Dealers Choice Diversion: Quantum Computing & Side Channels

Pre Lecture Facepalm...

Home / Cisco

Cisco

Cisc

MENU

US EN

Products & Services / Security /

Cisco Firepower Management Center

Hi

Centralize, integrate, and simplify management

This is your administrative nerve center for managing critical Cisco network security solutions. It provides complete and unified management over firewalls, application control, intrusion prevention, URL filtering, and advanced malware protection. Easily go from managing a firewall to controlling applications to investigating and remediating malware outbreaks.

Watch 3-minute overview

Watch demo now

Summ

A vulnera
authentic
the *root* u

The vulne
could exp

These vulnerabilities exist due to improper input validation. An attacker could exploit these

allow an attacker to execute arbitrary cor

vulnerabilities by sending crafted SQL queries to an affected device. A successful exploit could allow

Why This Digression...

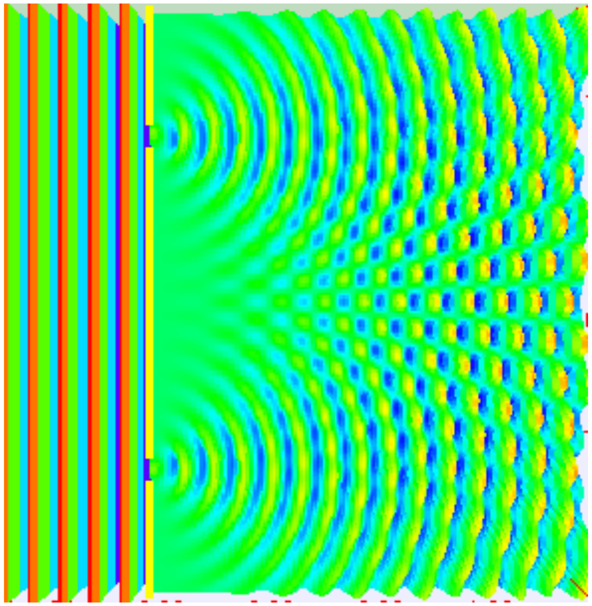
- It actually is remarkably important topics...
  - Well, side channels are. Quantum is why you can just chill (for now)
- We have space for digression lectures in the syllabus
  - So lets do one
- I'm out of town next week:
  - Raluca Popa will be covering Wednesday and Friday...
    And I want to leave her plenty of web-security stuff to talk about

Quantum Mechanics: The Weird Reality...

- At the scale of individual atoms, our intuition breaks down...
  - Things behave like both particles and waves
  - Things can pass through other things
  - Things can be in multiple states at once
  - Probabilities matter
- I don't think anyone really intuitively ***understands*** Quantum...
  - But it works...
- Disclaimer: I'm a failure at Quantum:
  - I got a C (I deserved an F) in Physics 137A, this is truly weird stuff!

Example Weirdness:
The Double Slit Experiment

- If you beam a light at a set of double slits
  - You get a wave diffraction pattern 🙂
- If you bean a beam of electrons...
  - You get a wave diffraction pattern?! 🤔
- But light is composed of "photons" and electrons are clearly particles
  - If you send them one at a time, each one arrives at single points, but…
  - Taken together you get a diffraction pattern 🤷‍♂️
- But if you ***measure*** which slit each particle went through...
  - You eliminate the diffraction pattern!
  - Single electrons and photon "particles" are interfering ***with themselves*** like a wave does! 🤨



So What Does This Mean?

- Things are both particles and waves?!?
- Things can be in two places at once?
- When you measure something, it behaves differently?
- EG, Schrodinger's cat...
  - A thought problem: You have a cat in a sealed box, a vial of poison, and a single radioactive atom...
    - At time T, there is a 50% chance the atom decayed, broke the poison, and killed the cat
  - Is the cat alive? Dead? Both?
    - "Yes", until you open the box!

