Vulnerability Scanning and Management

EXERCISE 1 - Installation

Prerequisites

- Linux Ubuntu 22.04 Server
- nmap installed
- git installed

Git

Git is a distributed version control system that is widely used for tracking changes in source code during software development. It's designed to handle everything from small to very large projects with speed and efficiency. Git enables multiple developers to work together on a project, allowing them to track and merge changes from various branches of development. Its main goals include data integrity, speed, and support for distributed, non-linear workflows.

GitHub, GitLab, and similar platforms are web-based services that provide Git repository hosting, along with additional features designed to enhance collaboration, project management, and code quality in software development projects. While all these platforms leverage Git for version control, they each offer unique features and integrations that cater to different aspects of the software development lifecycle.

GitHub

- **Overview**: GitHub is one of the most popular and widely used cloud-based hosting services for Git repositories. It provides a platform for software developers to store, manage, track, and collaborate on software projects.
- **Key Features**:
  - **Pull Requests** and **Code Review**: Facilitates code review and comments on changes before merging them into the main project.
  - **Actions**: Offers CI/CD (Continuous Integration/Continuous Deployment) capabilities, allowing automated testing and deployment.
  - **Project Management Tools**: Includes issues, projects (Kanban boards), and milestones for task tracking and management.

- **Community and Collaboration**: A vast community of developers, making it easy to contribute to open-source projects or use others' projects.
- **Marketplace**: Provides access to many integrations and tools to enhance development workflows.

GitLab

- **Overview**: GitLab is a web-based DevOps lifecycle tool that provides a Git repository manager providing wiki, issue-tracking, and CI/CD pipeline features, using an open-source license.
- **Key Features**:
  - **Integrated CI/CD**: Offers built-in continuous integration and deployment, making it straightforward to automate the testing and deployment of code.
  - **Comprehensive DevOps Lifecycle Support**: Designed to support the entire DevOps lifecycle in a single application, from project planning and source code management to monitoring and security.
  - **Auto DevOps**: Automatically configures CI/CD pipelines based on best practices with minimal configuration required.
  - **Issue Tracking and Project Management**: Similar to GitHub, it offers issue tracking, boards, and milestones, but with a more integrated approach to managing the DevOps lifecycle.
  - **Self-hosting**: In addition to the cloud service, GitLab offers self-hosted options, giving teams control over their environment and data.

Bitbucket

- **Overview**: Bitbucket, owned by Atlassian, is another Git repository management solution that offers source code and development project hosting.
- **Key Features**:
  - **Integrated CI/CD** with Bitbucket Pipelines: Allows for automated builds, tests, and deployments.
  - **Jira Integration**: Offers deep integration with Jira, another Atlassian product, for issue and project tracking, linking code changes to Jira issues directly.
  - **Code Review**: Pull requests and inline commenting facilitate code reviews and collaboration.
  - **Private Repositories**: Focuses on providing private repositories free for small teams, making it a popular choice for startups and private projects.

Other Platforms

- **Azure DevOps (formerly VSTS)**: Provides Git repositories, CI/CD pipelines, and a set of tools for planning and tracking work, testing, and deploying applications. Integrated with other Microsoft Azure services.

- **SourceForge**: One of the earliest platforms to offer open-source project hosting, SourceForge supports Git, Mercurial, and SVN. It's best known for providing comprehensive tools and services for open-source projects.

What Git is Used For:

1. **Version Control**: Git keeps track of every change made to the files in a project, allowing developers to revert back to previous states or explore the history of how the code has evolved.
2. **Collaboration**: It facilitates collaboration by allowing multiple developers to work on different features or sections of a project simultaneously without interfering with each other's work.
3. **Branching and Merging**: Git's branching capabilities allow developers to create isolated environments (branches) for developing new features, fixing bugs, or experimenting. These changes can then be merged back into the main branch of the project.
4. **Backup and Restore**: Git repositories can serve as a backup of your code. You can restore your project to any previous state.
5. **Code Review and Management**: Git supports pull requests (in platforms like GitHub, GitLab, Bitbucket), which are a way to submit code changes for review before they are merged into the main project. This promotes better code quality and team collaboration.

