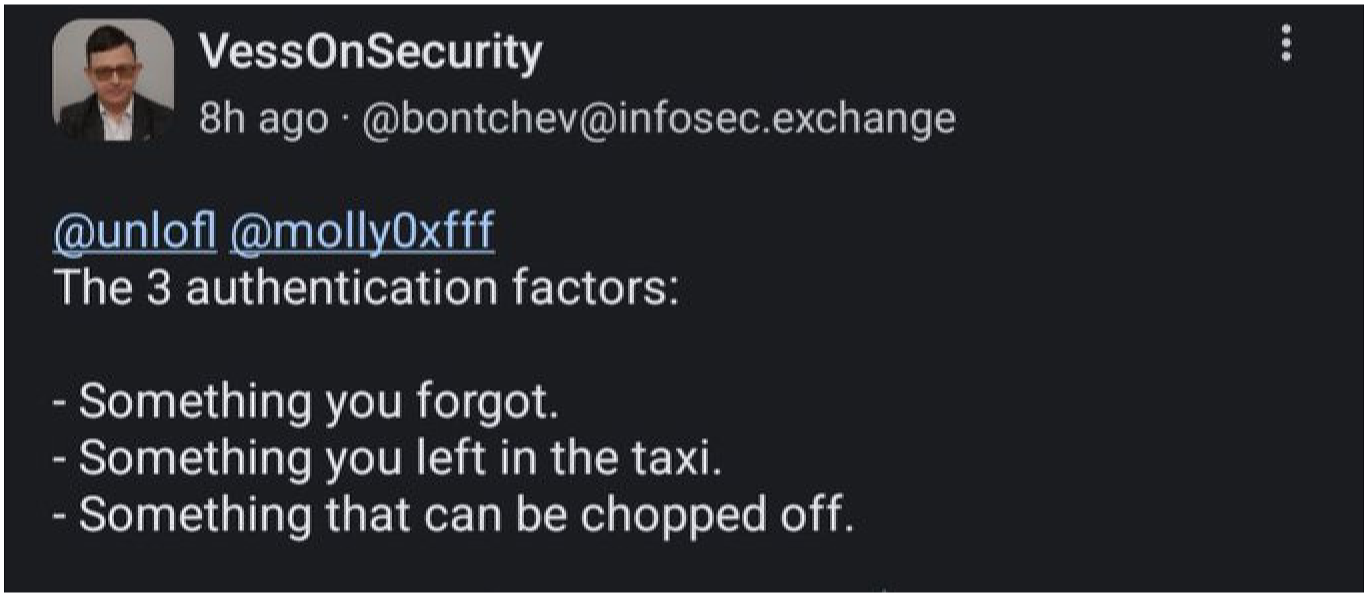
Security Now! #948 - 11-14-23

What if a Bit Flipped?

This week on Security Now!

Is your lack of privacy badgering you? And if so what can you do about it? What's the latest on last week's bombshell news of the EU's Article 45 in eIDAS 2.0? Who's lost how much money in online cryptocurrency? And is using seed phrases for your wallet that to get from a seed phrase suggestion site a good idea? Has there finally been a truly devastating and effective speculative execution flaw discovered in Intel's processors? Could it be their Downfall? What country has decided to ban all VPNs? And how bad are the two flaws found in OpenVPN? Why have I stopped working on SpinRite? What's the best backup for a large NAS? Should vulnerability researchers learn the assembly language of their target processors? If quantum computers threaten asymmetric crypto, why not return to symmetric crypto? Could someone explain exactly why Article 45 is a bad thing? What in the world is a Windshield Barnacle and why don't you want one? What's my latest Sci-Fi book series discovery? And just how bad could it be if a cosmic ray flipped a bit at just the wrong time?

A pessimistic view of multi-factor authentication:



VessOnSecurity
8h ago · @bontchev@infosec.exchange

@unlofl @molly0xfff
The 3 authentication factors:

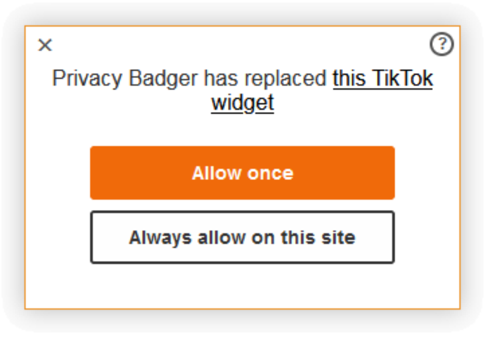
- Something you forgot.
- Something you left in the taxi.
- Something that can be chopped off.