Another Weirdness:
EPR entanglement

- Einstein ***hated*** quantum mechanics...
  - "God does not play dice with the universe"
  - Plus his genius idea, relativity, doesn't actually work with quantum...
    - If you can unite general relativity and quantum mechanics, you are getting a flight to Sweden to pick up your Nobel prize
- Einstein–Podolsky–Rosen came up with a "paradox"...
  - The "EPR pair"
  - Intended to go "See, this Quantum 💩 makes no sense..."
  - The problem is, it actually ***works!***

EPR "Paradox" in action

- We have two particles, A and B...
  - A is in an unknown state, 50% of the time A = 0, 50% of the time A = 1
    - Really, A is in a superposition of both states:
      The cat is alive and dead
  - If we measure A, we have a 50/50 chance at the time of measurement
  - But until we measure A, it continues to exist as probabilistic superposition of both states
- We then "entangle" B without measuring A
  - So that A=0 <-> B=0 and A=1 <-> B=1
  - And then separate the two, perhaps even by light years distance!
- Now when we measure
  - If A = 0 we will ALWAYS see B =0…
    - But if A = 1 we will see B = 1
- And it doesn't matter which way we order our observations
  - and it is still random which one it is?!?

As long as we maintain coherence...

- We can keep this up!
  - So lets take several bits, B_0, B_1, B_2
  - Put each one in an independent 50/50 state. These are now qbits (Quantum Bits)
- Now we do a computation:
  - $B_3 = B_0 \oplus B_1 \oplus B_2$
  - But while maintaining coherence
- Now the spooky thing...
  - We've really computed all quantum superposition of all possible values of B_3 as a function of B_0-B_2...
    - In hardware language it is like we computed the ***entire*** truth table in one go and things are existing in that superposition
- But if we ***measure*** them, we get just a single input/output pair

And Now The Quantum Miracle...

- So far, this is no more powerful than a conventional computer
  - After all, we still only get a single output for a single set of inputs...
- But then we get to the Quantum magic...
- We now take B_0-B_3 and pass them through another transformation
  - That basically self-interferes between the superposition of all the input/output pairs
- And now when we look...
  - We see some information about the ***relationship*** between all the bits!

So What Good Is This?

- Shor developed an algorithm to solve two different & related group theory problems
  - Find the order of a group
    - Given a group ***G***, a generator ***g***, how many elements are in the group?
    - You can reduce factoring to this problem
  - Find the discrete log
    - Given a group ***G*** of known order, a generator ***g***, and a value g^x mod ***G***, what is ***x***?
- The number of quantum bits (qbits) required:
  - $O((\log N)^2 (\log \log N) (\log \log \log N))$ with ***N*** the number to be factored
  - So still a lot of quantum state: millions of qbits for a 2048b RSA key
- Oh, and this is just about the only thing it is good for

This Breaks All Major Public Key

- Diffie/Hellman: Break discrete log
- RSA: Break factoring
- Elliptic Curve
  - It's more complicated because you don't know the order of the group...
  - Well, its not actually. See the footnote on the "factoring" algorithm!
    - You use the RSA algorithm to get the order of the group
    - And then use the discrete log problem
- But what does this actually mean?

Implications #1:
Is ECC better?

- In conventional computing: ECC is the same strength with fewer bits
  - 256b ECC ~= 2048b RSA & DH
    - There are sub-exponential shortcuts for the group-theory problems in the integers not present on elliptic curves
- But this isn't the case with quantum computing!
  - So if we could only build a "medium-sized-ish" quantum computer (tens of thousands rather than millions of qbits), ECC breaks first
- Speculation: Is this why in going from Suite B to CNSA, the NSA said...
  - "Whoah, hold off on going to ECC until we have post-Quantum public key...
    and until then you can use 3096b RSA and DH as well"

Implication #2:
Lots of work on "Post-Quantum Public Key"

- A major area of active research: public key algorithms without a quantum shortcut
  - Significantly larger keys: 400B (same as 3096b RSA) to 10,000B depending on the algorithm
- In practice, never used alone!
- EG, the "NewHope" TLS handshake experiment
  - Does both an ECDHE and post-quantum public key agreement:
    Both would have to be broken to break the system

Implication #3:
Don't Worry...

