CS378 Ethical Hacking: Group 5 Luke, Nima, Alfredo, Jesus

Identifying Common Vulnerable Network Services: AUTOSCAN of SSH, NFS, & MySQL

Background and Motivation

We have learned throughout this course that many networks have been exploited due to unpatched vulnerabilities, unused services remaining open, and the sheer complexity of the network as a whole. Just recently in April 2016, the "Panama Papers" incident reinforces how common these mistakes are and how detrimental they can be. Security experts posit that Panamanian law firm Mossack Fonseca were using old, outdated versions of Drupal and WordPress to contain and serve sensitive client information. This left the firm vulnerable to the exploit known as Revolution Slider, which was discovered in October of 2014.

Using events such as these as our motivation, our group scripted a simple BASH tool named AUTOSCAN that would identify and attempt to exploit several popular network services, namely MySQL, NFS and SSH. We chose BASH as our scripting language because it allows for easy use in the Unix and Linux family of operating systems and leverages the many utilities available in these operating systems. In particular, the Metasploit framework, which provides a suite of penetration testing tools and an internal PostgreSQL database to store results. AUTOSCAN follows the network methodology of landscape discovery, footprinting, and exploitation testing:

Network Methodology	Script Portion
Landscape Discovery	ARP Scan to discover and enumerate live hosts
Footprinting	Nmap for active NFS, SSH, & MySQL Services
Service Exploitation	Metasploit performs exploits and store information

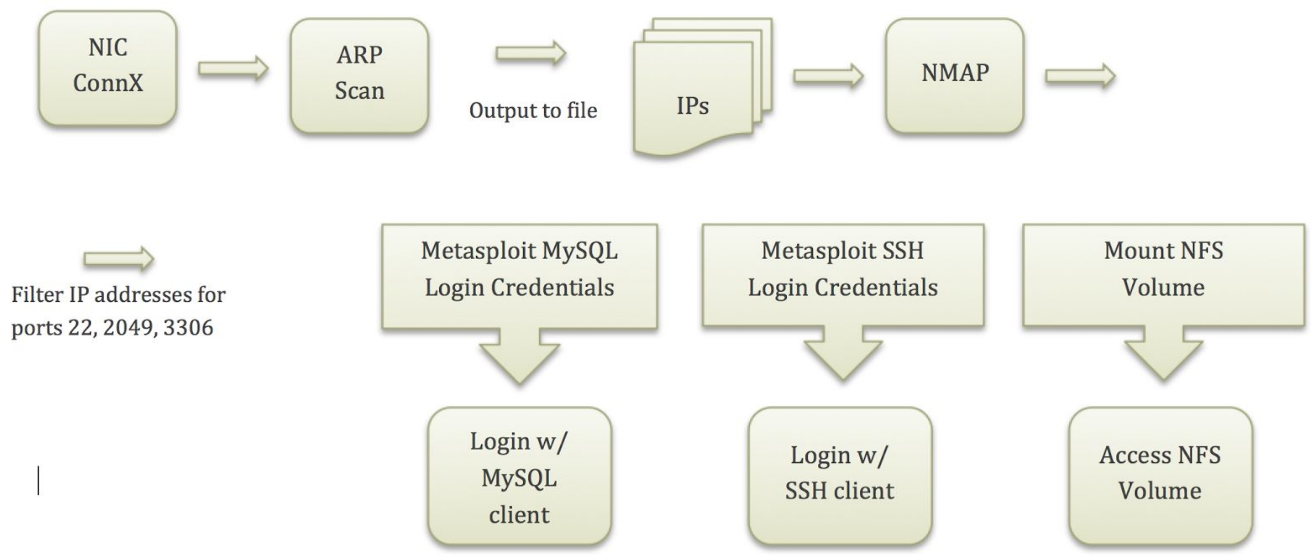
AUTOSCAN Flowchart

The basic flow of our tool is depicted below:

Determine device connected → ARP scan to find local hosts → branch into 3 paths:
1) NFS: find hosts with NFS service → determine which are mountable → mount and navigate to mounted directory

2) MySQL: find hosts with MySQL service → configure and run metasploit bruteforce attack on hosts → store credentials → launch terminal window

3) SSH: find hosts with SSH service → configure and run metasploit bruteforce attack on hosts → store credentials



Landscape Discovery: ARP Scan

The script begins by detecting the active network interface connected to the target network. Next, the tool searches for live hosts using an ARP scan over the local network. The "arp-scan" command was chosen over Nmap in the interest of speed. Arp-scan operates by sending an identification request to all hosts on a network at once, whereas Nmap sends requests serially, taking much longer. The relevant code snippet is below:

```
NIC=$(ip link show | grep 'state UP' | awk -F ': |:' '{print $2}')

arp-scan --localnet
```

Footprinting: NMAP for NFS, SSH, & MySQL Services

Once live hosts have been enumerated, specific service ports are targeted via Nmap. In particular, NFS (port 2049), SSH (port 22), and MySQL (port 3306) services are filtered for. This allows us to isolate only those IP addresses that have those specific ports open. We accomplished this by simply running Nmap with the port flags stipulated:

```
nmap -p22,3306 $ARPRESULT -oX $OUTFILE

nmap -Pn -p$NFS_PORT $ARPRESULT --open
```

For SSH and MySQL the results of the Nmap are output to a file. The Nmap scan for NFS is saved locally in a variable and used instantly.

