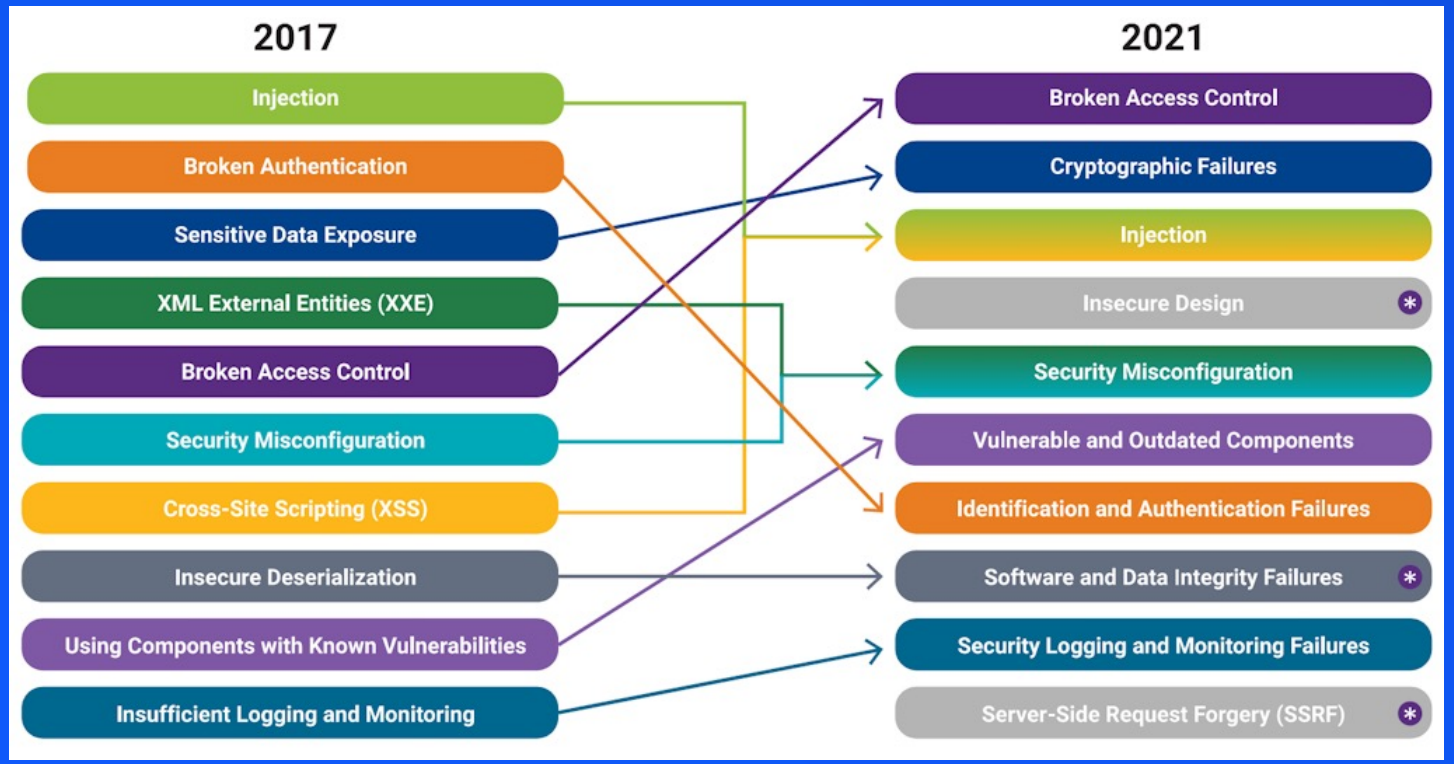
~~The Cost of Stunt Cryptography~~
The Hidden Cost of CVEs

Peter Gutmann, University of Auckland

(With help from the Bouncy Castle, cURL, cryptlib, GnuPG, OpenSSL, and Postfix projects)

What are the Threats?

In the security field we have good data on where the problems are



OWASP (Open Source Foundation for Application Security) top 10, last two revisions

What are the Threats? (ctd)

The results are remarkably stable over time

Project Information

- OWASP Top 10:2021
- Making of OWASP Top 10
- OWASP Top 10:2021 - 20th Anniversary Presentation (PPTX)

Flagship Project
Documentation
Builder
Defender

- Previous Version (2017)

A lot of the changes are just naming or classification updates

- The underlying problems remain the same

What gets all the Attention?

Consulting the OWASP top 100,000, from the Appendix to the Addendum to the Supplement to the Apocrypha, Volume 127, we see…

…

#17,245 Spectre

#17,246 POODLE

#17,247 Meltdown

#17,248 Rowhammer

#17,249 DROWN

#17,250 ROCA

….

What do all of these have in common?

