SSL, GONE IN 30 SECONDS
A BREACH beyond CRIME
AGENDA

<table>
<thead>
<tr>
<th>Proceed with caution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Review of <strong>CRIME</strong></td>
</tr>
<tr>
<td>✓ Introducing <strong>BREACH</strong></td>
</tr>
<tr>
<td>✓ <strong>In the</strong> <strong>weeds</strong></td>
</tr>
<tr>
<td>✓ <strong>Demo</strong> <strong>time!</strong></td>
</tr>
<tr>
<td>✓ <strong>Mitigations</strong></td>
</tr>
</tbody>
</table>
PREVIOUSLY...

**CRIME**  
Presented at ekoparty 2012

**Target**  
Secrets in HTTP headers

**Requirements**  
TLS compression  
MITM  
A browser

Juliano Rizzo  
Thai Duong
SO ABOUT CRIME...

The Compression Oracle:

✓ SSL doesn’t hide length

✓ SSL/SPDY compress headers

✓ CRIME issues requests with every possible character, and measures the ciphertext length

✓ Looks for the plaintext which compresses the most – guesses the secret byte by byte

✓ Requires small bootstrapping sequence

\[
\text{knownKeyPrefix}=\text{secretCookieValue}
\]
COMPRESSION OVERVIEW

- **DEFLATE / GZIP**
  - **LZ77**: reducing redundancy
    - Googling the googles -> Googling the g(-13,4)s
  - Huffman coding: replace common bytes with shorter codes
In most cases you can rely on clients having been patched to disable compression. If you want to perform this, you can test for SSL Compression using the SSL Labs service "Compression" in the Miscellaneous section or using "openssl s_client -connect <hostname>:443 -c <sslcompression>".

If you have Compression enabled, the method of doing so varies depending on your hardware device or software not listed here, you'll need to disable SSL Compression - it shouldn't be confused with SSL Compression.

Apache 2.4 using mod_ssl
Apache 2.4.3 has support for the SSLCompression feature by default.
August, 2012. SSLCompression is on by default.

TLS Compression Disabled

As part of the coordination process, we would like some clarification regarding this vulnerability. Is this vulnerability, specific to HTTPS responses, also mitigated by the same methods as the original CRIME vulnerability in HTTPS requests (CVE-2012-4929)? It is our understanding that patches have been released for modern web browsers and web servers that mitigate the original CRIME vulnerability, namely by disabling HTTPS compression, and we were wondering if you could confirm if these mitigations prevent the vulnerability you have submitted.

If you have any questions or concerns, please let us know.

Best Regards,

[Signature]

Vulnerability Analysis Team

CERT(R) Coordination Center  |  cert@cert.org
Software Engineering Institute  |  Hotline : +1.412.268.7090
DO NOT PANIC:

TUBES SECURE
Or are they?
let's bring it back to life

Liquid Nitrogen Cat being prepared for resuscitation
FIRST THINGS FIRST: FIX WIKIPEDIA

As of September 2012, the CRIME exploit has been mitigated by the latest versions of the Chrome and Firefox web browsers, and Microsoft has confirmed that their Internet Explorer browser was not vulnerable to the exploit. Some websites have applied countermeasures at their end.
NEW!

INTRODUCING BREACH

Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext
A CRIME AGAINST THE RESPONSE BODY

COMPRESS

ALL THE THINGS
(sample traffic)

GET http://www.microsoft.com/en-us/default.aspx HTTP/1.1
Accept: text/html, application/xhtml+xml, 
Accept-Language: en-US, en; q=0.8, es-ES; q=0.5, es; q=0.3
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)
Accept-Encoding: gzip, deflate
Host: www.microsoft.com
DNT: 1
Connection: Keep-Alive
Cookie: MC0=1375073809391; msdn=L=en-US; WT_FPC=id=29f8c879426e0c24a2f1373520155467; NAP=V=1.9&E=dfe&C=rhw...
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 16398
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
X-Powered-By: ASP.NET
X-Powered-By: ARR/2.5
X-Powered-By: ASP.NET
Date: Mon, 29 Jul 2013 04:56:24 GMT

blackhatUSA 2013
**BREACH / the ingredients**

- **GZIP**
  - Very prevalent, any browser

- **Fairly stable pages**
  - Less than 30 seconds for simple pages

- **MITM / traffic visibility**
  - No SSL tampering / downgrade

- **A secret in the response body**
  - CSRF, PII, ViewState... anything!

- **Attacker-supplied guess**
  - In response body

- **Three-character prefix**
  - To bootstrap compression

- **Any version of SSL / TLS**
SSL, GONE IN 30 SECONDS

[ PREFIX / sample bootstrap ]
BREACH / architecture

SSL, GONE IN 30 SECONDS
SSL, GONE IN 30 SECONDS

BREACH / command & control

evil-hacker.com/breach

Web Server Driver :81
(iframe streaming)

Web Server :82
(event callback listener)

MITM (ARP/DNS...)