With thanks to Robert Gauld / @RobertGauld

Security News

Privacy Badger

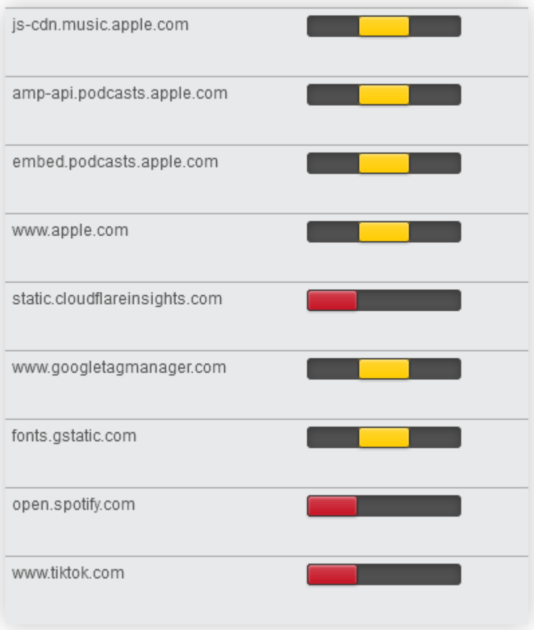
A bit of a follow-up on the annoyingly-named but nicely functioning Privacy Badger from the EFF. While assembling today's podcast I was greeted with this in the middle of a page at a news aggregation site:



Privacy Badger has replaced this TikTok widget

And my first thought was "thank you" – I have no interest in having my browser going to TikTok to retrieve whatever-it-was that this website wanted my browser to have – and in the process expose itself to TikTok as a first party. Thanks anyway. I don't know what would have been shown in that location where Privacy Badger posted a fill-in, and I don't care.

While catching up with my Twitter feed yesterday I ran across someone who was passing along the advice he'd heard from somewhere else that uBlock Origin and Privacy Badger should not both be installed because they interfered with one another and because uBlock Origin was a superset of Privacy Badger. I've seen zero evidence that any of that is true. Looking at the Privacy Badger icon for that web page I was on, I saw that it had prevented the loading of nine things on that page:



js-cdn.music.apple.com	🟨
amp-api.podcasts.apple.com	🟨
embed.podcasts.apple.com	🟨
www.apple.com	🟨
static.cloudflareinsights.com	🟥
www.googletagmanager.com	🟨
fonts.gstatic.com	🟨
open.spotify.com	🟥
www.tiktok.com	🟥

For its part, uBlock Origin blocked 10 fetches, preventing my browser from exposing itself to 4 of the page's 19 offered domains, including "sentry-cdn.com" and "datadoghq-browser-agent.com". I was morbidly curious about "DatadogHQ" which describes itself: *"The Datadog Agent is software that runs on your hosts. It collects events and metrics from hosts and sends them to Datadog, where you can analyze your monitoring and performance data."* That's right. Just as I have no interest in having my browser to snuggle up to TikTok, I don't need DataDog to be collecting any metrics about me, thank you very much. Needless to say, I've always been happy having uBlock Origin watching my back. And I'm equally happy having added the Privacy Badger to the team.

Article 45 of eIDAS 2.0
I looked around for anything happening on the eIDAS 2.0 front since last week and nothing much appears to be new. The Register weighed in on Wednesday in the inimitable snarky style with the headline *"Bad eIDAS: Europe ready to intercept, spy on your encrypted HTTPS connections."* But they didn't have anything new to add since there's nothing new to add. As I noted last week, we haven't quite reached the ultimatum stage where the OS and browser vendors will simply refuse to turn over the management of global trust to political interests. But given what we just witnessed a few weeks ago, where UK politicians refused to back down until every encrypted messaging provider, including Apple, made very clear that they would withdraw their services rather than comply with a bad law, the next thing we see may be a repetition of the same drama. I think the best way to view this is as we did last week: "The management of today's Internet and encryption technology is bigger than any single, or group, of governments." That lack of absolute control over their populace doesn't make governments happy, but it does appear to be the way the world is shaking out. And I think it's clearly the way it should shake out.

"Online CryptoCurrency" – what could possibly go wrong?
I have two quick news items to share, then something to add.

1. The Cryptocurrency exchange Poloniex lost $130 million worth of assets after hackers drained its hot wallet. (I guess the wallet was hot and now its cryptocurrency is hot.) Poloniex confirmed the hack, paused transactions, and promised to reimburse user losses. And this is the exchange's second heist after it was also hacked back in 2014. This $130 million loss ranks as the 14th-largest cryptocurrency hack ever.