Deploying Software via Git

In Linux, using Git can streamline the process of deploying software.

To check if Git is installed on an Ubuntu server, you can use the terminal or command line interface. Here are the steps you can follow:

1. **Open the Terminal**: Access your Ubuntu server's terminal. If you're accessing the server remotely, you can use SSH (Secure Shell) to connect to it.
2. **Check Git Version**: To see if Git is installed and to determine its version.

```
git --version
```

Expected Output (or similar):

```
git version 2.34.1
```

If Git is Not Installed: If you get a message indicating that 'git' is not recognized as a command, it means Git is not installed on your system. In such a case, you can install Git using the following command:

```
sudo apt update
```

```
sudo apt install git
```

Deploying Software via GIT

Clone the Repository: Clone the software's source code repository to your local machine or server where you want to deploy the software.

EXAMPLE:

```
git clone https://github.com/example/repo.git
```

Cloning or downloading git repositories (repos) is a simple way of downloading important software to your system.

This is just a fraction of what git can do, but the rest of it are much more for developers.

Task 1

We will be deploying the “vulnscan” – Vulnerability Scanning with Nmap Scanner to run alongside nmap on your Ubuntu Server. This is an open source tool and highly effective.

It won’t be as pretty or build nice reports like the paid vendor tools, but it gets the job done. Also, it’s GREAT to practice with, as a learner.

Step 1

Where in the world is nmap installed?

To find the executable of nmap, we’re doing to use the `find` command in the Linux CLI.

Here is a nice little tutorial by Digital Ocean (DO always has really nice tutorials)

https://www.digitalocean.com/community/tutorials/how-to-use-find-and-locate-to-search-for-files-on-linux

Using the Linux Syntax of <`command`> <`option`> <`argument`>

The option we are going to use is the `-name` option, which will allow us to search for the file name itself.

```
find -name nmap
```

Now you'll notice that you will get a BUNCH of "Permission Denied" messages, that just means you don't have the rights to search those folders. Its best to run this command starting with the sudo to escalate privileges.

```
sudo find -name nmap
```

./snap/core20/1974/usr/share/bash-completion/completions/nmap

./snap/core20/2182/usr/share/bash-completion/completions/nmap

./usr/share/lintian/overrides/nmap

./usr/share/bash-completion/completions/nmap

./usr/share/nmap

./usr/share/doc/nmap

./usr/bin/nmap

The folder out of the above, that nmap lives in is ./usr/share/nmap

Step 2

When I read the instructions to install VULNSCAN, it states: "Please install the files into the following folder of your Nmap installation: Nmap\scripts\vulscan*"

Let's navigate to to ./usr/share/nmap

```
ajay@server1:/$ sudo find -name nmap
./snap/core20/1974/usr/share/bash-completion/completions/nmap
./snap/core20/2182/usr/share/bash-completion/completions/nmap
./usr/share/lintian/overrides/nmap
./usr/share/bash-completion/completions/nmap
./usr/share/nmap
./usr/share/doc/nmap
./usr/bin/nmap
ajay@server1:/$ cd ./usr/share/nmap
ajay@server1:/usr/share/nmap$ ls -l
total 9188
-rw-r--r-- 1 root root   10556 Jan 12  2023 nmap.dtd
-rw-r--r-- 1 root root  717314 Jan 12  2023 nmap-mac-prefixes
-rw-r--r-- 1 root root 5002931 Jan 12  2023 nmap-os-db
-rw-r--r-- 1 root root   14579 Jan 12  2023 nmap-payloads
-rw-r--r-- 1 root root    6703 Jan 12  2023 nmap-protocols
-rw-r--r-- 1 root root   49647 Jan 12  2023 nmap-rpc
-rw-r--r-- 1 root root 2461461 Jan 12  2023 nmap-service-probes
-rw-r--r-- 1 root root 1000134 Jan 12  2023 nmap-services
-rw-r--r-- 1 root root   31936 Jan 12  2023 nmap.xsl
drwxr-xr-x 3 root root    4096 Feb 23 16:20 nselib
-rw-r--r-- 1 root root   48404 Jan 12  2023 nse_main.lua
drwxr-xr-x 2 root root   40960 Feb 23 16:20 scripts
ajay@server1:/usr/share/nmap$
```