What gets the Attention? (ctd)

No-one ever uses them

- There are 17,244 easier ways to carry out an attack
- This is why they've been referred to as "stunt cryptography"

Stunt cryptography attack

- You have a 0.00001% chance of recovering 2 bits of plaintext from a single message

Any of the OWASP top ten

- You have a 100% chance of recovering the plaintext of all the messages

What gets the Attention? (ctd)

People really like fancy headline-grabbing (but eminently impractical) things

- Are there any known cases of a real-life attacker ever using Spectre, Rowhammer, POODLE, or other stunt cryptography?
- (To date no-one in the audience has ever identified one)

Focusing on high-profile attacks that no-one uses draws resources away from the real problem, the actual attacks that *are* happening

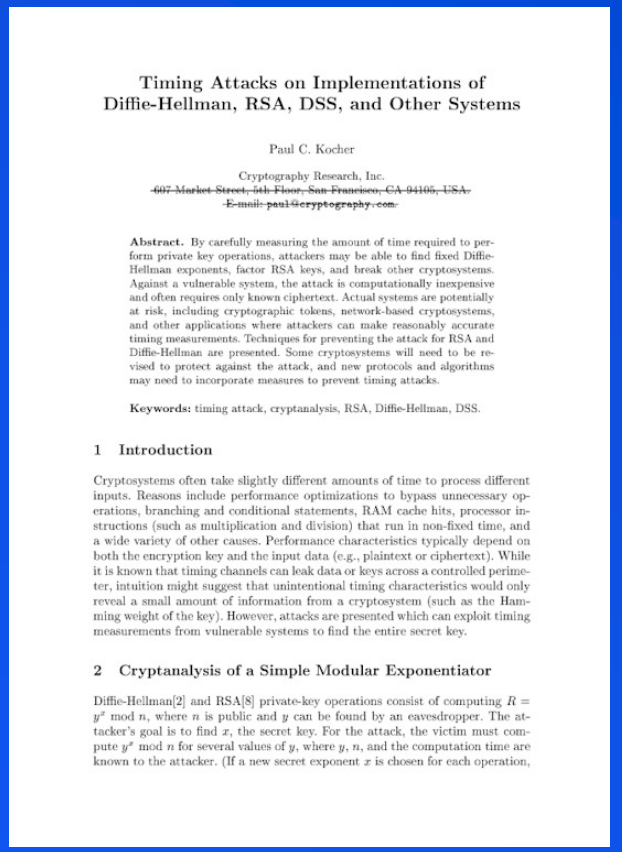
Stunt Cryptography

First side-channel attack paper was published in 1996

- "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", Paul Kocher

In the intervening 30 years…

- Tens of thousands of papers, articles, blog posts, and news headlines on the topic
- Google scholar reports a total of ~50,000 hits on variations of the term (side-channel, timing, power analysis, etc)
- There are zero known cases of a real-world attacker who didn't already have physical access to the device under attack using such an attack
  - (If you know of any, let me know)



Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher

Cryptography Research, Inc.
~~607 Market Street, 5th Floor, San Francisco, CA 94105, USA~~
~~E-mail: paul@cryptography.com~~

Abstract. By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Against a vulnerable system, the attack is computationally inexpensive and often requires only known ciphertext. Actual systems are potentially at risk, including cryptographic tokens, network-based cryptosystems, and other applications where attackers can make reasonably accurate timing measurements. Techniques for preventing the attack for RSA and Diffie-Hellman are presented. Some cryptosystems will need to be revised to protect against the attack, and new protocols and algorithms may need to incorporate measures to prevent timing attacks.

Keywords: timing attack, cryptanalysis, RSA, Diffie-Hellman, DSS.

1 Introduction

Cryptosystems often take slightly different amounts of time to process different inputs. Reasons include performance optimizations to bypass unnecessary operations, branching and conditional statements, RAM cache hits, processor instructions (such as multiplication and division) that run in non-fixed time, and a wide variety of other causes. Performance characteristics typically depend on both the encryption key and the input data (e.g., plaintext or ciphertext). While it is known that timing channels can leak data or keys across a controlled perimeter, intuition might suggest that unintentional timing characteristics would only reveal a small amount of information from a cryptosystem (such as the Hamming weight of the key). However, attacks are presented which can exploit timing measurements from vulnerable systems to find the entire secret key.

2 Cryptanalysis of a Simple Modular Exponentiator

Diffie-Hellman[2] and RSA[8] private-key operations consist of computing $R = y^x \bmod n$, where n is public and y can be found by an eavesdropper. The attacker's goal is to find x, the secret key. For the attack, the victim must compute $y^x \bmod n$ for several values of y, where y, n, and the computation time are known to the attacker. (If a new secret exponent x is chosen for each operation,

Stunt Cryptography (ctd)

Side-channel attacks are a perpetual-motion machine

```
Dr. Security: "I'm sorry."
System Manager: "Oh no! Not that!"
Dr. Security: "Yes ... I'm afraid it's ... channels."
```

— Bill Neugent, “Preposterous Opinions About Computer Security”, 1986

You can always find some new side-channel that’ll get you a conference paper or news headline

- The 40-year-old quote is about subliminal channels which have the same properties as side-channels, and are similarly impossible to ever eliminate fully
- Problem shown to be unsolvable by Gus Simmons, “The Prisoners’ Problem and the Subliminal Channel”, 1984

So What's the Problem?