- There may be exponential or near exponential difficulties in maintaining coherence as a function of the # of qbits
  - Open question: There is a lot of work on this, but 🤷‍♂️.
  - I've heard "Quantum Computers Real Soon Now" for nearly 25 years!
- The current "Quantum" computers really aren't
  - D-Wave is actually "quantum annealing":
    2-D simulated annealing with Quantum acceleration. Open question whether it is fundamentally faster
  - Google's "Quantum Supremacy":
    Better than a classical computer at computing how it will compute?!?
    Again, only 2D not generic operations
- True generic quantum computers have been built...
  Capable of factoring "15"

Side Channels & Other Hardware Attacks: Worry

- A side channel attack requires measuring some other piece of information
  - EG, time, cache state, power consumption, etc...
- And using it to deduce a secret about the system
- Side channels are very, ***very*** powerful

Requirements

- Often the biggest limitation is attacker requirements
- Timing attack
  - Need to measure the timing of the operation with potentially very high precision
- Power attack
  - Need physical access to the device:
    Generally only applicable to smart-cards and similar devices
- EMF ("Tempest")
  - Need close physical access
- Processor side-channel attacks
  - Need to co-locate the attacker code:
    EG, cloud computing, web browsers, etc

Example Timing Attack: Keystrokes...

- User is inputting a password
  - And the user is using a Bluetooth keyboard...
  - Or the user is using a remote connection over ssh
- Someone nearby can observe when keys are pressed
  - They are sent immediately
  - But not ***what*** keys are pressed
- Can this leak sensitive information? Of course!

Timing Leakage

- Some keys are faster to press
- Can use this to model timing
  - Either generically or specific to the user
- Lots of ways to do this
  - Hidden markov models
  - Throw machine learning at it...
- Really really hard to hide
  - Can't delay interactive requests without adding latency
  - "Cover traffic" only adds additional data, can't remove the underlying signal
- From https://people.eecs.berkeley.edu/~daw/papers/ssh-use01.pdf

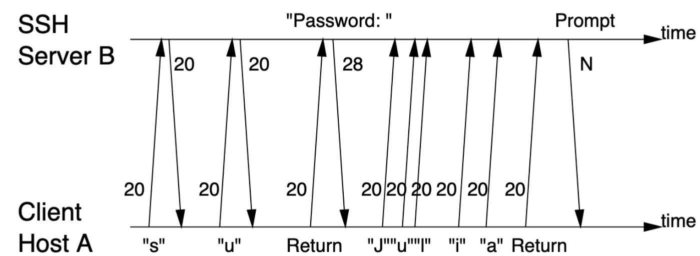


Figure 1: The traffic signature associated with running SU in a SSH session. The numbers in the figure are the size (in bytes) of the corresponding packet payloads.

Timing Attacks & Cryptography

- The classic timing attack:
  - Compute ***y^x^ mod n***
- Easy solution ends up being

```
Let s_0 = 1.
For k = 0 upto w − 1:
  If (bit k of x) is 1 then
    Let R_k = (s_k · y) mod n.
  Else
    Let R_k = s_k.
  Let s_{k+1} = R_k^2 mod n.
EndFor.
Return (R_{w−1}).
```

- https://www.paulkocher.com/TimingAttacks.pdf

Implications: Public Key Operations Need "Constant Time"

- Optimizing cryptographic code can be dangerous...
  - Instead it needs to take the same amount of time no matter what the input is
  - Even compiler optimizations can be a problem
- First identified 20 years ago...
  - So you think we'd have solved it...
    But you'd be wrong

Reminder DSA/ECDSA Brittleness...