2. And, second, the Decentralized finance (DeFi) platform Raft has lost $3.3 million worth of cryptocurrency after a hacker exploited a vulnerability in its platform. The company confirmed the hack on social media and paused the minting of its R stablecoin to investigate the incident.

The one observation I wanted to make was to once again pose our rhetorical question: "What... could possibly go wrong... when the integrity of the storage of shockingly large amounts of real world actual **money** depends upon software that very few people even understand? You don't have a $130 million dollar heist being ranked as the 14th largest in any industry that has any idea what it's doing. So I'll just reiterate to please be very careful and heed the ago old advice to never invest – just as in never gambling – more than you can afford to lose.

I don't give one IOTA!
And before we run away screaming from the topic of "what could possibly go wrong" with cryptocurrency, get a load of this one: UK police are currently in the process of returning around £1.9 million worth of cryptocurrency that was stolen back in January of 2018 by a Dutch hacker. This individual operated a website called: iotaseed.io. And you're gonna love this: As its name suggests, the iotaseed.io site was created to generate seed phrases for IOTA cryptocurrency wallets... which the wallet's users then apparently used without any change. So... who can guess what happened? Right. The site's criminal hacker/owner was recording all of the seed phrases

his site was suggesting that its visitors should use to protect their wallets. To no one's surprise other than all of those visitors whose money this guy then stole, the crook broke into their wallets and did just that. He was caught and following a Europol investigation was sentenced to four and a half years in prison. What's that expression? *"A fool and his money are soon parted."*

Downfall

"Downfall" is the name of yet another information disclosure vulnerability that was recently rediscovered in Intel's chips. And unlike Spectre and Meltdown, this one really has some teeth. Here's what it's rediscoverer wrote:

> *Downfall attacks target a critical weakness found in billions of modern processors used in personal and cloud computers. This vulnerability, identified as CVE-2022-40982, enables a user to access and steal data from other users who share the same computer. For instance, a malicious app obtained from an app store could use the Downfall attack to steal sensitive information like passwords, encryption keys, and private data such as banking details, personal emails, and messages. Similarly, in cloud computing environments, a malicious customer could exploit the Downfall vulnerability to steal data and credentials from other customers who share the same cloud computer.*
>
> *The vulnerability is caused by memory optimization features in Intel processors that unintentionally reveal internal hardware registers to software. This allows untrusted software to access data stored by other programs, which should not normally be accessible. I discovered that the* ***Gather*** *instruction, meant to speed up accessing scattered data in memory, leaks the content of the internal vector register file during speculative execution. To exploit this vulnerability, I introduced Gather Data Sampling (GDS) and Gather Value Injection (GVI) techniques.*

Whereas we never had any demonstrations for the Spectre and Meltdown attacks because they remained theoretical, the Downfall site (https://downfall.page/) shows videos of 128 and 256 bit AES keys being stolen from another user, of arbitrary data being stolen from the Linux kernel and more. All of Intel's "Core" processor from 6th generation Skylake to and the 11th generation Tiger Lake are affected. So this is nine years worth of Intel's processors, since the 6th-gen Skylake chips appeared back in 2014. And the attacks using this are practical. Its author wrote:

> *GDS is highly practical. It took me two weeks to develop an end-to-end attack stealing encryption keys from OpenSSL. It only requires the attacker and victim to share the same physical processor core, which frequently happens on modern-day computers, implementing preemptive multitasking and simultaneous multithreading.*

Intel has released a microcode update which blocks transient results of Gather instructions and prevents attacker code from observing speculative data from Gather. That's the good news. The bad news is that depending upon how much benefit the processor may have been obtaining from this speculative execution optimization, the impact on performance might be as much as 50%. This is Intel's own confession and estimation.

There is one thing all of these vulnerabilities have in common: They are all about speculative execution. In other words, Intel had this terrific idea that since the laws of physics were making it impossible to keep increasing the processor's clock, a way to speed up code further was to create a massive over abundance of processor power and allocate that excess to the task of executing code speculatively. The first idea is that the processor would be allowed to run ahead by pre-fetching from memory ahead of its actual execution. And if that prefetch system encountered a branch instruction, it would follow both code paths – the branched and the non-branched – by continuing to fetch instructions. Then, at some point in the future, once the processor's execution had caught up to the branch, it would know which path to take and all of the work on the other path could be discarded.

