Some Real Timing Attacks

Presented by Matt McCutchen
Stealing SSL server private keys
(Brumley & Boneh)

- During SSL handshake with RSA:
  - Client chooses $M$, sends $C = M^e \text{ mod } n$
  - Server calculates $M = C^d \text{ mod } n$
  - Session key derived from $M$
- Attacker can send arbitrary $C$ and measure response time
  - Handshake will fail; don't care
- Goal: Determine $q$ (and thereby private key) from response times in OpenSSL $< 0.9.7b$
Decryption

- $M = C^d \mod n$
- Or in 1/4 the time:
  - $M_q = (C \mod q)^{(d \mod (q-1))} \mod q$
    - Binary square and multiply: includes multiplications by $(C \mod q)$
  - $M_p = (C \mod p)^{(d \mod (p-1))} \mod p$
- Recover $M$ via Chinese Remainder Theorem
Montgomery reduction

- If $R > q$ is a power of two, then $a \odot b = abR^{-1} \mod q$ is faster to compute than $ab \mod q$
  - For random $a$, $a \odot b$ does an extra subtraction w/ probability $\approx b/2R$
- $(\mathbb{Z}_q, +, \cdot) \cong (\mathbb{Z}_q, +, \odot)$ via multiplication by $R$
- Work under isomorphism: multiplications $x \odot (CR \mod q)$
  - Extra subtractions depend on $(CR \mod q)$

![Graph showing extra subtraction time vs. \( CR \mod n: q, 2q \)]
Multiplication methods

- To multiply numbers \( r, s \) words long:
  - \( r = s \): Karatsuba algorithm, \( r^{1.59} \) word multiplications
  - \( r \neq s \): Long multiplication, \( rs \) word multiplications
- \( x \odot (CR \mod q) \): \( x \) is almost always full length but length of \( (CR \mod q) \) varies

Assuming 32-bit words and 512-bit primes (Not to scale!)
The resulting timing channel

- Function has this general shape, but:
  - Exact vertical offsets are highly sensitive to compiler optimizations, small code changes
  - Noisy on the level of individual \((CR \mod n)\) values
  - Trials on a real server are noisy
Searching for $q$

- Can probe $(CR \mod n) = x$ by sending $C = xR^{-1}$
- Want to find the discontinuity at $q$
  - Direction of jump depends on how close in we are
Search procedure

- Guess one bit at a time (bisection)
- Guessing $q_{k+1}$:
  - Is $q$ between $g = q_1...q_k00...0$ and $g_{hi} = q_1...q_k10...0$?
  - Probe values $\{g, g+1, ..., g+m\}$ $s$ times each and take average(?) , likewise for $g_{hi}$
    - Used $m = 400$, $s = 7$
  - If $|T_{\text{avg}}(g) - T_{\text{avg}}(g_{hi})|$ is:
    - "large" $\Rightarrow$ $q$ is between $g$ and $g_{hi}$; $q_{k+1} = 0$
    - "small" $\Rightarrow$ $q$ is not between $g$ and $g_{hi}$; $q_{k+1} = 1$
    - unclear: try larger $m
Results

- Stole a 1024-bit key over a campus network with \(\sim 10^6\) queries (2 hours)

Only 0-bits of \(q\) are shown. For 1-bits, \(T_{\text{avg}}(g) - T_{\text{avg}}(g_{hi}) \approx 0\).

The graph illustrates the size of \(|T_{\text{avg}}(g) - T_{\text{avg}}(g_{hi})|\) that would indicate a 0-bit.
Defense

- RSA blinding
  - Choose random $a$
  - $C' = a^e C \mod n$
  - $M' = (C')^d \mod n$
  - $M = a^{-1} M' \mod n$
  - Attacker does not know/control $(C'R \mod n)$ so timings are meaningless

- Rewrite code so timing is less input-dependent: not as foolproof
Timing web applications
(Bortz, Boneh, Nandy)

- Response time is dependent on the amount of work the app has to do
Direct timing

- Make a request and measure response time
  - With chunked encoding, can also measure chunk arrival times
- May leak existence or quantity of objects not visible to attacker
Direct timing: examples

- Is this username valid?
  - Try to log in with a bogus password
  - Valid usernames take longer before "bad login" is returned
Direct timing: examples

- How many private photo albums exist on the site?
  - Visit the list of albums
  - Takes time to filter the list
Cross-site timing

- User visits attack page that embeds application page with `<img>` / `<script src>`
- Embedding fails (wrong file type) but response time is reported to attacker
- Take advantage of user's session
- Must compare to a baseline to correct for unpredictable network conditions

```javascript
t1 = time();
<script src="http://app/..."/>
send(time() - t1);
```
Cross-site timing: examples

- Is the user logged in?
  - Embed the home page
  - Takes longer to incorporate user-specific data
Cross-site timing: examples

- How many items are in the shopping cart?
  - Embed the shopping cart page
  - Takes time to render the items
Cross-site request forgery + timing

- Traditional XSRF allows writing (only)
- Writing can facilitate reading
  - Ex: Does the shopping cart contain item X?
    - Measure number of distinct items in cart
    - "Add to cart" X
    - Measure number of distinct items in cart again
    - Remove X / reset quantity to 1 to avoid detection
- Reading can facilitate writing (Matt)
  - Discover parameters for a write
  - Check whether a write succeeded (make multi-step attacks more robust)
Defenses

- Quantize response time (or all chunk times)
  - Hurts perceived performance, not very satisfactory
- Cross-site: Server restricts what sites can embed session-dependent resources (require whitelisted Referer or secret query parameter)
  - For mashups, it is possible to allow iframes based on Accept and prevent them from being timed (Matt)