Nicholas Suchy – CMPT 416N Section 111 – Homelab

Introduction:

This document is to serve as a walkthrough guide to recreate the set up of a Homelab. This documentation will cover the setup of the VMware hypervisor, which will be used to virtualize an attack and defense box along with various security services, and ultimately illustrate an attack and implement a security measure to patch the vulnerability exploited in the attack.

Set up of the Hypervisor (VMware):

A screenshot of a computer

Description automatically generatedFor Objective 1 of the Homelab, we were to set up our virtualization technology. For my hypervisor, I used VMware Workstation Pro 17. All configurations of VMs and VMnets (used for networking) will be done in this environment.

Creation of the Attack Box (Kali Linux):

The attack box being used in this Homelab is Kali Linux. Proof of it running in VMware is shown below.

A screenshot of a computer

Description automatically generated

Creation of the Defense Box (Metasploitable2):

A screenshot of a computer

Description automatically generated The defense box to be exploited in Metasploitable2. Proof of it running in VMware is shown below.

Implementation of First Security Service (Nessus Vulnerability Scanner):

For my first security service, I implemented Nessus, specifically Nessus Essentials. Nessus is a vulnerability scanner that can act as a double-edged sword for security purposes. On one hand, Nessus can provide crucial information about potential vulnerabilities with an organization’s network. However, when used by an attacker, Nessus can give helpful advice on weakness to exploit and potential locations within an organization to start an attack.

Nessus was downloaded from the [Nessus](https://www.tenable.com/downloads/nessus?loginAttempted=true) website and was installed with the help of Chioma Ibeakanma’s article on [MakeUseOf.com](https://www.makeuseof.com/how-to-install-nessus-kali-linux/). In order to access the GUI for Nessus Essentials, the command in the screenshot below needs to be run to start to Nessus service.



The screenshot below demonstrates proof of Nessus running within the Homelab on Kali Linux, by providing a screenshot of the GUI, accessible via http://localhost:8834, showing completed scan that will be described later.

A screenshot of a computer

Description automatically generated

Implementation of Second Security Service (pfSense):

For my second security device, I implemented pfSense, an open-source firewall and router. In security, firewalls follow a set of rules which either allow or deny traffic through the firewall. Additionally, pfSense serves as a router, allowing different networks to communicate with one another.

The version of pfSense being run in the Homelab is 2.7.2 and was downloaded from the [pfSense](https://www.pfsense.org/download/) website. Installation and configuration was primarily guided by the walkthrough on [cyberwoxacademy](https://cyberwoxacademy.com/building-a-cybersecurity-homelab-for-detection-monitoring/).

A screenshot of a computer

Description automatically generated

Networking for the Homelab:

A diagram of a network

Description automatically generatedThe screenshot below shows a network topology for the Homelab.

One trouble I had with successfully setting up the network involved pfSense, which I was able to fix with help from an online [forum post](https://forum.netgate.com/topic/123611/communication-between-machines-on-2-different-subnets/4). As a default, pfSense uses 2 interfaces: one for a WAN connection and the other for a LAN connection. In the case of the Homelab, 2 LANs were needed. When creating a new LAN, pfSense doesn’t create any default rules for the interface the new LAN is sitting on and instead blocks all traffic that is going through said interface. By configuring a basic allow any rule on this A black screen with white text

Description automatically generatedA black background with white numbers

Description automatically generatedinterface, full connectivity was achieved, as demonstrated by the screenshots below.

Illustration of an Attack:

Scanning the Network and Target using Nmap:

A screenshot of a computer program

Description automatically generatedA screen shot of a computer

Description automatically generated Like every attack, the first step is to gather information about a potential target. For the sake of focusing on illustrating the technical components of an attack, it will be assumed that the network the vulnerable VM is on was found through passive reconnaissance. In order to find any interesting IP addresses on the network, we’ll use a Nmap ping sweep to discover active hosts on the network, as demonstrated in the screenshot below

A screenshot of a computer

Description automatically generatedThe results from the ping sweep return 2 IP addresses, including the IP of our machine, that are currently active on the network. As an attacker, we want to gather more information about these hosts and look for potential vulnerabilities. We can perform OS scans on IP addresses to try and gather potential information (shown in the surrounding screenshots).

