SSL, GONE IN 30 SECONDS
A BREACH beyond CRIME
PREVIOUSLY...

CRIME
Presented at
ekoparty 2012

Juliano Rizzo
Thai Duong

Target
Secrets in HTTP
headers

Requirements
TLS compression
MITM
A browser
COMPRESSION OVERVIEW

✓ DELATE:
  ▪ LZ77: reducing bits by reducing redundancy
    • Googling the googles → Googling the g(-13,4)s
  ▪ Huffman coding: reducing bits by employing an entropy encoding algorithm
    • aka. replace common bytes with shorter codes
SO ABOUT CRIME...

The Compression Oracle:

- SSL doesn’t hide length
- TLS/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

knownKeyPrefix=secretCookieValue
IT’S FIXED!

- TLS Compression Disabled

---

**Vulnerability**

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.¹ Some websites have applied countermeasures at their end.⁷
IT’S FIXED!

In most cases you can rely on clients having been patched to disable SSL Compression server-side also. You can test for it in the Miscellaneous section or using iSEC Partners’ SSL Compression test tool.

If you have Compression enabled, the method of disabling it in a hardware device or software not listed here, you’ll need to disable SSL Compression - it shouldn’t be confused with HTTP compression.

Apache 2.4 using mod_ssl

Apache 2.4.3 has support for the SSLCompression flag. This feature was introduced on August, 2012. SSLCompression is on by default - to disable it, use the setenv SSL_COMPRESSION off directive in the server or rewrite engine configuration.

As part of the coordination process, we’d 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-2012-4909)? Our understanding is 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@cert.org

CERT(R) Coordination Center
Software Engineering Institute | Hotline: +1 412.288.7090

 breach
SSL, GONE IN 30 SECONDS
DO NOT PANIC »

100% GUARANTEED SAFETY

« IT’S FIXED

SSL, GONE IN 30 SECONDS
let’s bring it back to life

Liquid Nitrogen Cat being prepared for resuscitation
SSL, GONE IN 30 SECONDS

INTRODUCING BREACH

Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext
BREACH / the ingredients

<table>
<thead>
<tr>
<th>GZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Very prevalent</td>
</tr>
<tr>
<td>· Highly <strong>impractical</strong> to turn off</td>
</tr>
<tr>
<td>· <strong>Any</strong> browser, <strong>any</strong> web server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fairly stable pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>· It only takes <strong>one</strong></td>
</tr>
<tr>
<td>· <strong>Less than 30 seconds</strong> for simple pages</td>
</tr>
<tr>
<td>· Minutes to hours for more complicated dynamic bodies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MITM / traffic visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>· No tampering / SSL downgrade</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSL / TLS [any version]</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Could be turned off ;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A secret in the response body</th>
</tr>
</thead>
<tbody>
<tr>
<td>· CSRF, SIDs, PII, ViewState...</td>
</tr>
<tr>
<td>· and <strong>much more</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attacker-supplied data</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Guess (in response body)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Three-characters prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>· To <strong>bootstrap compression</strong></td>
</tr>
</tbody>
</table>
SSL, GONE IN 30 SECONDS

[PREFIX / sample bootstrap]

- Guess (in response body)
- Target secret (CSRF token)
BREACH / architecture

VICTIM

C&C CENTER

TARGET SERVER

THE TUBES

E2E SSL

ROUTER
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

breach SSL, GONE IN 30 SECONDS
ORACLE

<table>
<thead>
<tr>
<th>ONE CHARACTER AT A TIME</th>
<th>AIRBAGS</th>
<th>COLLISIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Guessing byte-by-byte</td>
<td>· Random amount of padding</td>
<td>· Attempt recovery for multiple winners</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Detect &amp; roll-back from wrong path</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TWO TRIES</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>· Issue two HTTPs requests per guess</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

✓ Guess Swap
  ▪ Swap last two characters in the guess
  ▪ Measure overall size increase
  https://target-server.com/page.php?blah=blah2...
    &secret=4bf7
    &secret=4bf7

✓ Character set pool (to eliminate Huffman tree changes between guesses)
  ▪ Add all characters to all guesses, shifting the guessed character into position
  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-...
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 website
  ▪ Listens to **JS callbacks** upon **request completion**
  ▪ Oracle measures -inbound- packets **length**
  ▪ Has built-in intelligence for **conflict resolution** and **recovery**
ROADBLOCKS

✓ 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

✓ Conflict resolution & recovery mechanisms (I)
  (In case of conflict / no winners)

1. Dynamic **airbags**
2. **Look-ahead** (2+ characters) – more reliable, but more expensive
   • Best value
   • Averages
ROADBLOCKS

✅ Conflict resolution & recovery mechanisms (II)
  - Rollback (in-memory path, last-known conflict)
  - Detect substrings in secret/guess
    - Check compression ratio of guess string

✅ Page URL / HTML entity encoding
  - Can interfere with collision bootstrapping and secret key-space
MORE ROADBLOCKS

 ✓ Circumventing cache
   ▪ For targets & callback – random timestamp

 ✓ Block mode vs. stream cipher mode
   ▪ Align response to a tipping point and overflow into the next block
   ▪ Guess Window (keeping response aligned) – as we add characters to the guess, we remove others
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 (w/ random DOM blocks)
  ▪ Averaging – statistical outlier removal and detection

✓ Collateral effects of Huffman tree
  ▪ Weight (symbol) normalization

✓ Other Misc. Oracles
  ▪ Patent-pending
OVERWHELMED?

COCAIN.

SO MUCH COCAINE.
DEMO TIME
(let us pray)
SSL, GONE IN 30 SECONDS

THE TOOL
MITIGATIONS

- **RANDOMIZING THE LENGTH**
  - variable padding
  - fighting against math
  - /FAIL

- **SEPARATING SECRETS**
  - deliver secrets in input-less servlets
  - chunked secret separation (lib patch)

- **DYNAMIC SECRETS**
  - dynamic CSRF tokens per request

- **MASKING THE SECRET**
  - random XOR – easy, dirty, practical path
  - downstream enough

- **CSRF-PROTECT EVERYTHING**
  - unrealistic

- **THROTTLING & MONITORING**

- **DISABLING GZIP FOR DYNAMIC PAGES**
FUTURE WORK

✅ Better understanding of DEFLATE / GZIP

✅ Beyond HTTPS

- Very generic side-channel
- Other protocols, contexts?

✅ Stay tuned for the next BREACH
WANT MORE?

BreachAttack.com

PAPER | PRESENTATION | POC TOOL
THANK YOU EVERYBODY!

Angelo Prado
angelpm@gmail.com
@PradoAngelo

Neal Harris
neal.harris@gmail.com
@IAmTheNeal

Yoel Gluck
yoel.gluck2@gmail.com

Don’t forget to fill out* the questionnaire if you liked it
* ignore otherwise

BreachAttack.com