Sounds like a win-win

- Academics get a guaranteed neverending publication stream
- Crypto geeks get to claim *their* current code, on this specific architecture with this specific compiler, is more ideologically pure than *your* current code
- Attackers get to attack as before, completely ignoring the crypto code and/or side-channels

What's the problem? Everyone wins!

So What's the Problem? (ctd)

There is one big loser in all this

- The developer or organisation that has to respond to the stunt attack

What's the cost?

- Developer time wasted trying to fix an irrelevant problem
- Management (or more developer) time wasted responding to panicked inquiries
- Reduced security due to disabling a perfectly secure mechanism in a news headline and falling back to less secure one not in a news headline
  - Variation: Compromised system due to disabling a component
- Further reduced security due to having to focus on fixing a non-problem and thereby not fixing actual problems
- Environmental cost, e.g. due to CPU microcode patches

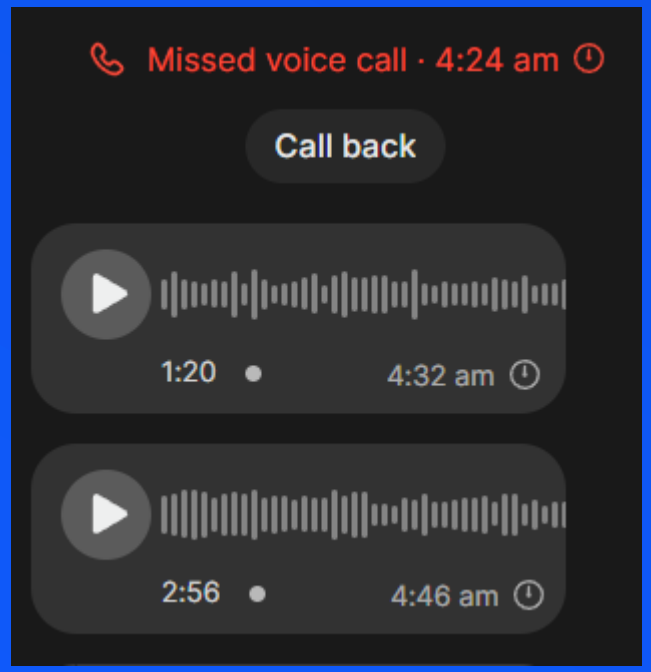
So What's the Problem? (ctd)

Security: Responsible disclosure

- "We have identified a vulnerability in XYZ. The vendor has 90 days to issue a patch, or more by negotiation if they need more time"

Stunt cryptography: Irresponsible disclosure

- "We've broken ECDSA! (Provided you do something explicitly disallowed in the standard and we have physical access to your computer)"
- Result for maintainers:



(Actual screen grab, not an edited image)

So What's the Problem? (ctd)

Case study: Developer time wasted

- EFAIL, allowing recovery of the content of encrypted email if the conditions were exactly right

Classical stunt cryptography attack

- Catchy name, logo, its own web site…
- You know it's going to be a winner

The fix: "Don't do that, then"

- Most problems were at the MUA level, not the crypto library level
- Lots of work to inform MUA developers

I can't measure the time Marcus Brinkmann and me spent on Efail

— Werner Koch, GPG

So What's the Problem? (ctd)

Case study: More developer time wasted

- GPGFail, various issues with GPG signature verification

(No fancy logo for once)

Summary: We fixed one bug that could cause a non-exploitable 1-byte buffer overflow, as well as two other bugs. None of these issues constituted a security vulnerability that could be exploited in practice. All other reports do not relate to defects in the GnuPG codebase, but to social engineering

— "Responsible Disclosure Requires Accountability: Putting Recent GnuPG Security Reports into Context"

The fix, again: "Don't do that, then"

- Time to prepare a detailed rebuttal for each claim: 3 person-weeks

So What's the Problem? (ctd)

Eventually the knowledge vanishes and the same problems reappear

- No new conference papers / publicity so they don't get addressed
- But the after-effects remain…

Case study: EFAIL

- Meetings with customers and security agencies (BSI)
- New requirements in public tenders and security policies
  - "Must have EFAIL mitigations"
  - This was specified for the crypto library, not the email apps that actually exhibit the problem

