Let the BEAST of CRIME and TIME be not so LUCKY

Pratik Guha Sarkar
Takeaway

Background
- BEAST
- Lucky 13
- RC4 biases
- CRIME
- BREACH
- TIME

Prevention
- With and
- without
- TLS 1.2
BEAST - What is it?

- **Browser Exploit Against SSL/TLS**
- Used crypto flaws in SSL to recover plaintext cookies
- Refined previous attacks on CBC in SSL to make them practical
  - Innovation was exploiting *chosen boundary* capability
CBC Attack in SSL
CBC Attack in SSL

Target Block For Decrypting

= 

? 

= 

Ciphertext block attacker can observe
CBC Attack in SSL

Plaintext, attacker doesn’t know, but wants to

Prior Ciphertext Block (Attacker observes)

Encrypted!

=Ciphertext block attacker can observe
CBC Attack in SSL

Ciphertext block attacker can observe?
CBC Attack in SSL

XOR

Plaintext, of attacker’s choosing

Prior Ciphertext Block (Attacker observes)

Encrypted!

Ciphertext block attacker can observe
CBC Attack in SSL

What shall we choose?

XOR

Prior Ciphertext Block (Attacker observes)

Encrypted!

Plaintext, of attacker’s choosing

Ciphertext block attacker can observe
CBC Attack in SSL

What shall we choose?

Prior Ciphertext Block

XOR

Prior Ciphertext Block (Attacker observes)

Encrypted!

Plaintext, of attacker’s choosing

Ciphertext block attacker can observe
CBC Attack in SSL

What shall we choose?

Prior Ciphertext Block = XOR Prior Ciphertext Block (Attacker observes)

Plaintext, of attacker’s choosing

Ciphertext block attacker can observe

Encrypted!
CBC Attack in SSL

What shall we choose?

Prior Ciphertext Block

Guess of Plaintext
We want to learn

Prior Ciphertext Block

XOR

Plaintext, of attacker’s choosing

Prior Ciphertext Block (Attacker observes)

Encrypted!

Ciphertext block attacker can observe
CBC Attack in SSL

Guess of Plaintext
We want to learn

XOR

Inverse of Plaintext
Ciphertext Block

Ciphertext block attacker can observe

Prior Ciphertext Block

Encrypted!

Guess of Plaintext
We want to learn

XOR
CBC Attack in SSL

Encrypted! = Ciphertext block attacker can observe

Guess of Plaintext
We want to learn

XOR

Prior Ciphertext Block

Encrypted!

Ciphertext block attacker can observe
CBC Attack in SSL

• Remember This Slide?
CBC Attack in SSL

- Remember This Slide?
CBC Attack in SSL

Plaintext, attacker doesn’t know, but wants to

XOR

Prior Ciphertext Block

Encrypted!

Guess of Plaintext

We want to learn

XOR

Prior Ciphertext Block

Encrypted!

Ciphertext block attacker can observe
CBC Attack in SSL

- plaintext, attacker doesn't know, but wants to
- Prior ciphertext block (Attacker observes)
- Encrypted!

Prior ciphertext block attacker can observe

\[ \text{Guess of Plaintext} \]

\[ \text{We want to learn} \]

\[ XOR \]

\[ = \]

\[ \text{Encrypted!} \]

\[ = \]

\[ \text{Ciphertext block attacker can observe} \]
CBC Attack in SSL

- Plaintext, attacker doesn't know, but wants to
- XOR
- Prior Ciphertext Block (Attacker observes)
- Encrypted!

\[ \text{Prior Ciphertext Block} \oplus \text{Guess of Plaintext} = \text{Encrypted!} \]

Ciphertext block attacker can observe
CBC Attack in SSL

- Plaintext, attacker doesn’t know, but wants to

  XOR

  Prior Ciphertext Block (Attacker observes)

  Encrypted!

  XOR

  Ciphertext block attacker can observe

  Encrypted!

  =

  Guess of Plaintext
  We want to learn

  XOR

  Prior Ciphertext Block

  =

  Encrypted!

