Side Channels

Scribe: Austin Murdock

November 11, 2015

Side information - Information a physical system leaks outside of input and output

1 Examples

There are several examples of side information channels outside of cryptography:

- Typing patterns of passwords
- Timing attacks on SSH
- Timing attacks on caches
- Mobile sensors
- Packet sizes
- Paged out pages in SGX

2 Types of Side Channel Attacks

2.1 Timing Attacks

Introduced by Kocher in 1995, extracted key by RSA signature

```
$Pk : N, e \quad Sk : d$
$Sig(M) = M^d \ mod \ N$
Repeated Squaring Protocol
$Signature = 1$
$P = M(M^2, M^{2^2}, M^{2^3} ...)$
given a binary number $d$ where
$d = d_0 ... d_{L-1}$
$for \ i = 0, \ L-1$
  $if \ d_i = 1$
    $Sig = \{Sig * P\} \ [mod \ N]$
   {small or large}  [ $mod(v)$ :
                        $if \ V < N$
                          ...
                        $else$
                          $V \ mod \ N$
                        $return \ V$ ]
  $P = P * P \ mod \ N$
```

```
$d_0 = 1$
  $Sig = 1 * M \ mod \ N$
  $// \ if(M < N)\{no \ time\} \ else \ \{longer\}$
  $P = M * M \ mod \ N$
$d_0 = 0$
  $P = M * M \ mod \ N$
$M$ small $d_0 = 1$ fast | signal
$M$ large $d_0 = 1$ slow | signal
Repeating, identify $d_0$
```

Fixed by Rivest

$Sig(M) = [M^d] \ mod \ N$

$[\ (y^e)(M * y^e)^d = yM^d \ ]$

2.2 Power Analysis

Every operation on a machine has a unique power signature. A secret can be determined by examining the sequence of operations corresponding to the power signature.

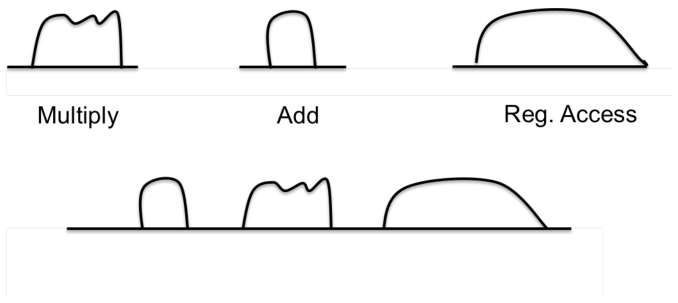

Figure 1: Power signature corresponding to a hypothetical operation

2.3 Cache Attacks

The timing of lookup operations in AES depend on cache hits or misses

$$T\ [index] \rightarrow$$

2.4 Acoustic Analysis

Although the first attacks were made in the 1950s, modern analysis was introduced in 2005 and proven by Genkin, Shamir, and Tromer in 2013. Traditionally acoustic analysis was applied to computer keyboards, printers, and electromechanical cipher machines. Recently an attack on GnuPG was demonstrated using analysis of ultrasonic noise originating from capacitors and inductors on the motherboard.

3 Remote Timing Attacks Against OpenSSL

3.1 Attack

Remote Timing Attacks Against OpenSSL describes a side channel attack by Brumley and Boneh that uses time measurements to determine an RSA private key. They send several client key exchange messages with a guess g of the secret key. This message will result in an error on the endpoint after decryption and it will respond with an alert message. Their attack computes the time difference from sending g to receiving the alert response as the time to decrypt g. They determine the critical components of the secret key via a binary search for q where N = pq in the RSA specification. They approach the actual key over time by refining their guess according to the time delay of the endpoint's response.

3.2 Timing Factors

There are two main factors in the OpenSSL protocol that affect server response delay: use of Montgomery Reduction and choice of multiplication routine.

3.2.1 Montgomery Reduction

Montgomery Reduction transforms a reduction modulo q into a reduction modulo to some power of two denoted by R. At the end of a reduction, the algorithm checks to see if the output cR is greater than q. If true, the algorithm subtracts q from the output, resulting in an extra step and thus a difference in runtime.

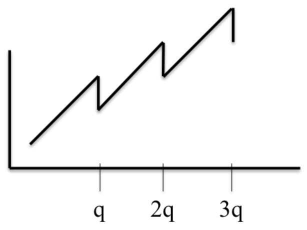

Figure 2: Extra reductions as a function of the input g

3.2.2 Multiplication Routines

OpenSSL uses two multiplication routines, Karatsuba and normal. OpenSSL uses Karatsuba multiplication, which runs in $O(n^{\log_2 3})$ when multiplying two number with an equal number of words. OpenSSL uses normal multiplication, which runs in $O(nm)$ when multiplying two numbers with an unequal number of words of size n and m. The choice of multiplication routine has the opposite effect on the response time from Montgomery Reduction. The attack by Brumley and Boneh uses both the effects of Montgomery Reduction and choice of multiplication routine in their attack, with each effect having a dominant impact at different stages.

3.3 Defense

The preferred method for protecting against timing attacks is to use RSA blinding. The RSA blinding operation calculates $x = r^e g \mod N$ before decryption, where r is random.

4 Exploring Information Leakage in Third-Party Compute Clouds

Exploring Information Leakage in Third-Party Compute Clouds presents techniques for using the cloud computing environment to expand and exploit the attack surface of a victim running software on a web services platform. In the paper the authors present an attack on Amazon's Elastic Compute Cloud (EC2) to achieve information leakage. First the attacker must achieve co-residency on the physical hardware being used to host the victim's instance. The attacker can use several techniques to achieve co-residency:

- Create instances using an IP address with a location similar to that of the victim
- Request instances at the same time as the victim
- Start a large number of instances and check to see if one is a co-resident by matching the Dom0 IP address

Once the attacker has achieved co-residency, they can induce information leakage by establishing several covert channels. For example,

- Use a prime, trigger, probe technique to infer victim's VM memory activity from shared cache delays
- Perform thousands of cache loads and measure the difference in expected time and actual time, the difference in time represents the estimated traffic activity of the victim
- Use cache-based load times to estimate the time between victim keystrokes

5 Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow

Analysis of web applications demonstrates that some of their fundamental features are the root cause of side-channel vulnerabilities.

- Frequent small communications
- Diversity of contents exchanged in state transitions

- Stateful communications

There are several abstract strategies for preventing side channels in web applications, such as: padding packets, faking packets, chopping packets into fixed-size segments, and merging or splitting application states. However, to effectively and efficiently prevent side channels web developers need to create application specific solutions. Development of these solutions require the analysis of web application semantics, information flow, and network traffic patterns.