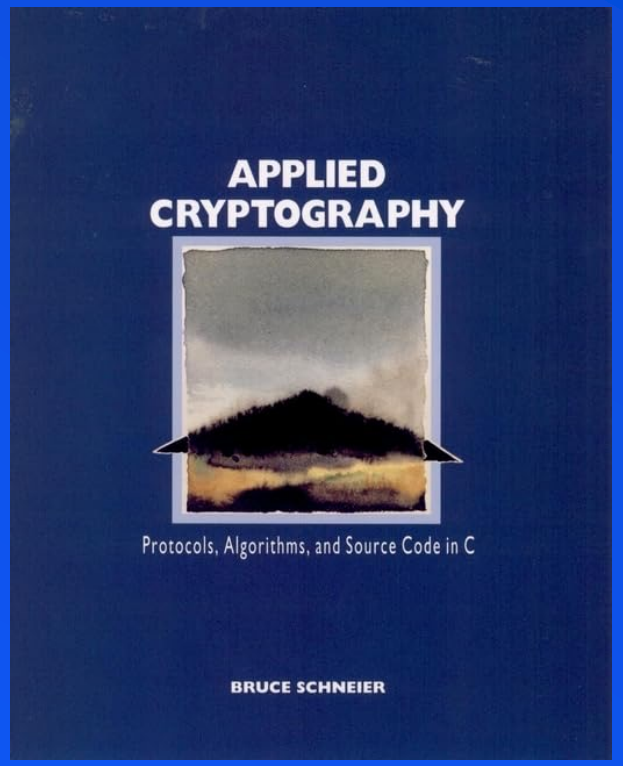
So What's the Problem? (ctd)

Case Study: Disabling a secure mechanism

- Logjam attack on DHE (Diffie-Hellman ephemeral key exchange in TLS)
- Reuse 512-bit DH values in export-grade crypto from 1994

Should-be fix:

- You're not affected, nothing to see here
  - Only broken implementations still supporting known-to-be-insecure museum-grade crypto were affected
- Use DH like Applied Cryptography (1st Ed) from 1996 tells you to



Actual fix:

- Disable DHE entirely
- Fall back to RSA, the least secure TLS key exchange mechanism there is

So What's the Problem? (ctd)

Case Study: Soft-bricking your system

- Windows has shipped with a Microsoft-maintained copy of cURL for many years, `\Windows\System32\curl.exe`

```
[C:\]which curl
curl is an external : C:\Windows\system32\curl.exe
```

- Low-severity vulnerability discovered, CVE-2022-43552 "HTTP Proxy deny use after free"
- Nothing to do, low severity and Windows Update automatically fixes it

But…

- Security scanners start reporting it as a problem
- Solution: Delete `\Windows\System32\curl.exe`
- Windows Update now detects tampering with critical files in `System32` and refuses to update

So What's the Problem? (ctd)

Case study: CPU microcode patches

- Neverending stream of papers taking advantage of assorted go-faster-stripes technology in CPUs

Should-be response:

- "That's a very clever trick, I can see you put a lot of work into it. Well done you"

Actual response:

- Patch CPU microcode to disable the go-faster stripe in question
- Cumulative patches cause between 0% and 40% performance loss depending on CPU and application
  - Let's call it 20% average

So What's the Problem? (ctd)

How does that scale?

- Around 2 billion x86 CPUs in use today
- Power draw is also wide-ranging, 5W to over 250W
  - Let's say 100W, taking into account the large number used in data centres
  - Also, "100" makes the maths easier
- Of that, 20W is wasted by microcode patches

That's 40GW of power burned unnecessarily

- 100 – 300 million tons of CO_2 per year depending on the power source (coal, gas, oil, renewables)

Forget about data centres, you can avoid a minimum 100M tonnes of CO_2 pollution each year just by undoing all of the performance-crippling patches

So What's the Problem? (ctd)

Case study: The airtight hatchway problem

> "Oh yes, I thought of something", panted Ford. Arthur looked up expectantly. "But unfortunately", continued Ford, "it rather involved being on the other side of this airtight hatchway". He kicked the hatch they'd just been through
>
> — Douglas Adams, "The Hitch Hiker's Guide to the Galaxy"

Typical manifestation: The "You need to be admin to get admin" attack

Case study: CVE-2023-51767 / "Mayhem: Targeted Corruption of Register and Stack Variables", AsiaCCS 2024

- Step 1: Send a SIGSTOP to sshd as root
- Step 2: Several pages of stunt cryptography
- Step 3: Profi…. ahh, Publish!