Once we have identified the potentially vulnerable IP addresses, our script imports and executes 3 separate, individual BASH scripts for each targeted service, named for the services they attack (nfs.sh, ssh.sh and mysql.sh). The use of importing smaller scripts allows us to modularize the tool, which has two main benefits: debugging and extensibility. In future iterations of the project, different services can be designed and debugged separately from the main tool.

The imported NFS script detects if an NFS port is open on a host, these are queried with 'showmount.' If anyone has made the mistake of sharing a home directory on the network, it's ripe for exploitation.

In the case of the SSH and MySQL imported scripts, the procedures are similar. Both use the Metasploit framework to brute-force user/password combinations by assuming standard user names and using a word list to try passwords. Once a valid credential is discovered, it is stored in the MSF built-in database for future use during exploitation.

Service Exploitation

In the case of NFS, exploitation is a matter of searching for improper configurations across all hosts running the service. After hosts running an NFS server are found, the search is done by executing:

```
showmount -e $nfs_host
```

This will either produce an error or display mountable directories. With some BASH parsing, the desired information is passed to the mount command along with a freshly-created mount point in the /tmp/ directory:

```
mount -t nfs $NFS_TARGET:/home/ /tmp/automount
```

From here, the script opens a new terminal window and navigates to the mounted directory so the user can get down to business and traverse the directory.

For MySQL service exploit, the script opens a new a terminal window and begins a mysql client background process that uses the discovered credentials to login to the MySQL service.

```
gnome-terminal --command="mysql -h $Host -u $Username --password=$Password" &
```

The user can then query and manipulate the MySQL database.

In the case of SSH, the exploitation uses the previously acquired data from Nmap and adds it to the metasploit database. It then queries to get the machines with openssh and sets them as RHOSTS by doing : services -p 22 -u -R. The final thing needed is set the username and password list. During the brute force exploitation any credential it finds it will save it to the metasploit database. Additionally, an SSH shell is opened and connects to any machine it found a valid password for.

Problems

A few problems cropped up in the footprinting stage that had to be dealt with or circumvented. A minor problem (mentioned earlier) was the performance hit that using Nmap incurred on host discovery. It would have been much simpler to just use Nmap within the Metasploit framework to store enumerated hosts in the built-in database, but the increase in runtime was unacceptable. Instead, we used BASH scripts to parse the output of arp-scan into a modular format that could easily be inputted to Nmap.

Another issue came with attempts to spoof a mac address. Nmap has a feature to spoof mac addresses, but none of the queries succeeded with that option on. Further, macchanger requires the network adapter to be disconnected from a network, so we attempted to disconnect beforehand, and reconnect to the same network afterwards, but the interaction with the network manager command line interface seemed unreliable and slow, so we omitted the spoofing step from our process entirely.

When we were setting up our MySQL script, one problem that we ran into was how difficult it was to extract credentials from the built-in Metasploit database. The credentials database didn't allow us to traverse to the specific username and password of a given host with any command line flags. We were able to solve this problem by spooling our console into a log file before we started running our MySQL script so that we could grep the specific username and password out of the log file and into the next step of our automation.

Another issue we had using Nmap within the metasploit framework was using the db_nmap command to automatically store the results of the scan into the database. The command would only store the first host in the results into our database and would result in a loss of data. We worked around this by exporting the results of the Nmap scan to a local file, and then importing that file back into the metasploit database.

In the SSH script a major problem that was encountered was the network dropping our connection during the brute force execution. This problem was resolved by running the brute force at a slower pace to prevent the network from dropping our connection. Additionally, some problems were encountered with the database in metasploit as it was keeping the previous IPs and services from earlier sessions. The file was appending previous runs of the script but this problem was resolved by creating a new workspace in metasploit each time the script is run.

Improvements

The footprinting stage is pretty effective-- the ARP scan results are fast and accurate. For the Nmap scanning portion, we could probably stand to unify some of the scans and rely more on the Metasploit database to manage information instead of using BASH for some variables.

Improvements to the NFS service explorer could be made; currently, the script simply mounts the first exploitable host's shared directory and presents the user with a terminal to navigate it. A better procedure would be to incorporate user and group name spoofing. With this, the script could crawl through all of the subdirectories, emulating the various users required for access, dumping all of the files from the mounted file system locally for later perusal-- the less time you have to spend interacting with the victim's system, the less likely you are to leave a footprint of your own.

Improvements to the MySQL brute force script could be made by going a step further than the MySQL command line and actually dropping a table or pulling valuable data. We could potentially feed the MySQL client a file with pre-typed SQL statements. Another improvement we could do is to run the mysql_hashdump exploit from within Metasploit and grab the password hashes from all of the database user accounts and crack them with John the Ripper..

Improvements to the SSH brute force service can be made by allowing the user to select specific targets. Currently the script looks for all the machines with SSH service open and tries to gain brute force access to them which can be a time consuming process if doing so for a large network. Another improvement that could be made is adding a larger list of username and passwords the current implementation tries to gain root access by using only the username root as well as the list of passwords from rockyou.txt.