Lab 4 - Vulnerability Scanning

Objective

The objective of this lab is to introduce you to vulnerability scanning. You will be scanning a Docker container running Ubuntu 14.04 using OpenVAS to identify security vulnerabilities in that container. Equipped with the knowledge of vulnerabilities on the container, you will implement protections to mitigate most of the identified vulnerabilities.

Background

OpenVAS is an open-source vulnerability assessment system for remotely detecting publicly disclosed vulnerabilities in Operating Systems and Applications. Learn how OpenVAS works by reading their software description page. Also read about `NVT` and `SCAP` to understand how they are useful in detecting vulnerabilities. A vulnerability scan can reveal many potential security vulnerabilities in a remote system. A cybersecurity professional can use these reports to identify and remediate vulnerabilities before an attacker can exploit one. However, attackers also can use these reports to find insecure services running on a system and then build or use publicly available exploits to compromise a system. As such, it is important for professionals to run these scans regularly and remediate vulnerabilities before an attacker.

Setup

Login to the Linux VM that you used in the previous labs.

1. You will be scanning a Docker container created specifically for this lab. This container is called `comp3611/scanning` and is installed on your Linux VM. Start this container by typing the following command in a terminal on your Linux VM.

   ```
   docker run --detach --name lab4 comp3611/scanning
   ```

2. `OpenVAS-9` is already installed on your Linux VM. The scanner requires updated `NVT`, `SCAP` and `CERT` feeds to find the latest security vulnerabilities. Follow the instructions on this link to update the feeds and start the required services. *Ensure that you follow the instructions for OpenVAS version 9, and that there are no errors.*

3. The `docker run` command will start the Docker container at the IP address `172.17.0.2`. Before proceeding further, run `nmap` on the container using the following command. You should see four open ports on that IP address running services with versions that are quite old. As such, a vulnerability scan may report several high and medium severity vulnerabilities for this container.

   ```
   nmap -sV 172.17.0.2
   ```

Part 1 - Initial Scan

1. OpenVAS uses a web interface called Greenbone Security Assistant (`GSA`) to schedule and manage vulnerability scans, and to generate comprehensive reports for every scan. Using a web browser on your Linux VM, visit `https://localhost:4000` to access the `GSA` interface. *This interface uses a self-signed certificate that is untrusted by the browser. As such, accept the risk and continue when prompted that the website is not safe to access.* The default username and password for `GSA` are both `admin`.

2. From `GSA`, configure a "Full and fast" scan for the `lab4` Docker container using its IP address. *To reduce memory consumption from scanning, set the number of concurrent NVTs to 1.* Run the scan and download the completed scan report as a HTML file to submit with your lab report. *This report should show about 9+ high and 35+ medium severity vulnerabilities. If you see lower number of vulnerabilities than this, clear any filters that may have been applied.*

Part 2 - Implementing Protections

Let's assume that the organization has the following **requirements** for this container -

- A SSH server for remote system administration
- An Apache2 web server to serve web content to users
- All other services can be disabled

Analysis

1. Analyze the scan report and look at all identified high and medium severity vulnerabilities.
   - You will observe that most vulnerabilities are associated with `Apache2`, `OpenSSH` and `Samba`. This is because their versions are outdated and many vulnerabilities were publicly disclosed for those versions. Let's not worry about `Samba` going forward, as that is not part of the requirements.
   - Read the remediation guidelines for `Apache2` and `OpenSSH` vulnerabilities in the report. You will observe that most vulnerabilities can be mitigated by upgrading the services to recent versions. So, let's focus on upgrading the services to more recent versions.
2. There are several approaches for upgrading the services to more recent versions.
   - **Upgrading packages from source** - In this approach, you would have to download the latest source code for a service, backup all existing files, configure the Makefile and then install the service. This approach will ensure that the latest service is installed, but it can be error-prone.
   - **Upgrading package repositories** - In this approach, you could change the `apt` package sources to point to a newer version of Ubuntu. Running `apt update && apt upgrade` would upgrade all the packages to newer versions. However, there is a high probability that certain packages don't work as intended and thorough testing will be required.
   - **Re-creating the Docker container** - In this approach, you would update the exiting `Dockerfile` to build a Docker container with newer package versions. This is the easiest approach for this lab, and one that you'll implement in the remediation step.

Remediation

1. The `Dockerfile` used to create the `comp3611/scanning` image is attached with this lab on Blackboard. Read this file carefully and make sure you understand each line of this file. Now, update this `Dockerfile` to (1) adhere to the organization's requirements, and (2) use updated versions of the `Apache2` and `OpenSSH` packages. *Note that old versions of Ubuntu may not support newer package versions, so select the Ubuntu version accordingly.*

2. Once you are satisfied with your updates, build the Docker image (from the same folder as the `Dockerfile`) using the following command. *Ensure that there are no errors in the build process.* If there are any errors, fix those errors, and re-build the image.