Research vs. Actual Attacks

Attacks in a research setting have different criteria than actual attacks

- Reduced-round ciphers
- Instrumented code
- Physical access to the hardware
- Run inside test harnesses
- Direct access to internal/low-level functions
- Able to rerun the same attack thousands or millions of times

These are legitimate *research* results, but…

Should not be passed off as having real-world or operational relevance

CVEs

Anyone can file one, and create their own severity level

- People get incentives to file CVEs, CV-Enhancements

CVEs with a web site and logo are particularly problematic

- High public visibility
- Pressure to find a fix
- Often low (as in epsilon or zero) exploitability

cURL calls these lower-than-LOW severity CVEs

> Issues we believe we serve humanity better by *not* issuing a CVE for. To avoid the security dance when it seems unnecessary
>
> — Daniel Stenberg, “A CVE Dispute”

CVEs (ctd)

Hall of Fame Worst CVEs

- This could be a looooooong talk…

Let's just take some particularly bad crypto ones

- That still makes for a very long list

OK, just the really outstandingly bad ones then

CVEs (ctd)

CVE2016-2427

- RFC 5084 “Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)”

```
aes-ICVlen              AES-CCM-ICVlen DEFAULT 12
```

- For those who don’t read ASN.1, that means the ICV (integrity check value) is 12 bytes or 96 bits
  - Value cargo-culted from IPsec, where 96 bits was all they could fit in the header
  - Also cargo-culted into TLS
  - Also cargo-culted into SSH
  - Also cargo-culted into …
- This is a standard length for AES-CCM/AES-GCM ICVs

CVEs (ctd)

The 96-bit MAC/ICV length is used everywhere

- Android
- IPsec
- TLS
- SSH
- …

The CVE, for using an ICV of the correct length as per the standard, was filed against Bouncy Castle

- Not “every crypto library and product on the planet that does this”, only Bouncy Castle
- CVE filed by Google, who have a competing Bouncy-Castle-derived library under Android

CVEs (ctd)

My favourite bogus CVE

- Enable an undocumented internal compile option that turns on extra code used for coverage/fuzz testing
  - Ignore the compile-time warnings that tell you not to do that
- Report it as a vulnerability

Ignore the fact that the same debug code implements RFC 9850 SSLKEYLOGFILE

- Diagnostic facility that dumps the plaintext TLS master secret to disk
- Needed by e.g. Wireshark to display TLS traffic

CVEs (ctd)

Recommended remediation for this “vulnerability”

- Take the undocumented internal testing/debugging option
- Draw people’s attention to it by documenting it
- Once it’s been documented, document that people shouldn’t use it

Arthur Dent: What happens if I press this button?
Ford Prefect: I wouldn’t-
Arthur Dent: Oh.
Ford Prefect: What happened?
Arthur Dent: A sign lit up, saying ‘Please do not press this button again’.

— Douglas Adams, “The Hitch Hiker’s Guide to the Galaxy”

CVEs (ctd)

Ends up in a Kafkaesque mess

- There's a CVE so it needs to be fixed
- There's no actual vulnerability present so it can't be fixed

Need to respond to each person who's seen the CVE to explain that it's bogus and there's nothing to fix

CVEs (ctd)

Speaking of which: Hall of Fame Most Kafkaesque CVEs

- IETF standard defines a particular security measure
  - "It works, because we say it does"
- It doesn't actually work
  - It's never worked
  - It was known before the first of the RFCs was even published that it didn't work
- Researcher writes a report showing all the ways in which it fails

What do you do with the ensuing CVE when/if it's filed?

- Developers should not need to solve abstract philosophical / logic problems to respond to a CVE
- My suggestion: File the CVE against the IETF standard

The Cost of Bogus CVEs

A (bogus) CVE doesn't mean you just apply a patch and push out an update

- (I'll leave aside for now the philosophical problem of being expected to provide a fix for a problem that doesn't exist)

This also affects every downstream user of your code

- Updating software at large organisations typically takes 5 to 6 months for a bugfix release
  - Many orgs have certified processes for updates that can't be changed
- 3 months for a high-severity security bug
- Using the "fast lane" in emergency situations takes at least 6 weeks
  - More usually up to 3 months as for high-severity bugs
  - Very expensive and resource-intensive so seldom used

The Cost of Bogus CVEs (ctd)

For the specific case of crypto code, FIPS makes it much worse

- CVE is platform-agnostic
- Removes FIPS certification for *all* platforms

Typical cost for a redo of FIPS-validated code

- $30,000 – 50,000 USD
- 3-6 months work
- People who have rebranded your FIPS-validated code also need to redo
  - A great many FIPS evals are rebrands of someone else's one because it's so hard to do yourself
  - Company X uses OpenSSL in their product, does a rebrand eval starting from the OpenSSL one

The Cost of Bogus CVEs (ctd)

If something like a Black Duck Security Advisory says you can't use the code due to a CVE, you need to redo

> A control system installed in a refinery in 2003 may still be running the same Windows XP-based HMI today […] because replacing it requires vendor certification, process validation, extended downtime, and capital expenditure that may run into the millions
>
> — "Practical Considerations for Legacy Malware Cleanup in OT"

The Cost of Bogus CVEs (ctd)

PCI DSS

- Critical and high vulnerabilities need to be patched or mitigated within 30 days
- All other vulnerabilities get 90 days

Updated in 2025 to allow targeted risk analysis (TRA)