Re-reading the instructions to make sure I'm PERFECTLY CLEAR on what they are asking, I find: Nmap\scripts\vulscan*"

That means I need to go into the scripts folder. (Notice that in the above image)

So immediately after entering the scripts folder, I encourage you to list the contents of that directory. Holy amount of scripts Batman!

```
-rw-r--r-- 1 root root   48404 Jan 12  2023 nse_main.lua
drwxr-xr-x 2 root root   40960 Feb 23 16:20 scripts
ajay@server1:/usr/share/nmap$ cd scripts/
ajay@server1:/usr/share/nmap/scripts$ ls -l
total 4880
-rw-r--r-- 1 root root  3901 Jan 12  2023 acarsd-info.nse
-rw-r--r-- 1 root root  8724 Jan 12  2023 address-info.nse
-rw-r--r-- 1 root root  3345 Jan 12  2023 afp-brute.nse
-rw-r--r-- 1 root root  6891 Jan 12  2023 afp-ls.nse
-rw-r--r-- 1 root root  7001 Jan 12  2023 afp-path-vuln.nse
-rw-r--r-- 1 root root  5671 Jan 12  2023 afp-serverinfo.nse
-rw-r--r-- 1 root root  2621 Jan 12  2023 afp-showmount.nse
-rw-r--r-- 1 root root  2262 Jan 12  2023 ajp-auth.nse
-rw-r--r-- 1 root root  2983 Jan 12  2023 ajp-brute.nse
-rw-r--r-- 1 root root  1329 Jan 12  2023 ajp-headers.nse
-rw-r--r-- 1 root root  2590 Jan 12  2023 ajp-methods.nse
-rw-r--r-- 1 root root  3051 Jan 12  2023 ajp-request.nse
-rw-r--r-- 1 root root  6719 Jan 12  2023 allseeingeye-info.nse
```

My screenshot simply doesn't do it justice how many scripts are in there. Keep in mind that all that is AVAILABLE for you to use if you need it.

Now that I am in the scripts folder, let's look at the next command I'll need to run.

Clone the GitHub repository like this:

git clone https://github.com/scipag/vulscan scipag_vulscan

ln -s `pwd`/scipag_vulscan /usr/share/nmap/scripts/vulscan

The clone action will create a new folder called scipag_vulnscan and then the next command will generate a symbolic link to be able to run this script from anywhere.

Go ahead and run the 2 commands above while in /usr/share/nmap/scripts

Ooops, it gave you "Permission Denied" errors. How do you think you can get around that? 😂

```
ajay@server1:/usr/share/nmap/scripts$ sudo git clone https://github.com/scipag/vulscan scipag_vulscan
Cloning into 'scipag_vulscan'...
remote: Enumerating objects: 297, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 297 (delta 12), reused 16 (delta 4), pack-reused 264
Receiving objects: 100% (297/297), 17.69 MiB | 12.67 MiB/s, done.
Resolving deltas: 100% (175/175), done.
ajay@server1:/usr/share/nmap/scripts$ sudo ln -s `pwd`/scipag_vulscan /usr/share/nmap/scripts/vulscan
```