   ```
   docker build -t comp3611/scanning_sec .
   ```

3. Stop the insecure old container, and start the secure updated container using the following commands. The `docker run` command will start the updated Docker container at the IP address `172.17.0.2`.

   ```
   docker stop lab4
   docker run --detach --name lab4_sec comp3611/scanning_sec
   ```

Part 3 - Validation Scan

1. From `GSA`, configure another "Full and fast" scan for the updated `lab4_sec` Docker container using its IP address. Run the scan and download the completed scan report as a HTML file to submit with your lab report. *This time, your scan should identify much less high and medium severity vulnerabilities. Most vulnerabilities reported in Part 1 should be no longer reported.*

2. Verify that the container is running correctly. To test this, first try to SSH into the container as user `student` using the same password as your Linux VM. Take a screenshot showing a successful login.

Next, verify that the web server is running by opening `http://172.17.0.2` in a browser. The page should load up an image that matches Figure 1. Take a screenshot of the browser window displaying the image.

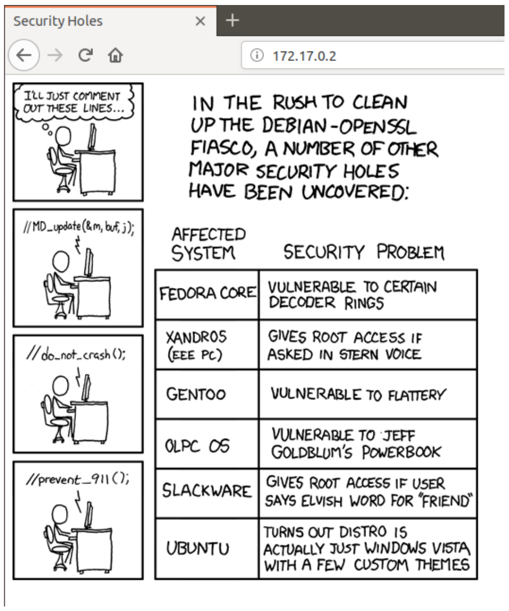
Security Holes
172.17.0.2
I'LL JUST COMMENT OUT THESE LINES...
//MD_update(&m, buf, j);
//do_not_crash();
//prevent_911();
IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:
AFFECTED SYSTEM
SECURITY PROBLEM
FEDORA CORE
VULNERABLE TO CERTAIN DECODER RINGS
XANDROS (EEE PC)
GIVES ROOT ACCESS IF ASKED IN STERN VOICE
GENTOO
VULNERABLE TO FLATTERY
OLPC OS
VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK
SLACKWARE
GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND"
UBUNTU
TURNS OUT DISTRO IS ACTUALLY JUST WINDOWS VISTA WITH A FEW CUSTOM THEMES

Figure 1: The webpage that should be displayed on accessing http://172.17.0.2

Cleanup

Stop and remove the containers by typing the following commands in a terminal on your Linux VM.

```
docker rm lab4
docker stop lab4_sec && docker rm lab4_sec
```

Lab Report

For this lab, each student must submit a report with the following information:

1. Submit all the required screenshots.
2. Submit the "Full and fast" scan reports before and after remediation.
3. Submit the `Dockerfile` for the secure updated container.

Grading

- 50 points – Successfully scanned the Docker containers and generated the correct reports
- 50 points – Successfully remediated many high and medium severity vulnerabilities

Optional Extra Credit - Create your own network scanner

Grading - 2.5% added to the final grade

In this part, you MUST implement your own network scanner that is similar in functionality to `nmap`. Your scanner must implement the following flags -

- `-T` for TCP connect scan
- `-U` for UDP scan
- `-S` for TCP SYN scan

The above flags must work using the same techniques as the corresponding `nmap` flags. Make sure that you read the `nmap` manual and understand how these flags work. For simplicity, your network scanner should accept as input just one flag and one IP address for a single run. It should be programmed to scan ports 1 - 1024. The scanner must run on your Linux VM, and you MUST also provide a `setup.sh` shell script for installing your code (inclusive of installing the dependencies). It must execute as follows:

```
# ./net_scan <flag> <IP>
    Scanning <IP> using the <flag> flag...

    PORT                STATE
    1                   <open/closed/filtered>
    2                   <open/closed/filtered>
    3                   <open/closed/filtered>
    ---snip---
    1022                <open/closed/filtered>
    1023                <open/closed/filtered>
    1024                <open/closed/filtered>
```