The side-channel edge case that Intel failed to pay sufficient attention to was that all of that extra work that was discarded left some effect behind. Memory was fetched that wasn't wasn't ever needed or used. The cache was filled with some of that. Other contents that would have been in the cache were evicted to make room. Remnants were left behind in the branch prediction logic that's used to make decisions when insufficient information is available, and on and on and on. This has been biting Intel and, to a somewhat lesser degree, AMD, for many years now.

In the author's Q&A the question is posed: "Why is this called Downfall?" and the author replies: *"Downfall defeats fundamental security boundaries in most computers and is a successor to previous data leaking vulnerabilities in CPUs including Meltdown and Fallout. In this trilogy, Downfall defeats all previous mitigations once again."*

At the start of this I referred to this author as the "rediscoverer" of this. The reason is that Intel was informed of this vulnerability twice before, back in 2018, and chose, for whatever reason at the time they opted to ignore the problem, perhaps this was due to the damaging effect it might have on their public relations, or maybe because of the massive performance hit any patched microcode would force. Either way, its rediscovery, complete with ample proofs of concept and source code on Github cannot be swept under the rug.

And, predictably, a class action lawsuit has been filed by five aggrieved so-called victims and their ambulance chasing attorney. It's necessary to show actual concrete damage rather than to simply be put out by Intel's past behavior. So it's unlikely that this is anything more than a ploy to get Intel to payout some modest settlement to make this go away.

No VPN's for you!!
We've touched on this before, but I wanted to note that Russian officials are preparing to formally ban the use of all VPN services in the country. **Roskomnadzor** has been testing blocks for various VPN protocols and services over the past year in preparation for creating a formal blockage. Officials say a formal VPN block is needed for the safety of the Russian internet.

OpenVPN
Speaking of VPN's, if you're using OpenVPN check for two security updates. Neither one is critical, but one could potentially provide access to memory. So updating would be wise:
https://openvpn.net/security-advisory/access-server-security-update-cve-2023-46849-cve-2023-46850/

SpinRite

I've stopped working on SpinRite... and not because I've been distracted by anything else. But because every indication is that both its DOS and its Windows code is well and truly finished. It took five release candidates of each one, for us to get there. But there is nothing left for me to do there. So I've immediately turned my attention to the next piece of the work, which is updating our GRC server's Windows executable code delivery system to perform code signing on the fly. Until now, code signing has always been a manual process. This was fine when I was signing the DNS Benchmark, Never10 or InControl where that one download would live on for years. But now that needs to be automated so that it can be performed on GRC's server, since each purchased and licensed copy of SpinRite is unique to its user. So I'm working on getting code signing automation in place. Once we have that, the existing pre-release page at GRC will begin offering the final hybrid DOS/Windows code and I guess I'll remove the pre-release caveats from the final product. Then I'll spend some time putting an owners manual online, then I'll bring up an eMail facility so that I can begin sharing the news of v6.1 with all of SpinRite's current users. And then I'll be able to start work on SpinRite 7!

Closing the Loop

EdgeIT / @Edge_IT2

@SGgrc Great show.. I have been using Sophos XG/XGS firewalls for my customers for a while. They have had this automatic hot fix feature which is enabled by default for many years. And a few years ago when there was a vulnerability, it was patched for every customer while I was sleeping.

Jim Harbin / @jharbinjr

Hey Steve. Long time listener and so glad I can continue to be for a long time more. Question around NAS backups. Both you and Leo mention Synology's and Syncthing.. I've went with a QNAP and have 30 TB of RAID storage in a four bay unit, but outside of this NAS this data isn't replicated. What would be the best/most cost efficient way to create some redundancy, other than my single NAS? Long ago are the old tape backup days.

This is really a problem. Spinning magnetic storage has become astonishingly inexpensive. So I doubt there's any superior solution than simply using more spinning storage. As I noted last week, I run RAID-6 exclusively. So each array has two drives of redundancy. What you might do, if you just want another layer of redundancy and access in the event that the entire NAS goes down – which would not be unheard of – would be to run another single spinning drive on some always-on machine and keep it synchronized with the NAS. But the bottom line is, I doubt that it's possible to do better than spinning drives, and then make up for the possibility of their failure with redundancy.

Jorge Moran / @0xjams

Hi Steve, Knowing how skilled you are at actually producing asm code, I was wondering if you've ever tried to do some malware analysis or exploit development? After some time *studying this, I've come to the conclusion that it might be better to step back and actually learn to write asm code from scratch first. What do you think?*

I guess I'm of two minds about that. On the one hand, everything is ultimately assembly code since that's what our chips still run. Various attempts to create higher-level chips have been undertaken. The "Forth" processor comes to mind. But we appear to be moving in the other direction, from CISC architectures to RISC instruction sets. So being able to understand code at its lowest levels can certainly be handy.