Most of the time, it will be smooth and EZ PZ, but sometimes you might get some errors like I did. It was just internet weather, and I successfully got past it.

```
ajay@server1:/usr/share/nmap/scripts$ git clone https://github.com/scipag/vulscan scipag_vulscan
ln -s `pwd`/scipag_vulscan /usr/share/nmap/scripts/vulscan
fatal: could not create work tree dir 'scipag_vulscan': Permission denied
ln: failed to create symbolic link '/usr/share/nmap/scripts/vulscan': Permission denied
ajay@server1:/usr/share/nmap/scripts$ sudo git clone https://github.com/scipag/vulscan scipag_vulscan
Cloning into 'scipag_vulscan'...
fatal: unable to access 'https://github.com/scipag/vulscan/': Could not resolve host: github.com
ajay@server1:/usr/share/nmap/scripts$ sudo git clone https://github.com/scipag/vulscan scipag_vulscan
Cloning into 'scipag_vulscan'...
fatal: unable to access 'https://github.com/scipag/vulscan/': Could not resolve host: github.com
ajay@server1:/usr/share/nmap/scripts$ nslookup github.com
;; communications error to 127.0.0.53#53: timed out
;; communications error to 127.0.0.53#53: timed out
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   github.com
Address: 140.82.112.3
```

So, go ahead and navigate into scipag_vulnscan and validate that all the data is there.

```
ajay@server1:/usr/share/nmap/scripts$ cd scipag_vulscan/
ajay@server1:/usr/share/nmap/scripts/scipag_vulscan$ ls -l
total 40404
-rw-r--r-- 1 root root       27 Feb 25 00:24 _config.yml
-rw-r--r-- 1 root root    70364 Feb 25 00:24 COPYING.TXT
-rw-r--r-- 1 root root 16756993 Feb 25 00:24 cve.csv
-rw-r--r-- 1 root root  1864748 Feb 25 00:24 exploitdb.csv
-rw-r--r-- 1 root root    53779 Feb 25 00:24 logo.png
-rw-r--r-- 1 root root  1524310 Feb 25 00:24 openvas.csv
-rw-r--r-- 1 root root  6718903 Feb 25 00:24 osvdb.csv
-rw-r--r-- 1 root root     5817 Feb 25 00:24 README.md
-rw-r--r-- 1 root root   683851 Feb 25 00:24 scipvuldb.csv
-rw-r--r-- 1 root root  7227028 Feb 25 00:24 securityfocus.csv
-rw-r--r-- 1 root root  1826138 Feb 25 00:24 securitytracker.csv
-rw-r--r-- 1 root root      361 Feb 25 00:24 update.ps1
-rw-r--r-- 1 root root      320 Feb 25 00:24 update.sh
drwxr-xr-x 4 root root     4096 Feb 25 00:24 utilities
-rw-r--r-- 1 root root    17230 Feb 25 00:24 vulscan.nse
-rw-r--r-- 1 root root  4576711 Feb 25 00:24 xforce.csv
ajay@server1:/usr/share/nmap/scripts/scipag_vulscan$
```

Now its time to use this Vulnerability scanner build within the nmap application.

Grand Moff Tarkin, “You may fire when ready…..”