  =

  Ciphertext block attacker can observe

If we guessed right, we will see an output that MATCHES a cipher text block we saw previously!
CBC Attack in SSL

If we guessed right, we will see an output that MATCHES a ciphertext block we saw previously!
HTTP Request – What Do I Know?

POST /login HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0)
Gecko/20100101 Firefox/16.0
Cookie: a=secrets298fc1c149afbf4c8996fb924

I know everything but the cookie!
# Chosen Boundary in a Slide

<table>
<thead>
<tr>
<th>What the attacker knows</th>
</tr>
</thead>
<tbody>
<tr>
<td>What the attacker DOESN’T know.</td>
</tr>
<tr>
<td>Different</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G</th>
<th>E</th>
<th>T</th>
<th>/</th>
<th>...</th>
<th>C</th>
<th>o</th>
<th>k</th>
<th>i</th>
<th>e</th>
<th>:</th>
<th>a =</th>
<th>s</th>
<th>e</th>
<th>c</th>
<th>r</th>
<th>e</th>
<th>t</th>
<th>...</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>
# Chosen Boundary in a Slide

<table>
<thead>
<tr>
<th>What the attacker knows</th>
<th>What the attacker DOESN’T know.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different</td>
<td>Ciphertext</td>
</tr>
<tr>
<td></td>
<td>Blocks</td>
</tr>
</tbody>
</table>

Attacker pads to a block boundary!

---

<table>
<thead>
<tr>
<th>GET</th>
<th>/</th>
<th>...</th>
<th>Cookie:</th>
<th>a = secret</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>↓</td>
<td></td>
<td></td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GET</th>
<th>/</th>
<th>A</th>
<th>...</th>
<th>1</th>
<th>1</th>
<th>\</th>
<th>\</th>
<th>Cookie:</th>
<th>a = secret</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>↓</td>
<td></td>
<td></td>
<td>↓</td>
<td></td>
<td>↓</td>
<td></td>
<td></td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>

Put It Together: BEAST

- Attacks SSL 3.0 TLS 1.0 with CBC Cipher suites
- Steals Cookies
- Works on HTTPS-only sites
  - Sorry Paypal
BEAST – Feasibility

• Ability to eavesdrop on the network

• Force victim to visit attackers page

• Ability to inject plaintext in an active SSL/TLS session
BEAST – Counter Measures

• Upgrade browsers

• Enable TLS 1.1, preferably 1.2

• Use RC₄
Interlude: Protocol Downgrades

• We’ve mentioned TLS 1.1 and 1.2
• They’re great!
• There’s a problem:
Interlude: Protocol Downgrades

• We’ve mentioned TLS 1.1 and 1.2
• They’re great!

• There’s a problem:

They provide no security at all against an active attacker
Interlude: Protocol Downgrades

User → TLS 1.2 → Server
drops the packet

User → TLS 1.1 - Browser retries!
drops the packet

User → TLS 1.0 - Browser retries!
drops the packet

User → SSLv3 No Extensions - Browser retries!
“Success”
Interlude: Protocol Downgrades

Why Do Browsers Support Fallback?

- Networks Are Hostile to TLS 1.1+
  - Middleboxes don’t recognize it and choke

- Sites Can’t Speak TLS 1.1+
  - Sometimes an error (not so bad)
  - Sometimes they just hang (quite bad)
Interlude: Protocol Downgrades

- Until Browsers Remove Fallback to TLS 1/SSLv3 we cannot fully rely on TLS 1.1+
- Until sites stop breaking for TLS 1.1+ Browsers can’t Remove the Fallback

- Not to call anyone out…. But....
  - [https://www.imperialviolet.org/2013/10/07/f5update.html](https://www.imperialviolet.org/2013/10/07/f5update.html)
Lucky 13

DO YOU FEEL LUCKY.... PUNK?

WELL.... DO YA?
Lucky 13 - What is it?