With the information above, we can see the 192.168.2.101 host is likely to be more vulnerable than the 192.168.2.1 host for several reasons. Most notably, the .101 host has a significantly higher number of open ports compared to the .1 host. Additionally, the OS scan revealed information about potential operating systems and the .101 host is shown to have an outdated version of Linux, meaning there are likely vulnerability to exploit. Given this, as an attacker we would likely target the .101 system. To get some further insight as to what’s going on with our newfound target, we can also use Nmap to aggressively scan our target’s ports, providing OS information, service information, and an overall better view as to what is occurring on each open port.

A computer screen shot of white text

Description automatically generatedA screenshot of a computer

Description automatically generatedA computer screen shot of white text

Description automatically generated

Using Nessus to conduct a vulnerability scan:

A screenshot of a computer

Description automatically generatedFurther, to get an even better understanding of our target we can conduct a vulnerability scan on the target. As described above, Nessus can scan our target for vulnerabilities and report them back to us. As an attacker, we are taking the results of the vulnerability scan and using them to try and exploit, rather than patch these vulnerabilities. Additionally, after a scan is completed, Nessus lists the vulnerabilities in order from more critical and dangerous, to simply helpful information about the system. The screenshot below shows what Nessus found to be the most critical vulnerabilities. As an attacker, we would likely want to exploit these as they will have the highest potential to damage and inflict trouble on our target.

A screenshot of a computer

Description automatically generatedHighlighted in the above screenshot are multiple issues with SSL that can grant us a shell remotely. By clicking on it and further investigating we can see that there exists a bug in the random number generator making it easy for an attacker to get the private part of a key, allowing the attacker to establish a man-in-the-middle attack. Nessus also tells us what port and service that this bug is directly affecting, allowing us to further narrow our potential launching point for an attack. With Postgres being a database service, it could be a treasure trove to attackers, containing a wide variety of sensitive information such as names, credit card numbers, and even protected health information, depending on the organization this network belongs to. Therefore, for the next steps, I’ll be targeting port 5432 and the Postgres service running there.

Finding valid Postgres credentials

A screen shot of a computer

Description automatically generated Now that we have a target, we can use the Metasploitable framework to try and extract information from the database we’ve found. I found the Metasploit Framework’s [documentation](https://docs.metasploit.com/docs/pentesting/metasploit-guide-postgresql.html) very helpful throughout this section. We can access the framework using the “msfconsole” command shown below.

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generatedOnce in the console, we can then search for modules that pertain to Postgres by issuing the command “search postgres”.

Several modules are revealed with a wide range of uses from capturing authentication credentials to executing a payload on the host remotely. Assuming the database is password protected, we probably want to search for credentials first. To do this, I used the postgres\_login module. A screenshot of options to configure within the module is to the right.

From the description of the module, we can gather that this module attempts to authenticate into the Postgres service from given username and password files, or by directly inputting potential credentials. For this case, we’ll attempt to authenticate using Metasploit’s default credential lists for Postgres. To execute the attack, we need to input the host that we are targeting.

A screenshot of a computer screen

Description automatically generatedNow that everything is set up, we can execute the brute force attack to check for usage of default credentials by running the command “run”. The screenshot below shows the output of this module. In the output, we can see the there was a successful login attempt using “postgres” for both the username and password. Now that we have found valid credentials, we can take our next step in conducting an attack.

Exploiting the Defense Box

A screen shot of a computer

Description automatically generatedGoing back to the previous screenshot which demonstrates the various modules that can be used to exploit Postgres, there is a module called postgres\_payload which we can use to gain access to the service remotely. As we’ve discovered before with our OS scans of the target, the host is running Linux, meaning we must choose the postgres\_payload module that was created for Linux execution. By using the command illustrated below, we can gain access to this module.

