Tales from the Crypt(analysis)

A Survey of Side-Channel Attakcs

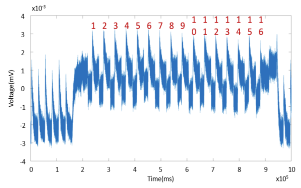
Charles Averill

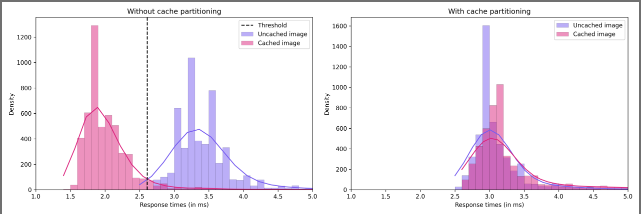
Computer Security Group
The University of Texas at Dallas

April 9, 2025

What is a Side-Channel Attack?

- A Side-Channel Attack is an exploit utilizing non-algorithmic, usually physical flaws in the implementation of code
- SCAs are most often used in a read-only fashion - they extract sensitive information that is not exposed through the code itself





Why should I care?

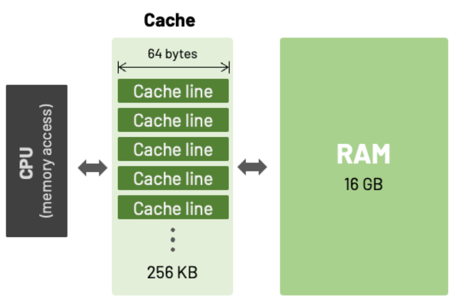
- In short: everybody is vulnerable all the time in many ways, even **you**
- They are usually after cryptographic secrets, like your passwords and private keys
- Because SCAs are primarily used only to listen, they are incredibly difficult to detect
- Mitigation is also very difficult for some classes of attack, but is straightforward for some common attacks
- Many attacks have been shown to be possible in the wild, but very few have ever been caught!

What kinds of attacks are there?

- Timing - code runtime leaks sensitive information
- Cache - leaking sensitive information by timing memory accesses - shorter access time if some data is cached
- Fault injection - apply physical stress to the system to induce a fault that leaks information or gives control
- Power analysis - the amount of power going into or out of a system leaks information, sometimes through other side channels (acoustic, optical)

Cache Attacks Primer

- *Cache* is fast memory used to store frequently-used data
- Memory addresses map to *cache lines/blocks* that hold their data
- Usually a subset of timing attacks, but these are so distinct and successful they deserve their own category



Meltdown

Goal

Read memory the attacker **can't legally access** (e.g. kernel space)

- Construct `arr` of size $256 \cdot 4096$ - cache line for each byte value
- Run **ATTACKER CODE**
- Iterate over `arr` and access each value at index i
- Time of accesses indicate the value of `SECRET`!
- Can be done in quick succession to read memory illegally

USER SPACE
`0x0 - 0x100000: arr`

KERNEL SPACE
`0xff00: SECRET`

ATTACKER CODE

```
0: lw r0, 0xff00     // will fault
4: mul r1, r1, 4096 // get offset into arr
8: lw r1, 0x0(r0)    // caches arr[SECRET]
```

Spectre

- Similar to meltdown, but harder to defend against
- Run **ATTACKER CODE** several times with $i < 256$ - this *mistrains* the branch predictor to choose the "true" branch more often
- Run **ATTACKER CODE** such that `&data + i` indexes into `SECRET`
- Do access time analysis on `arr` to again reveal contents of memory
- Generally more capable than meltdown - its variants can do illegal memory accesses in a broader range of situations
- **Foreshadow** - can break VM separation, break into secure enclaves

ATTACKER SPACE

```
0x0 - 0x100000: arr
0x10000-0x100100: data
```

VICTIM SPACE

```
0xff00 - 0xffff: SECRET
```

ATTACKER/SHARED CODE

```
if (i < 256) {
  value = data[i]; // OOB i can read SECRET
  x = arr[value * 4096];
}
```