- Successor of Padding Oracle Attack
- Timing attack on CBC encryption mode
- 13 bytes of header information in TLS MAC calculation leaks timing information during decryption.
Lucky 13 – How it works

Time to validate Padding
Lucky 13 – How it works

- Time to validate MAC
- Time to validate Padding
Lucky 13 – How it works

Time to validate Padding
Lucky 13 – How it works

Data | MAC Tag | Padding

Validate Padding

Data | MAC Tag

Validate MAC

Error

Ideal time to validate MAC
Time to validate Padding
Lucky 13 – How it works

- Extra time to validate MAC
- Ideal time to validate MAC
- Time to validate Padding
Lucky 13 – Feasibility

• Needs Man-in-the-middle

• CBC-mode encryption in versions of TLS are potentially vulnerable.

• Requires huge number of request

• Requires no Network jitter
Lucky 13 – Counter Measures

• Uniform processing time to decrypt ciphertexts

• Add random timing delays to the decryption for any timing attack

• Using stream cipher like RC4

• Using an authenticated encryption algorithm, such as AES-GCM
RC⁴ Biases - What is it?
RC4 Biases - What is it?
RC4 Biases – Feasibility

• Force victim to renegotiate.

• This attack will require over 4 billion SSL connections or re-negotiations for an individual HTTP session.
RC4 Biases – Counter Measures

• Researchers still working on finding mitigations of this issue.

• Temporary mitigations
  • Throttle client initiated re-negotiations and connections from individual IP addresses
  • If possible use block ciphers with mitigations of timing and CBC mode encryption attack mitigated
Compression
How compression works?

• DEFLATE compression mechanism
CRIME - What is it?

- Compression Ratio Info-leak Made Easy
- Chosen plaintext attack on HTTP request
- Uses size information in TLS compression to recover plaintext cookies
CRIME – How it works

HTTPS Connection

Unencrypted Connection

JavaScript

CRIME Attacker

Victim User
CRIME – How it works

GET /evil_request_path HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Cookie: sessionid=d3b0c44298fc1c149afbf4c8996fb924

Attacker doesn’t control entire request, but can see its cipher text on the wire
CRIME – How it works

GET /evil_request_path HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Cookie: sessionid=d3b0c44298fc1c149afbf4c8996fb924

Attacker fully controls request path
GET /evil_request_path HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Cookie: sessionid=d3b0c44298fc1c149afbf4c8996fb924

Attacker does not see, but can infer these values
CRIME – How it works

GET /evil_request_path HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Cookie: sessionid=d3b0c44298fc1c149afbf4c8996fb924

Attacker cannot see/control, wants to steal
CRIME – How it works

GET /sessionid=a HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Cookie: sessionid=d3b0c44298fc1c149afbf4c8996fb924

=> Compressed Length = 12,494 bytes – Not a match

GET /sessionid=d HTTP/1.1
Host: bank.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Cookie: sessionid=d3b0c44298fc1c149afbf4c8996fb924

=> Compressed Length = 12,493 bytes – Possible match
CRIME – Feasibility

- The attacker can intercept the victim's network traffic.
- Victim authenticates to a website over HTTPS and negotiates TLS Compression with the server.
- Victim accesses a non-HTTPS website.
- Browser supporting TLS Compression
CRIME – Counter Measures

- Disabling TLS compression on both Browser and Server side.
- Updated Browser versions:
  - Chrome: 21.0.1180.89 and above
  - Firefox: 15.0.1 and above
  - Opera: 12.01 and above
  - Safari: 5.1.7 and above
- Apache 2.2 using mod_SSL:
  SSLCompression flag is set to “SSLCompression off”
- Apache using mod_gnutls:
  GnuTLSPriorities flag = “!COMP-DEFLATE"
BREACH
BREACH – What is it?

- **Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext**

- Chosen plaintext attack on HTTP response

- Uses difference of response size information in due to varying sizes of HTTP compression to recover plaintext secret information

- Resurrection of CRIME
BREACH – How it works

...
<td nowrap id="tdErrLgf">
<a href="logoff.aspx?CSRFtoken=4bd634cda846fd7cb4cb0031ba249ca2">Log Off</a></td>

Attacker can control this value
Attacker cannot control this parameter, want to steal it
BREACH – How it works

GET /product/?id=12345&user=CSRFtoken=a HTTP/1.1
Host: example.com

<form target="https://example.com:443/products/catalogue.aspx?id=12345&user=CSRFtoken=a" >
...
<td nowrap id="tdErrLgf">
<a href="logoff.aspx?CSRFtoken=4bd634cda846fd7cb4cb0031ba249ca2">Log Off</a></td>
</form>

Size of response < Previous size = Match
Size of response >= Previous size = Mismatch
• The application supports **HTTP compression**.