- Still need to mitigate, but organisations can set their own time frames
- Even low-scoring vulns can’t be “swept under the rug” with this
  - A QSA who sees a company doing the bare minimum may fail them on the assessment
- Much debate over this, e.g. the TRA can be more effort than resolving the vulnerability

The Cost of Bogus CVEs (ctd)

A single bogus CVE for a project widely used across dozens or hundreds of organisations can snowball

- Hundreds of thousands of dollars cost
- Years, possibly decades, of human-hours work

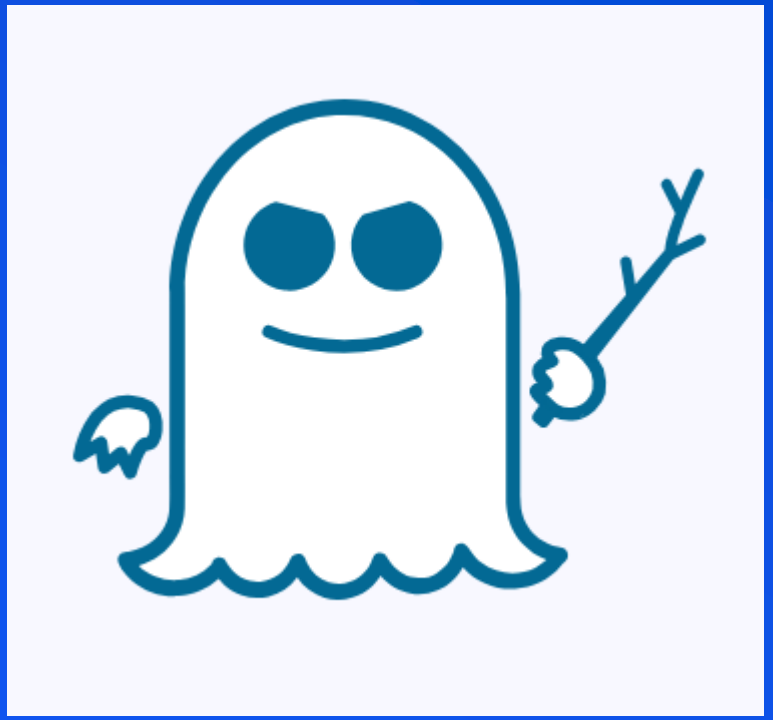
libcurl is installed in around 30 *billion* instances

> Every CVE we publish triggers activities in many security teams all over the world, leading to a significant number of patches and subsequent software updates […] Every CVE thus has this huge cost tied to it […] we don't ring the alarm for theoretical problems that will not trigger any vulnerability
>
> — Daniel Stenberg, "A CVE Dispute"

The CVE Flowchart

Vulnerability reporter CVE flowchart

- Think of a vulnerability
- Develop a PoC
  - For extra points, AI slop it
- Come up with a clever name, logo, and web site
- Alert the media (embargoed until the paper is published)
- Submit the paper for publication



The CVE Flowchart (ctd)

Maintainer CVE flowchart

- Get notified of the CVE
  - Alternatively, read about it on a news site or get a panicked call from a user
  - "Responsible disclosure? What's that?"

The PostgreSQL Security Team was made aware of CVE-2020-21469, which was filed without the prior knowledge of the PostgreSQL Security Team. THIS IS NOT A SECURITY VULNERABILITY

— "PostgreSQL: CVE-2020-21469 is not a security vulnerability"

- (To save you looking, it was an airtight-hatchway attack)

The CVE Flowchart (ctd)

Decide whether it's actually legitimate

Not legitimate

- Do we need to respond to it?
- Should we write a rebuttal (possibly weeks of work)?
- Should we just declare it fixed (makes it go away with little effort but affects downstream users)?

We'll just take the hit and claim it's fixed

— Anonymous open source maintainer with better things to do with his life

- What if it's an imaginary vulnerability that you can't announce a fix for because the problem that it's supposed to fix doesn't exist?

I'll tell you what I'm gonna do. I'm going to go home, drink moderately, and pass out

— Steve Rhoades, "Married with Children"

The CVE Flowchart (ctd)

Legitimate

- How serious is it?
- Can it actually be fixed
  - Underlying OS/hardware problem
  - Tradeoff between two or more bad alternatives
  - Arising from interaction with external components
  - Badly-designed standards require it
  - Inherent in the way the system works

The CVE Flowchart (ctd)

Legitimate (ctd)

- How will it affect downstream users?
  - Will a quick fix at our end require hundreds of downstream users to patch millions of systems?
  - Will it require support techs to visit each site and reflash the firmware in each device on site, at $10K and five days work per site?
  - Will it require re-testing and re-certifying each system that's updated?
  - In a highly regulated industry, will it require a year-long recertification process?

The CVE Flowchart (ctd)

Legitimate (ctd)

- Is it a FIPS issue, which will decertify any downstream derivative and cost tens of thousands of dollars and take months of work for each instance to recertify?
- If it's not feasible to update software/devices, how many weeks of expert time will it take to write up the risk assessment, and do we have anyone on staff who can actually do this?
- Will far more damage be caused by the knock-on effects of responding to an issue than the issue itself?