Step 3

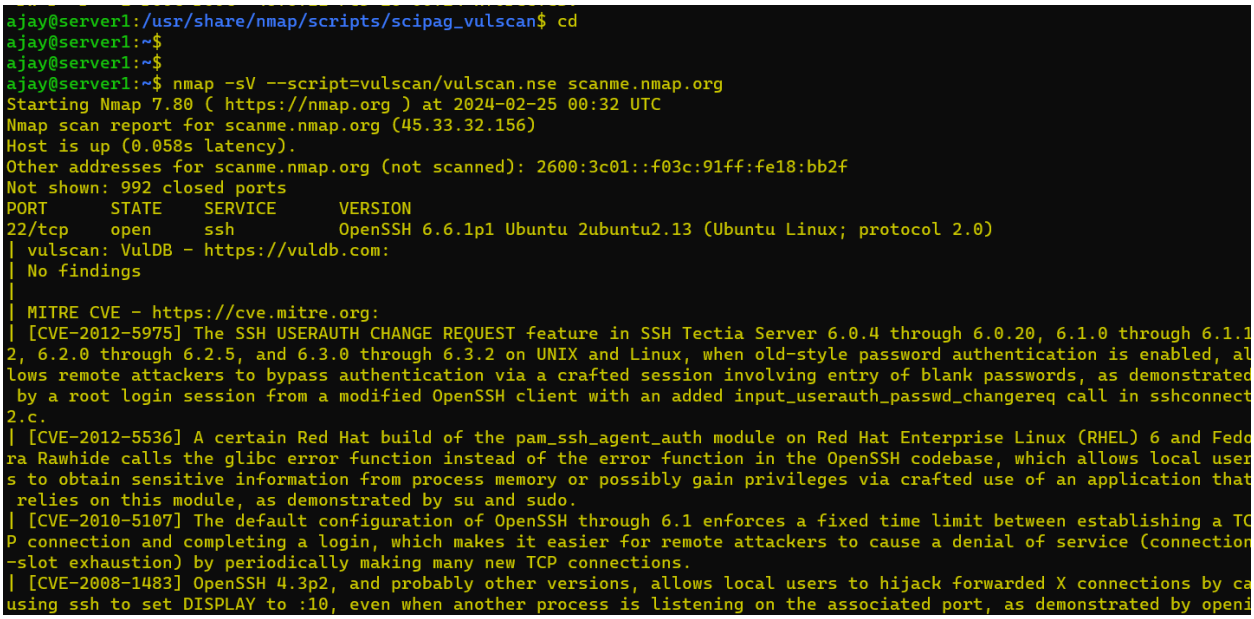
Navigate back to your home folder by using the `cd` command.

nmap syntax in this context

<command> <option1> <option2> <argument>

`nmap -sV --script=vulscan/vulscan.nse scanme.nmap.org`

You will see a long/large output as it identifies all the vulnerabilities on `scanme.nmap.org`



Vulnerability Database used/queried by VulnScan:

There are the following pre-installed databases available at the moment:

- scipvuldb.csv - https://vuldb.com
- cve.csv - https://cve.mitre.org
- securityfocus.csv - https://www.securityfocus.com/bid/
- xforce.csv - https://exchange.xforce.ibmcloud.com/
- expliotdb.csv - https://www.exploit-db.com
- openvas.csv - http://www.openvas.org
- securitytracker.csv - https://www.securitytracker.com (end-of-life)
- osvdb.csv - http://www.osvdb.org (end-of-life)

Now this large output isn't very helpful because it's so large, so its best to redirect this back to a file by using simple linux redirection.

nmap -sV --script=vulscan/vulscan.nse scanme.nmap.org > scanme.nmap.org_vulnscan

This will enable one to read, edit, transform, and work with the output with greater facility.

Github Repo for VULNSCAN: https://github.com/scipag/vulscan

Task 2

Give yourself permission to scan your own network for vulnerabilities. 😂

Obtain your network scope and size. (This should be a task you can complete w/o instructions)

Run a local vulnerability scan on your home network.

EXAMPLE (Your network will likely be different)

nmap -sV --script=vulscan/vulscan.nse 10.0.0.0/24

EXAMPLE (Your network will likely be different)