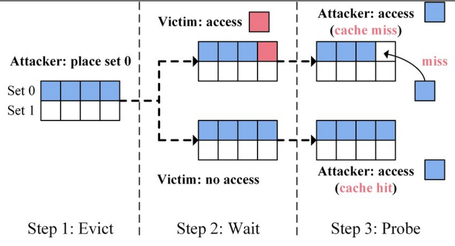
Prime + Probe

Goal

Determine which cache lines, memory regions a VM neighbor is using

1. Fill up a cache line with your own data
2. Let victim run their code in another VM (same machine)
3. Time memory accesses for your data - if slow, victim overwrote your cache line because they accessed memory in a similar range to you

Can be used to accelerate Meltdown/Spectre-style attacks



SSH Username Enumeration

Goal

Determine whether a user has an account on a remote machine

- Construct a malformed login packet for a given username
- Old versions of OpenSSL will take longer to reject the malformed packet for existing users than non-existent users
- `authctxt->valid` checks for user validity/existence

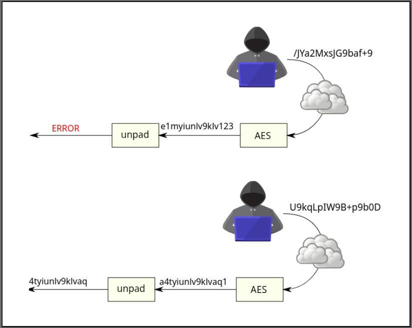
```c
static int
userauth_pubkey(struct ssh *ssh)
{
...
        if (!authctxt->valid) {
                debug2("%s: disabled because of invalid user", __func__);
                return 0;
        }
        if ((r = sshpkt_get_u8(ssh, &have_sig)) != 0 ||
            (r = sshpkt_get_cstring(ssh, &pkalg, NULL)) != 0 ||
            (r = sshpkt_get_string(ssh, &pkblob, &blen)) != 0)
                fatal("%s: parse request failed: %s", __func__, ssh_err(r));
```

Lucky Thirteen

Goal

Recover plaintext from arbitrary encrypted TLS packets

- Old TLS uses CBC mode to encrypt packets with block ciphers
- Attacker tweaks padding bytes in captured ciphertext, send the tweaked packet back to the server
- Packet rejection time depends on whether padding bytes are valid - so recovering padding length is easy
- Once padding length is found, plaintext can be recovered by again tweaking bytes in ciphertext and measuring server response times



Diffie-Hellman, RSA, DES Timing Attacks

Goal

Extract cryptographic keys by observing differences in decryption time

- Modular exponentiation ops in RSA, DH, and DES vary in runtime depending on key
- Repeatedly listen to decryption, each time gaining a bit of the key
- Works over a network if jitter is low enough
- Experimentally shown to be 85% successful

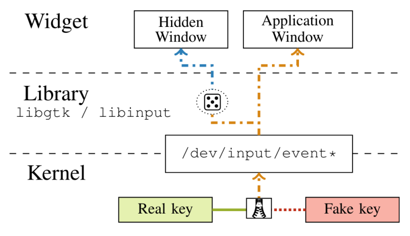
```
def modexp(y, x, n, w):
  s = 1  # Init accumulator
  # Loop over bits of x
  for k in range(w):
    if (x >> k) & 1:
      # Timing leak: slower if bit is 1
      R = (s * y) % n
    else:
      # Faster if bit is 0
      R = s
    s = (R * R) % n
  return R  # = y^x mod n
```

Keystroke Timing Attacks

Goal

Capture user input by monitoring keyboard interrupt times

- Large class of attacks because there are many channels to exploit - /proc, Chrome events, pure Javascript, cache/DRAM timing
- Different keystrokes take different amounts of time (μs) to process - more samples $\rightarrow$ higher chance to reconstruct original input
- Mitigations like Keydrown involve the OS flooding the system with fake keystrokes only it can distinguish



Rowhammer

Goal

Induce errors in arbitrary processes by modifying their memory

- Memory cells in DRAM are really small, less than 100nm
- Electrical noise from one cell can induce a fault in another
- Loading from a cell can disrupt the value of its neighbors
- Can cause Linux privelege escalation, DMA, Android roots
- Exploitable via Javascript!

```
hammer:
  mov (X), %eax   // read from address X
  mov (Y), %ebx   // read from address Y
  clflush (X)     // flush cache for address X
  clflush (Y)     // flush cache for address Y
  jmp hammer
```