• The response should **reflect** back user's input.

• The response should have some **sensitive/secret** information embedded in the body.
BREACH – Counter Measures

• Mask the secret:
  • new secret = random || (random $\oplus$ previous secret)

• Enable anti-automation techniques

• Monitor your traffic

• Separate secrets from user input

• Disable HTTP compression
TIME - What is it?

- **Timing Info-leak Made Easy**

- Chosen plaintext attack on HTTP response

- Uses difference of response time information in due to varying sizes of HTTP compression to recover plaintext secret information

- Resurrection of CRIME
TIME – How it works

MTU = 1500 Bytes

Data packet

Packet size < MTU

MTU: Maximum Transmission Unit
RTT: Round Trip Time
TIME – How it works

MTU = 1500 Bytes

Data packet

Packet size = MTU

MTU: Maximum Transmission Unit
RTT: Round Trip Time
TIME – How it works

MTU = 1500 Bytes

Data packet

Packet size > MTU

MTU : Maximum Transmission Unit
RTT: Round Trip Time
TIME – How it works

MTU = 1500 Bytes

Data packet 1

Data packet 2

Packet size <= MTU

MTU : Maximum Transmission Unit
RTT: Round Trip Time
TIME – How it works

Add articles - guess@gmail.com

Find articles that you've written and add them to your profile. Later, you can edit or delete the articles in your profile or add more articles to your profile.

author: guess@gmail.com

Screenshot credit - Tal Be'ery BH presentation
TIME – How it works

- MTU = 1500 Bytes
- Data packet
- Padding

Case 1

Data packet Padding

Case 2

Data packet 1 Padding

Data packet 2

1 extrabyte

RTT

MTU: Maximum Transmission Unit
RTT: Round Trip Time
TIME – Feasibility

- No requirements for Man-in-the-Middle

- Concentrate on HTTP responses

- The attacker creates HTTP request with JavaScript and response timing leaks the request size.

- Repeat for few times to void aberration due to network jitter.
TIME – Counter Measures

• Adding random timing delays to the decryption

• Browser should support and respect `X-Frame-Options`

• Strict restriction on reflection of user input in the response.

• Enable anti-automation techniques like CAPTCHA, CSRF token
Anti-Automation Recommendations

• Rate limiting using HAProxy

• Rate limiting via various DDOS protection
## Comparison of counter measures

<table>
<thead>
<tr>
<th></th>
<th>BEAST</th>
<th>Lucky 13</th>
<th>RC4 Biases</th>
<th>CRIME</th>
<th>BREACH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLIENT SIDE</strong></td>
<td>Upgrade browsers</td>
<td></td>
<td></td>
<td>Upgrade browsers with no TLS compression support</td>
<td></td>
<td>Upgrade browsers with X-Frame-Options</td>
</tr>
<tr>
<td><strong>SERVER SIDE</strong></td>
<td>Use RC4</td>
<td>Use RC4</td>
<td>Do not use RC4</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
<td>Restrict reflection of user input</td>
</tr>
<tr>
<td></td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Separate secrets from user input</td>
<td>Random timing delay</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of counter measures

<table>
<thead>
<tr>
<th>CLIENT SIDE</th>
<th>BEAST</th>
<th>Lucky 13</th>
<th>RC4 Biases</th>
<th>CRIME</th>
<th>BREACH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrade browsers</td>
<td>Upgrade browsers</td>
<td>Throttle client initiated renegotiations</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
<td>Restrict reflection of user input</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SERVER SIDE</th>
<th>BEAST</th>
<th>Lucky 13</th>
<th>RC4 Biases</th>
<th>CRIME</th>
<th>BREACH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use RC4</td>
<td>Use RC4</td>
<td>Do not use RC4</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
<td>Restrict reflection of user input</td>
<td></td>
</tr>
<tr>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Separate secrets from user input</td>
<td>Random timing delay</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Random timing delays**