- DSA algorithm
  - Global parameters: primes ***p*** and ***q***, generator ***g***
  - Message ***m***, private key ***x***, public key $y=g^x \bmod p$
  - Sign: select random ***k*** from 1 to ***q***-1
    $r = (g^k \bmod p) \bmod q$ (retry if r = 0)
    $s = (k^{-1} (H(M) + xr)) \bmod q$ (retry if s = 0)
- ***k*** needs to be random and secret and unique
  - An attacker who learns or guesses ***k*** can find ***x***
    - An attacker can even just try all possible ***k***s if the entropy of ***k*** is low
  - Even just learning a few bits of ***k***, and then having several signatures with different ***k*** for each one, and you break it!

Just ***This Week***:
The Minerva Attack

- A timing side-channel attack to get a few bits of ***k*** from the ECDSA signatures on Athena smart cards and lots of others
  - So have the smart card generate a lots of signatures
  - Then some math and brute force to get the actual ***x***
- These devices were certified… Including that they were supposed to resist timing attacks!
  - But, naturally, the certification doesn't actually test whether they are vulnerable to timing attacks...
- The root cause for many was a common code component: The Atmel Toolbox 00.03.11.05 library

Guess the Problem Here...


```
- M10.6 the TSF shall provide digital signature
confirming to EC-DSA standard.
    - Secure digital signature generate
    - Secure digital signature verify
    - Fast digital signature generate (see note*)
    - Fast digital signature verify (see note*)

- M10.7 the TSF shall provide point multiplication on an elliptical
curve, conforming to EC-DSA standard.
    - Secure multiply
    - Fast multiply (see note*)

    * The Fast functions of M10.3, M10.4, M10.5, M10.7, M10.8, M10.9, do
not offer any DPA/SPA protection and must not be used for secure data.
```

Guess the Problem Here...

- M10.6 the TSF shall provide digital signature confirming to EC-DSA standard.
    - Secure digital signature generate
    - Secure digital signature verify
    - Fast digital signature generate **(see note*)**
    - Fast digital signature verify **(see note*)**

- M10.7 the TSF shall provide point multiplication on an elliptical curve, conforming to EC-DSA standard.
    - Secure multiply
    - Fast multiply **(see note*)**

* The **Fast** functions of M10.3, M10.4, M10.5, M10.7, M10.8, M10.9, do not offer any DPA/SPA protection and **must not** be used for secure data.

Once Again: Bad API

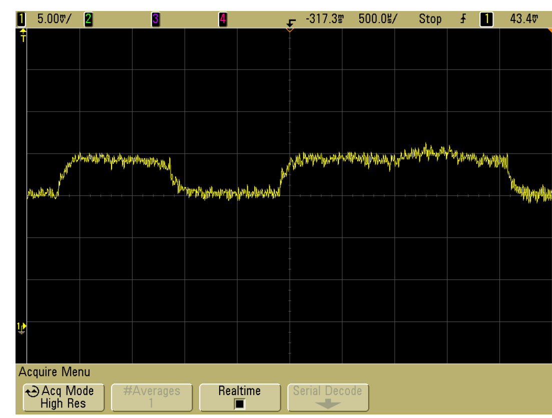
- Once again we have a case of “If you offer a programmer two ways, >50% of the time they will chose the wrong way”
  - In this case “why wouldn’t I chose the fast version?”
- You have a now growing list of “red flag/canary APIs”
  - system(), raw SQL, now this example
- Keep a growing list as a “cheat sheet”
- When you get to an existing software project…
  - Search the code for these APIs
- When you start a new project
  - ***NEVER*** use the dangerous version, even if you are using it safely…
    (EG, never use system(), only execve())

Power Attacks:
The Bane of Smart Cards...