> Systems that had been running continuously for years, carrying infections that were stable and effectively headless, failed during cleanup attempts on fragile and low-resourced legacy hardware […] In each case, the infection itself had not caused a significant operational incident. The responders did
>
> — "Practical Considerations for Legacy Malware Cleanup in OT"

The CVE Flowchart (ctd)

Case study: Pre-CVE bug report

- Technically this would be a CVE

For every CVE with CVSS > 7 we need to verify whether it's exploitable. If yes, an immediate patch is required. If not, patch by the next release, at the latest in six weeks. So we need to read each advisory and see if it affects us in any way […] this costs hours of time

— "WernerderChamp", Heise Online (transl.)

- With grade inflation, far too many CVEs end up with CVSS > 7

Is it worth actually raising a CVE over?

The CVE Flowchart (ctd)

Raise a CVE

- Dooms dozens or hundreds of downstream users to what's on the previous slide
- In turn can require software updates, firmware updates, device replacement…
- If updates are too hard to do, device wears the Mark of Cain forever

After lengthy analysis, don't raise a CVE

- Technically the found bug is a problem, but highly unlikely to ever trigger
- Spares downstream users a huge amount of pain
- Can be accused of trying to cover up a vulnerability
  - Bug reporter sees it as a critical flaw
  - Responder sees it as a way to make downstream users waste huge amounts of time and money fixing an irrelevant issue

The CVE Flowchart (ctd)

AI makes this particularly problematic

- Not one report but a sudden deluge of 100 problem reports
- Not AI slop, actual real problems
  - Once you get past the hype, frontier models are capable of impressive levels of analysis if correctly applied
  - Code that has had 20-30 years of use and endless human audits is suddenly found to have a whole catalogue of issues that no-one spotted before
- Some of these are deep-rooted, e.g. going back to design decisions made 20 years ago
  - Several OSS projects have estimated that around half of AI-discovered issues date back 20 years or more

The CVE Flowchart (ctd)

The CVE system was never designed for this

- File 100 CVEs?
  - (That's a rhetorical question, you're just creating a CVE DoS that way)
- File an umbrella CVE for 80 of them, "there are some minor issues, nothing to worry about, it'll be auto-fixed in the next update"?

Various approaches

- With Postfix, my priority is issuing 'must fix' updates rather than issuing CVEs. I may issue some CVEs after the storm subsides
  — Wietse Venema
- My approach: Contact large users and ask them how much pain it would cause them if I raised issue XYZ (minor things, not 0day) as a CVE
- Typical response: "Please, please, don't!"

Problems with Tools

Security scanners report large numbers of make-the-scanner-look-useful "vulnerabilities"

Case study: Failing to sanitise memory

- Every security scanner obsessively warns about this
  - It's low-hanging fruit for any tool that can do taint analysis / tracking
- No-one has ever been attacked with it
  - If an attacker can perform arbitrary memory reads then it's already game over

It's just noise

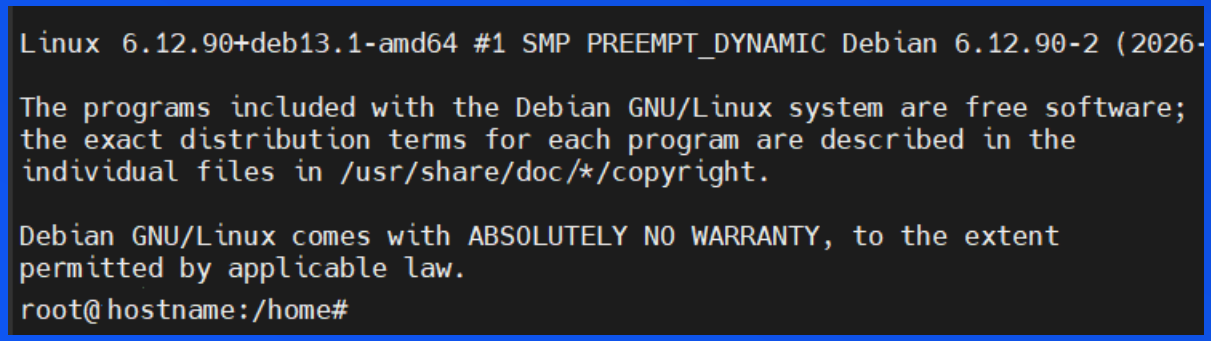
- `zeroise(data,length)` → `virtueSignal(data,length)`

Problems with Tools (ctd)

However you can still get CVEs for these scanner non-vulnerabilities

Case study:

- User has physical access to the server
- Logs in as root
- Runs memory-scanning software
- Files a CVE because they could see sensitive data in memory
- CVE was approved and issued



You were serious about dat?

