] with 32 bytes of data:
Request timed out.
Request timed out.

Ping statistics for 172.253.63.113:
    Packets: Sent = 2, Received = 0, Lost = 2 (100% loss),
Control-C
PS C:\Users\Administrator> ping fluxwindc01.flux.loc

Pinging fluxwindc01.flux.loc [172.30.1.10] with 32 bytes of data:
Reply from 172.30.1.10: bytes=32 time=1ms TTL=128
Reply from 172.30.1.10: bytes=32 time=1ms TTL=128
Reply from 172.30.1.10: bytes=32 time=1ms TTL=128
Reply from 172.30.1.10: bytes=32 time=1ms TTL=128

Ping statistics for 172.30.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

Figure 3

```
PS C:\Users\Administrator> ping fluxwinms01.flux.loc

Pinging fluxwinms01.flux.loc [172.30.1.20] with 32 bytes of data:
Reply from 172.30.1.20: bytes=32 time<1ms TTL=128
Reply from 172.30.1.20: bytes=32 time<1ms TTL=128
Reply from 172.30.1.20: bytes=32 time<1ms TTL=128
Reply from 172.30.1.20: bytes=32 time<1ms TTL=128

Ping statistics for 172.30.1.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\Users\Administrator> ping fluxwinms2.flux.loc

Pinging fluxwinms2.flux.loc [172.30.25.20] with 32 bytes of data:
Reply from 172.30.25.20: bytes=32 time=1ms TTL=128
Reply from 172.30.25.20: bytes=32 time=1ms TTL=128
Reply from 172.30.25.20: bytes=32 time=1ms TTL=128
Reply from 172.30.25.20: bytes=32 time=1ms TTL=128

Ping statistics for 172.30.25.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

Figure 4

Research Questions

1. How could we enable the member server on the private subnet to access the Internet? (This is something I haven't figured out how to implement in a Windows environment, yet, but it is possible)

You could enable rules in the subnet/VPC to allow outbound connections to the internet.

2. You may have noticed that, in this lab, we configured the member servers to hit both DC01 and DC02 (if DC01 isn't available). Is there a way, or a technology, we could use to configure our member servers to hit a single IP, but still maintain redundancy in terms of two, three, or even 50 domain controllers? [hint: the short answer is 'yes.' And that technology is something we've talked about this semester]. Consider...an appliance that sits between the Internet and multiple servers that shows a single IP address to the Internet and routes all incoming traffic internally. What could that be?

Yes, by using a load balancer we can point all the member servers at a single IP address and allow the load balancer to direct the requests to a suitable target.

3. Briefly define and describe Network Address Translation. What is it? What does it do?

NAT is a technique used in networking to modify network address information in the packet headers while they are in transit. This helps conserve IP address by allowing multiple devices in a private network to share a single public IP address and enhances security by acting as a barrier between the private network and the public network.

4. Why might we want to 'hide' a machine like FluxWinMS02 from the Internet?

This helps enhance the security on the FluxWinMS02 server if the server doesn't *need* internet access. By removing access to the internet, it makes the server only reachable from the inside and reduces vulnerability points.

5. Briefly define and describe Software Defined Networking. How does it apply (or is it applied) to the AWS environment?

SDN is an approach to network management that enables dynamic, programmatically controlled network configuration and operation. In the context of the AWS environment, SDN principles are applied through services like Amazon VPC and AWS Global Accelerator.

6. If we wanted to be able to spool up member servers in our new environment that will allow the ICMP protocol through, how could we make that happen without having to manually configure the firewall of each after we launch it?

You could use the SSM from AWS to run commands to activate the protocol while the server is spooling up or create a GPO in AD to enable this protocol for all computers in the group we wanted.

7. We saw in the lab that, between the AWS Security Group, FluxSG, and the Windows Servers' Windows Defender application, we, in effect, have two layers of security implemented in our environment. What is another service/application that we could enable to further harden our cloud environment (I can think of two, off-hand)?

We could use the AWS WAF or Web Application Firewall to help harden security on web applications. Additionally, we can use the AWS Inspector to run automated security assessments to help improve security compliance.

8. We saw the inconvenience associated with having a server that can't access or be accessed via the Internet, but why might placing servers out of reach of the Internet be desirable?

This helps enhance the security on the server if the server doesn't *need* internet access. By removing access to the internet, it makes the server only reachable from the inside and reduces vulnerability points.