A computer screen shot of white text

Description automatically generatedA screenshot of a computer program

Description automatically generatedOnce in the module we can configure the options to suit our needs for the attack. We can configure RHOSTS to be our target, LHOSTS to be the Kali box so that we can remotely control Metasploitable, and finally configure the credentials the module will use in the exploit to the ones we found in our brute force attack. Then we can leave the rest of the options as their defaults and execute the attack. The first screenshot below shows a configuration of the options used with the second screenshot showing the attack being successfully executed.

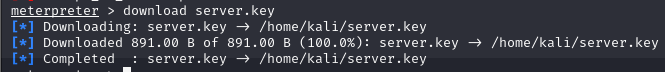
A screenshot of a computer

Description automatically generatedWhile this demonstrates a successful attack, I wanted to go a step further and exfiltrate a file from the defense box. With remote access to the service, there are a various number of possibilities that we can do. First, we can gather some context about where we are by using the command “ls”.

A screen shot of a computer

Description automatically generatedSeveral interesting files and directories appear, with most notably being “server.key”. By opening that file, we can see that the file contains cryptographic key information! The leakage of cryptographic key information is detrimental to organizations as anything that is encrypted with an expose encryption key must be considered exposed as well. This could result in compliance issues resulting in substantial legal fees, fines, and lawsuits if sensitive information is exposed. Attackers can download this file to their devices and use it in the future to decrypt communications and any other files can get encrypted with this key.

A screenshot of a computer

Description automatically generatedTo finish my attack, I downloaded the server.key file back to Kali. This illustrates an attack that was successful in exfiltrating sensitive data. The screenshots below show the commands used to download the file and show proof of the file in Kali Linux.

Patching the Vulnerability:

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated From a defensive perspective, it is apparent that the exploitation was able to occur due to a variety of factors. The usage of default passwords and unrestricted access to ports that should be closed are two of the key components that allowed the attack in the Homelab to be successful. Therefore, to patch the vulnerability we need to fix these issues. Firstly, I used pfSense to help limit the number of accessible ports. Through pfSense I gave machines outside the defense box’s network access to only 3 ports, 22 for SSH, 80 for HTTP, and 443 for HTTPS. Additionally, I allowed the defense box to be pinged, for ease of future modifications to networking within the Homelab. Otherwise, all other traffic would be blocked. These are highlighted by the screenshot of the new firewall rules below. Additional screenshots provided shows that Kali and Metasploitable can still ping each other.

A screenshot of a computer screen

Description automatically generated To test the effectiveness of this defense, the Metasploit framework was brought up and the same attack described above, using the same modules, was conducted again. This time however, as illustrated below, no session was opened as all previously open ports have been blocked by the firewall.

While this patch does protect against the attack, it also limits what services can be reached remotely by other machines, which could potentially have a negative impact on business operations. To account for this, I tried to leave open ports that are commonly accessible within organizations and their devices. And although there are potential negatives, this defense also serves to protect against numerous other attacks. Certain services, such as the database service exploited in the lab, should usually not be available to machines on the internet. This defense closes off those ports and helps protect the variety of services running on the defense box from being exploited as they can be accessed remotely by users on the internet.

A black screen with white text

Description automatically generatedYet, there is still the underlying vulnerability of using a default password for the database service. To strengthen the patch, I went and changed the Postgresql service’s password from the default password, using the command highlighted in the screenshot below.

A screen shot of a computer

Description automatically generatedTo test to check if the vulnerability was patched, I went back to the module within the Metasploit framework, which was previously referenced in the lab to obtain the default credentials of the Postgres service. The screenshot below demonstrates that the module, now operating against a new, non-default password, can’t crack the credentials. Without the credentials, the previous exploit should not be able to occur anymore within the environment. This is highlighted by the second screenshot, with the exploit module returning an error stating “authentication failed”. With this, the attack used within the Homelab has been patched.

A screen shot of a computer

Description automatically generated

Conclusion

Overall, this lab serves as a reminder to use strong, complex passwords and always change passwords from their defaults and close ports not in use or that are running sensitive services. By simply changing passwords and other credentials from their default values, organizations can block many potential threats and deter attackers looking for an easy target. And by restricting port access, organizations can harden their systems and help prevent a wider variety of attacks.