Bypassing Smart Card Tamper Resistance

Goal

Circumvent tamper resistance to extract keys or break functionality

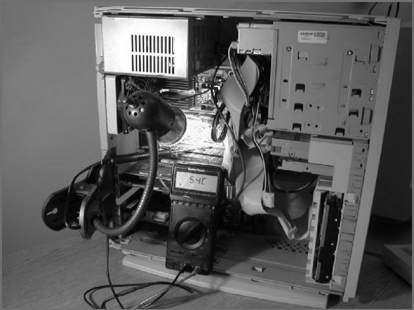
- Smart Cards (bank, ID, transit, etc.) contain small microprocessors
- Essentially zero demand from customers for tamper-hardened chips
- Trivial to remove chips from the cards, allows for reverse-engineering sensitive information and altering behavior
- Fault can be injected via UV light, a knife, acid, an electrical connection, etc.

Java Type-Safety Bypass via Lightbulb

Goal

Cause type-safety violations in Java by inducing bit flips in memory

- Researchers shined an off-the-shelf lamp on memory chips
- Light and heat exposure caused single-event upsets (bit flips)
- Causes illegal type casts, e.g., making an integer look like a pointer
- Allows attacker to take complete control of JVM



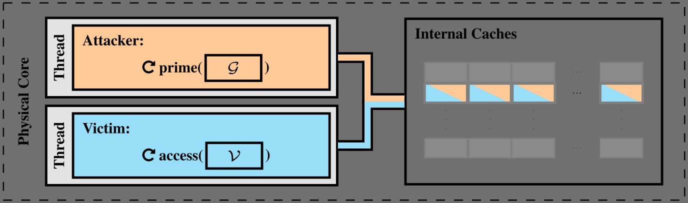
Cold Boot Attacks on Encryption Keys

- Incorrect assumption: secrets in memory are wiped when the machine turns off due to DRAM volatility
- An attacker with physical access can reboot the machine into a controlled OS and dump the contents of DRAM
- Secrets (such as cryptographic keys) can be extracted from the memory dump
- More theoretical than other attacks in this presentation - limited applicability

Figure 5: Before powering off the computer, we spray an upside-down canister of multipurpose duster directly onto the memory chips, cooling them to $-50\,^{\circ}\mathrm{C}$. At this temperature, the data will persist for several minutes after power loss with minimal error, even if we remove the DIMM from the computer.

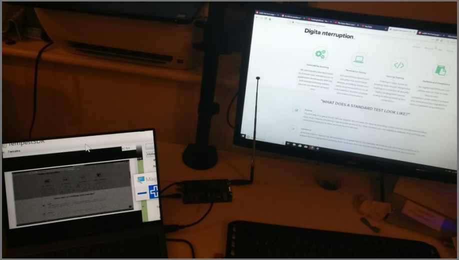
Collide + Power

- CPU/memory contains both sensitive user data and attacker data at the same time
- If attacker knows how much power their data uses, subtract it from the total to get user data
- Similar to meltdown: fill up cache, victim overwrites it - this results in power utilization closely related to the value of the written data
- Practical and requires no special privileges on modern systems



TEMPEST

- NSA project on hardening sensitive systems against spying via EMF
- Declassified documents specify several ways in which adversaries can spy on a number of mechanical devices
- Various kinds of screens emanate their contents far enough that antennae can read them
- **Distance**: physical isolation of sensitive network from unsecured network
- **Shielding**: implement physical barriers that block EMF



Common Threads

- Attackers are smarter and craftier than you
- Side-channels pervade every facet of computing
- These attacks are difficult to detect and often difficult to mitigate, but must be considered in the implementation of secure systems

Resources

- Meltdown/Spectre Writeups
- SSH User Enumeration Demo
- Lucky Thirteen: Breaking the TLS and DTLS Record Protocols
- Timing attacks on RSA, Diffie-Hellman, DES, etc.
- Keydrown
- TEMPEST - A Signal Problem
- Tamper Resistance - a Cautionary Note
- Using Memory Errors to Attack a Virtual Machine
- Lest We Remember: Cold Boot Attacks on Encryption Keys