**RC4 Biases**

**CRIME**

**BREACH**

**TIME**

**CLIENT SIDE**

**SERVER SIDE**
## Comparison of counter measures

<table>
<thead>
<tr>
<th></th>
<th>BEAST</th>
<th>Lucky 13</th>
<th>RC4 Biases</th>
<th>CRIME</th>
<th>BREACH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLIENT SIDE</strong></td>
<td>Upgrade browsers</td>
<td></td>
<td></td>
<td>Upgrade browsers with no TLS compression support</td>
<td>Upgrade browsers with X-Frame-Options</td>
<td></td>
</tr>
<tr>
<td><strong>SERVER SIDE</strong></td>
<td></td>
<td></td>
<td>Throttle client initiated renegotiations</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
<td>Restrict reflection of user input</td>
</tr>
<tr>
<td></td>
<td>Use RC4</td>
<td>Use RC4</td>
<td>Do not use RC4</td>
<td>anti-automation techniques</td>
<td>anti-automation techniques</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Comparison of counter measures

<table>
<thead>
<tr>
<th>CLIENT SIDE</th>
<th>BEAST</th>
<th>Lucky 13</th>
<th>RC4 Biases</th>
<th>CRIME</th>
<th>BREACH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrade browsers</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade browsers with no TLS compression support</td>
<td>Throttle client initiated re-negotiations</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
</tr>
<tr>
<td>SERVER SIDE</td>
<td>Use RC4</td>
<td>Use RC4</td>
<td>Do not use RC4</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade browsers with X-Frame-Options</td>
</tr>
<tr>
<td></td>
<td>anti-automation techniques</td>
<td>anti-automation techniques</td>
<td>Restrict reflection of user input</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Comparison of counter measures

<table>
<thead>
<tr>
<th></th>
<th>BEAST</th>
<th>Lucky 13</th>
<th>RC₄ Biases</th>
<th>CRIME</th>
<th>BREACH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLIENT SIDE</strong></td>
<td>Upgrade browsers</td>
<td></td>
<td></td>
<td>Upgrade browsers with no TLS compression support</td>
<td>Upgrade browsers with X-Frame-Options</td>
<td></td>
</tr>
<tr>
<td><strong>SERVER SIDE</strong></td>
<td>Use RC₄</td>
<td>Use RC₄</td>
<td>Do not use RC₄</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
<td>Restrict reflection of user input</td>
</tr>
<tr>
<td></td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>anti-automation techniques</td>
<td>anti-automation techniques</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BEAST</td>
<td>Lucky 13</td>
<td>RC4 Biases</td>
<td>CRIME</td>
<td>BREACH</td>
<td>TIME</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------------------------------</td>
<td>------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><strong>CLIENT SIDE</strong></td>
<td>Upgrade browsers</td>
<td></td>
<td></td>
<td>Upgrade browsers with no TLS compression support</td>
<td>Upgrade browsers with X-Frame-Options</td>
<td></td>
</tr>
<tr>
<td><strong>SERVER SIDE</strong></td>
<td>Use RC4</td>
<td>Use RC4</td>
<td>Do not use RC4</td>
<td>Disable TLS compression</td>
<td>Mask the secret</td>
<td>Restrict reflection of user input</td>
</tr>
<tr>
<td></td>
<td>Upgrade to TLS 1.1 +</td>
<td>Upgrade to TLS 1.2</td>
<td>Upgrade to TLS 1.2</td>
<td>anti-automation techniques</td>
<td>anti-automation techniques</td>
<td></td>
</tr>
</tbody>
</table>
Thank You

• Special Thanks to Shawn, Tom, Michael, Javed, Jonathan, Tim, Josh, Alban, Ryan, Aaron and everybody in iSEC

• Whitepaper: Attacks on SSL (bit.ly/1cAqL70)

• Pratik Guha Sarkar
  • Security Consultant at iSEC Partners
  • psarkar@isecpartners.com | @pragusa55

• Shawn Fitzgerald
  • Principal Security Consultant at iSEC Partners
  • shawn@isecpartners.com