- Smart Cards are effectively small computers
  - In a handy credit-card sized package...
- Some are used to hold secrets on behalf of the cardholder
  - So really, if the person holding the card can get the secrets, 🤷‍♂️
- Some are used to hold secrets ***from*** the cardholder
  - So if the user can extract the secrets, 🤦🏼‍♂️
- The bane: Power Analysis
  - SPA == Simple Power Analysis
  - DPA == Differential Power Analysis

The Idea...

- Different operations use different amounts of power
  - EG, square vs multiply in RSA
- Hook up smart card to a reader that can measure the power
  - Have it encrypt/sign something
  - Look at the power trace to get information about hidden secrets
    - Including statistical techniques



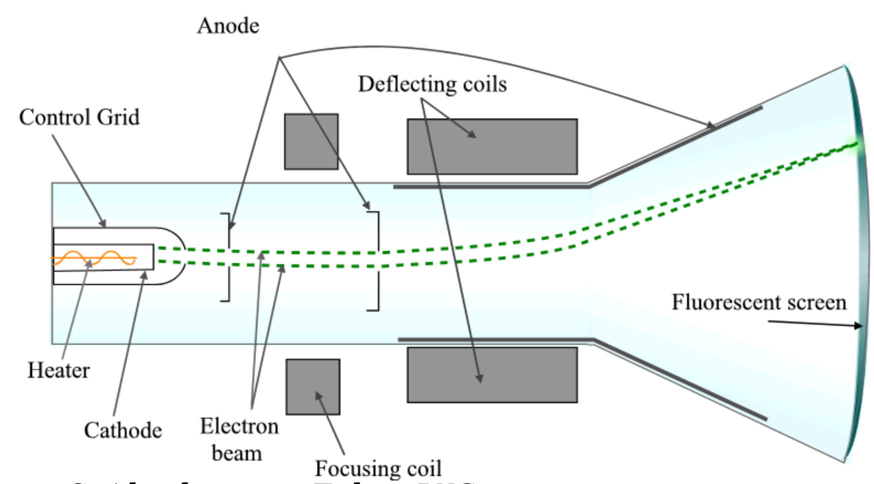
https://en.wikipedia.org/wiki/Power_analysis#/media/File:Power_attack_full.png

Countermeasures...

- Lots of work can make "simple" power analysis not work
  - But now you are using more power: Have to use the max all the time for the encryption
- Harder for more detailed differential analysis
  - Which can detect even small leaks
- If possible, punt!
  - Use your systems in a way where the person who holds the card is not your adversary!
- EG, you are building a “stored value” smart card
  - Option #1: The smart card has the value:
    If you tamper with the smart card, you can change the value
  - Option #2: The smart card just has an ID:
    You actually look up in the central database

Real Freaky: Elecromagnetic Emissions...

- Every time a circuit switches...
  - It leaks out some radio frequency energy
- Some sources are even easier
  - A old-school monitor paints the image with an electron beam on the screen...
- Which means it is a radio!
  - Transmitting an image of the screen!
- Cheap, too
  - $15 in 1984 for van Eck to read images off a monitor!



Solution: The SCIF

- The US government's paranoia: The SCIF (Sensitive Compartmented Information Facility)
  - A room (or even a whole building) specifically designed for Top Secret "stuff"
- Multiple layers of security:
  - Physical access to the building
  - No outside electronics
    - With some caveats, fit bits can be OK depending...
  - No windows
    - Beam a laser at a window and can detect vibrations!
  - Electromagnetic shielding
    - So your cellphone wouldn't work in there anyway

And Funky Hardware SideChannels...

- The recent Meltdown and Spectre Intel bugs...
  - Both were effectively side-channels
- The key idea:
  - You could trick the speculative execution engine to compute on memory that you don't own
  - And that computation will take a different amount of time depending on the memory contents
- So between the two, you could read past isolation barriers
  - Meltdown: Read operating system (and other) memory from user level
  - Spectre: Read in JavaScript from other parts of the web browser