Basic Oracle Logic

Traffic Monitor
(Packet filter & Length)

Advanced C&C Engine

SECURED BY 128 BIT SSL ENCRYPTION
C&C/ logic

✔ Traffic Monitor
  - Transparent relay SSL proxy

✔ HTML/JS Controller
  I. Dynamically generated for specific target server
  II. Injects & listens to iframe streamer from c&c:81 that dictates the new HTTP requests to be performed (img.src=…)
  III. Issues the outbound HTTP requests to the target site via the victim's browser, session-riding a valid SSL channel
  IV. Upon synchronous completion of every request (onerror), performs a unique callback to c&c:82 for the Traffic Monitor to measure encrypted response size

MITM: ARP spoofing, DNS, DHCP, WPAD…
C&C/ logic

✔ Main C&C Driver
  - Coordinates **character guessing**
  - Adaptively **issues requests** to target site
  - Listens to **JS callbacks** upon request completion
  - **Measures** -inbound- packets **length**
  - Has built-in intelligence for **compression oracle runtime recovery**
THE ORACLE

| MEASURE SIZE DELTA | GUESSING BYTE-BY-BYTE | ERROR RECOVERY |

Science Cat is stealing your Internets
SSL REVEALS LENGTH

TCP connection

SSL records

HTTP clear text

SSL cipher text

10 bytes
**COMPRESSSION ORACLE (I)**

```
<html>
...
  tkn=supersecret
  ...
  guess=supersecreX
</html>
```

- **48 bytes**

```
<html>
...
  tkn=supersecret
  ...
  guess=(-22, 10)X
</html>
```

- **38 bytes**

**after gzip**
COMPRESSION ORACLE (II)

```
<html>
  ...
  tkn=supersecret
  ...
  guess=supersecret
</html>
```

48 bytes

```
<html>
  ...
  tkn=supersecret
  ...
  guess=(-22, 11)
</html>
```

37 bytes
Safety begins with you.
THE ORACLE
Huffman Coding Nightmares

✓ **Correct** Guess

https://target-server.com/page.php?blah=blah2...
&secret=4bf b (response: 1358 bytes)

✓ **Incorrect** Guess

https://target-server.com/page.php?blah=blah2...
&secret=4bf a (response: 1358 bytes)
THE ORACLE
Fighting Huffman Coding

✓ Two Tries + random [dynamic] padding

https://target-server.com/page.php?blah=blah2...
&secret=4bf7{}{}(...){}{}{}{}
&secret=4bf{}{}(...){}{}{}{} 7

✓ Character set pool + random padding

https://target-server.com/page.php?blah=blah2...
&secret=4bf7{}{}(...){}{}{}{}---a-b-c-d---5-6-8-9---
&secret=4bf8{}{}(...){}{}{}{}---a-b-c-d---5-6-7-9---
THE ORACLE
Two Tries Reality

✔ Less than ideal conditions:
  - In theory, **two-tries** allows for short-circuiting once winner is found
  - In practice, still need to evaluate all candidates
  - **Huffman encoding** causes collisions
RODBLOCKS

✓ Conflict & Recovery mechanisms
  (no winners / too many winners)
  - **Look-ahead** (2+ characters) – reliable, but expensive
    - Best value / averages
  - **Rollback** (last-known conflict)
  - Check **compression ratio** of guess string

✓ Page URL / HTML entity encoding
  - Can interfere with **bootstrapping**

```html
<input type="hidden" value="b95825dd78a7ccc95f1f6f5a62b247f753fc2a5d"
    name="authenticity_token" class="authenticity_token">
<data-query="Can I Haz _token value='?">
MORE ROADBLOCKS

✓ Stream cipher vs. block cipher

Stream cipher reveals exact plain text length

10 bytes

Compressed HTTP response
SSL cipher text
MORE ROADBLOCKS

✔ Stream cipher vs. block cipher

| Block cipher **hides** exact plain text length

16 bytes

- Compressed HTTP response
- SSL cipher text

- Align response to a tipping point
- Guess Window (keeping response aligned)
EVEN MORE ROADBLOCKS

✔ Keep-Alive (a premature death)
  - Image requests vs. scripts vs. CORS requests

✔ Browser synchronicity limits (1x)
  - Hard to correlate HTTP requests to TCP segments

✔ Filtering out noise
  - Active application?
  - Background polling?
YET MORE ROADBLOCKS

✓ ‘Unstable’ pages \textit{(w/ random DOM blocks)}
  - Averaging & outlier removal

✓ The war against \textbf{Huffman coding}
  - Weight (symbol) normalization

✓ Circumventing cache
  - Random timestamp

✓ Other Oracles
  - \textit{Patent-pending!}
OVERWHELMED?

COCAINEx

SO MUCH COCAINE.
SSL, GONE IN 30 SECONDS

DEMO TIME
(let us pray)
MITIGATIONS

<table>
<thead>
<tr>
<th>RANDOMIZING THE LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>· variable padding</td>
</tr>
<tr>
<td>· fighting against math</td>
</tr>
<tr>
<td>· /FAIL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DYNAMIC SECRETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>· dynamic CSRF tokens per request</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEPARATING SECRETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>· deliver secrets in input-less servlets</td>
</tr>
<tr>
<td>· chunked secret separation (lib patch)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CSRF-PROTECT EVERYTHING</th>
</tr>
</thead>
<tbody>
<tr>
<td>· unrealistic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASKING THE SECRET</th>
</tr>
</thead>
<tbody>
<tr>
<td>· random XOR – easy, dirty, practical path</td>
</tr>
<tr>
<td>· downstream enough</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>THROTTLING &amp; MONITORING</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DISABLING GZIP FOR DYNAMIC PAGES</th>
</tr>
</thead>
</table>

SSL, GONE IN 30 SECONDS
FUTURE WORK

- Better understanding of DEFLATE / GZIP
- Beyond HTTPS
  - Very **generic** side-channel
  - Other protocols, contexts?
- Stay tuned for the next BREACH
THANK YOU EVERYBODY!

WHO’S AWESOME?
You’re Awesome!
BREACHATTACK.COM

Angelo Prado
angelpm@gmail.com
@PradoAngelo

Neal Harris
neal.harris@gmail.com
@IAmTheNeal

Yoel Gluck
yoel.gluck2@gmail.com

If you liked the talk*, don’t forget to scan your badge for the evaluation survey
* ignore otherwise