nmap -sV --script=vulscan/vulscan.nse <your network ID and CIDR>

```
Nmap scan report for 10.0.0.158
Host is up (0.014s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
23/tcp open  telnet  BusyBox telnetd
| vulscan: VulDB - https://vuldb.com:
| [99059] Honeywell Intermec up to 10.11/10.12 Busybox /usr/bin/lua access control
| [97860] BusyBox up to 1.22.x Kernel Module Loader modutils/modprobe.c add_probe input validation
| [94808] BusyBox up to 1.23.1/1.4.x cmdline arp_main memory corruption
| [65544] BusyBox up to 1.16.1 access control
| [29484] BusyBox 1.1.1 unknown vulnerability
|
| MITRE CVE - https://cve.mitre.org:
| [CVE-2011-4862] Buffer overflow in libtelnet/encrypt.c in telnetd in FreeBSD 7.3 through 9.0, MIT Kerberos Version 5 A
pplications (aka krb5-appl) 1.0.2 and earlier, Heimdal 1.5.1 and earlier, GNU inetutils, and possibly other products all
ows remote attackers to execute arbitrary code via a long encryption key, as exploited in the wild in December 2011.
| [CVE-2011-2716] The DHCP client (udhcpc) in BusyBox before 1.20.0 allows remote DHCP servers to execute arbitrary comm
ands via shell metacharacters in the (1) HOST_NAME, (2) DOMAIN_NAME, (3) NIS_DOMAIN, and (4) TFTP_SERVER_NAME host name
options.
| [CVE-2010-4965] /etc/rc.d/rc.local on the D-Link DCS-2121 camera with firmware 1.04 configures a hardcoded password of
 admin for the root account, which makes it easier for remote attackers to obtain shell access by leveraging a running t
elnetd server.
| [CVE-2009-0641] sys_term.c in telnetd in FreeBSD 7.0-RELEASE and other 7.x versions deletes dangerous environment vari
ables with a method that was valid only in older FreeBSD distributions, which might allow remote attackers to execute ar
bitrary code by passing a crafted environment variable from a telnet client, as demonstrated by an LD_PRELOAD value that
 references a malicious library.
| [CVE-2008-0153] telnetd.exe in Pragma TelnetServer 7.0.4.589 allows remote attackers to cause a denial of service (pro
cess crash and resource exhaustion) via a crafted TELOPT PRAGMA LOGON telnet option, which triggers a NULL pointer deref
erence.
| [CVE-2007-0956] The telnet daemon (telnetd) in MIT krb5 before 1.6.1 allows remote attackers to bypass authentication
```

You'll notice that I managed to identify some vulnerabilities within the network I was using. (Don't worry I got permission)

Redirect the output of your report to a file, and then time some time reviewing the CLI out put of the vulnerability scan that you conducted.

Task 3

Next exercise we will be downloading a version of Linux called “DamnVulnerableLinux” 🤯

No, it really is a thing, and yes, it is super vulnerable to things. You’ll never be “INSTALLING” the DVL operating system. You just boot to the ISO to make it work.

You can follow this walk through, or my walk through, it’s ok.
https://www.computersecuritystudent.com/SECURITY_TOOLS/DVL/lesson1/index.html

ajay’s walkthrough downloading DVL:

Download DVL 1.5 https://download.vulnhub.com/dvl/DVL_1.5_Infectious_Disease.iso

1. Create a NEW VM
2. Load the ISO into the VM
3. Set the memory to 512m
4. Set 1 Core of CPU
5. 8gb for Storage
6. Set Networking to BRIDGED
7. When you see BOOT: in the CLI, just hit the ENTER KEY
    a. This will boot you into the DVL
8. Username will be root
9. Password will be toor (so clever, I wonder if a kid came up with that one)

Now that your DVL has loaded, run `ifconfig` to obtain the IP address.

Using the CMD prompt from your host OS, which should still be SSH’ed into your Linux Server – Vulnerability Scanner.

EXAMPLE

nmap -sV --script=vulscan/vulscan.nse 10.0.0.52

use the ip address you just got from the DVL CLI

nmap -sV --script=vulscan/vulscan.nse <your DVL ip address>

Now get ready to have a party. 🎉

It should light up like a Christmas tree.

Go ahead and rerun the command with the output going to a file.

View the file.

Identify 3 vulnerabilities and generate a short executive summary on each.