— Vinny Gambini, “My Cousin Vinny”

Problems with Tools (ctd)

You need to respond to these each time

- Many companies have rules that you need to respond to any vulnerability scanner warning within *X* days

Case study: The vulnerability scanner game

- Company needs a backport of fixes for old copies of software
  - Long-obsolete versions or features that are never used
- Fix for every CVE that any version is listed as vulnerable to, even if it's never used
- Can now publish an update for each CVE record saying that it's fixed
- No effect on security, done purely to stop vulnerability scanners from complaining

How do we Fix This?

You can make any project look bad

- So how about fixing it?
- Project developer makes a decision, driven by constraints at the time, that is less than absolutely perfect a decade later
- Easy to criticise
  - I've done this myself – oops! Sorry folks :-/

What are some ways to lessen the problem?

How do we Fix This? (ctd)

CVE target gets to define the severity, not the CVE author

Examples of problems that have a severity level of 0.0:

- Airtight hatchway problem
- Doesn't exist / manufactured vulnerability
- Decades-old attack that no attacker has ever used in all that time

Allow the CVE target to indicate its severity level

- Including, importantly, "not applicable"
- Also, "vanity" and "my quarterly bonus depends on this"

Would help stem the flood of CV-Enhancements

- Especially the severity level of "vanity" and "my bonus depends on this"

How do we Fix This? (ctd)

CVEs assume that every bug is a vulnerability

- Used by the Linux kernel folks for malicious compliance

> Due to the layer at which the Linux kernel is in a system, almost any bug might be exploitable […] Because of this, the CVE assignment team is overly cautious and assign CVE numbers to any bugfix that they identify

- This led to the exact problems covered in this talk

> To say this is a complete disaster is an understatement
>
> — Bradley Spengler, Open Source Security

Many defects are just plain ordinary bugs, not vulnerabilities

- Alongside “vanity”, allow “bug” as a category
- Neither of these are vulnerabilities

How do we Fix This? (ctd)

AI surfaces issues in large batches

- Referring to legitimate reports here, not AI slop
- The CVE process is unable to deal with this
- Creates a CVE DoS

Allow for batch CVEs

- “A lot of minor issues, none too serious”
  - Even adding a one-line summary of each issue has problems
  - Summary: A remote user can reformat your hard drive
  - Analysis: 3-page writeup on why the conditions to allow this never occur
- Update when you can

How do we Fix This? (ctd)

Don’t be OpenSSL

- Any totally impractical attack + OpenSSL = conference paper
- “Differential Timing Attack on PolyCrypt” = one-line mention in Crypto Dispatches
- OpenSSH and GnuPG are the fallbacks if you can’t target OpenSSL for the publicity

Conference referees

- Would this paper still get accepted if the attack wasn’t against OpenSSL?

How do we Fix This? (ctd)

Become your own CNA

- Allows you to catch bogus CVEs before they're issued
- A number of CNAs exist purely as a result of especially egregious CVEs

I personally think it is rather problematic that this basically is the only available means to prevent this level of crap

— Daniel Stenberg, "Bogus CVE follow-ups"

- Unfortunately this isn't practical / feasible for projects run by individuals

How do we Fix This? (ctd)

Push back against bogus CVEs

- OpenSSH tried this with CVE-2023-51767
- Very difficult to do
  - "We'll just take the hit and claim it's fixed" makes it go away
  - Just to be clear, this is for a "vulnerability" that doesn't exist
  - See earlier comments about the Kafkaesque situation this creates since you can't "fix" a vuln that doesn't exist in the first place
- At best you can get a "disputed" status after months of effort even for a totally invalid CVE
  - CVE2016-2427 for example is "disputed"
  - No timeline for disputes, they essentially remain locked in limbo forever

How do we Fix This? (ctd)

Make it acceptable in academia to publish a paper that says “here’s a cool attack, totally impractical but a neat trick”

- I can’t see this ever happening
- Decades of precedent require that you have to have a real-world problem for your paper to solve

Case study: Researcher comes up with an interesting algorithm

- Invents a problem for it to solve
- Submits it for publication at a security conference
- The next year the same conference gets three more submissions of papers to solve the same (invented) problem

Conclusion

Stunt cryptography comes with a real cost

- Triggers a governance process in which maintainers need to respond to an often nonexistent threat
- Can take months of work and cost tens of thousands of dollars to address

Like payments processing, the entire system is optimised to move in the forwards direction

- Chargebacks are extremely painful and often just not worth it

Needs a mechanism to correct the one-way flow and reduce the cost to projects

- “Severity = Vanity” / “My bonus depends on this” is probably the most effective way to fix this
- Some way to deal with AI-generated batched reports is also needed

Endnotes

This talk was prepared with input from a number of OSS projects

I’m always interested in more information, feedback, and thoughts on this

- In particular, data and case studies illustrating the cost of bogus CVEs