But on the other hand, we just saw two weeks ago that the code reverse engineering tool, Ghidra was producing high-level decompiled code. So it wasn't just serving as a disassembler but also as a decompiler – taking the disassembled code up another level to create much more context for what the binary bits were doing. The code that the guys who were tracking down the Citrix Bleed flaws received from Ghidra was beautiful higher-level code.

Trevor @ infosec.exchange

Hi Steve, Insert often repeated "logging back into twitter for the first time in a long time just to send this spiel"

I've been thinking about crypto and prime factorization lately and how to apply the "assume breach" way of thinking to this problem. My thinking is: assume that a bad actor has or will have access to a sufficiently capable quantum computer, how can you protect against this? Something in the context of a messaging app.

Public key crypto and Diffie-Hellman are both in the crosshairs of quantum computing since Shor's algorithm can efficiently factorize the large primes that RSA depends on and can efficiently solve the discrete logarithm problem on which Diffie-Hellman is based.

Post-quantum algorithms are also still in their infancy and as we have seen before, new crypto systems need sufficient and constant beatings until they are deemed acceptably secure

My first thought was: find a way to stop using public key crypto or reliance on Diffie-Hellman for your communications with another person, maybe find a way to get them a symmetric key.

Well how can you communicate a shared secret securely without having an already pre shared key? You would need to use public key crypto/Diffie-hellman to share the key over the wire but that transmission could be broken and your secret revealed. So I was thinking about you saying "if you really want to communicate privately, strip naked and sit in the middle of a field with a blanket covering the both of you while you whisper your secret"

Why can't we just exchange symmetric keys with people we interact with physically? We can now share our contact information with another party by touching our iphones together, how come we can't at the same time just exchange a list of (n) number of keys and use those for iMessage (or another messaging platform) and then we do not run the risk of intercepted communications being broken? We could then use a single key until there is reason to believe it needs to be changed for any reason and cycle down the list.

Anyway those were just my thoughts as someone who is interested in, but not in the business *of secure communications. Would be happy to hear why I'm thinking about this wrong or why there are better ideas. Anyway, hope you're doing well. Big fan of the show. Thanks, Trevor*

Trevor is thinking about all of this correctly. Though today it feels like throwing the baby out with the bath water. The largest leap forward we made in cryptography was in switching from the early proprietary and secret non-keyed one-off encryption schemes over to open and public ciphers whose specific behavior was specified and set with a key. This, of course, was the invention of symmetric key crypto.

The next huge step forward was the invention of asymmetric crypto which gave us the ability to have public keying negotiations without fear of eavesdropping and the incredible power that flows from the use of separate public and private keys.

Given the tremendous power of asymmetric keying, it's difficult to imagine returning to the limitations of a symmetric keying only world. For example, one of the other features made possible by asymmetric keying is the ability to have perfect forward secrecy. Since asymmetric keying allows us to encrypt under a different symmetric key for every new connection or conversation. This means that if a symmetric key were ever to somehow escape, only that single conversation would be at risk. Neither any other past conversations nor any in the future would be subject to disclosure. This is a powerful feature that we loSe if we're unable to safely negotiate new symmetric keys.

So Trevor is right that it's only the asymmetric keying that's put at risk by quantum computing. And he's right that the new quantum safe replacement algorithms need time to mature and be tested. But since virtually all of today's applications are based upon asymmetric keying, I cannot imagine going backward.

The good news is that the industry's inertia, which we spend so much time lamenting on this podcast will not pose a problem for this industry. Closed systems like Apple's, Signal's and others will be able to move seamlessly overnight. And open systems like the web can be enforced from the end with the fewest players, namely the browsers. Once Google announces the Chrome browser's and Chromium's drop-dead date for non-quantum safe algorithms, web servers will have no choice other than to support the new safe standards or find their visitors receiving scary looking warnings meant to put pressure on those server operators to get their act together. And at some point Chrome and other browsers will stop being willing to connect.

So I suspect that the wisdom we're already seeing by the industry moving forward on next generation crypto will serve us well and will likely keep us well ahead of the day when quantum computing is first able to crack any asymmetric crypto. I think that in this instance, it makes a lot more sense to move forward than backward. (What was back there was so limited!)

Someone in GRC's newsgroup asked:

I feel really embarrassed about this, but ………, I've never really understood how the root store in the OS and the keys in the browser give the protection or as in last week's podcast mentioned, how having a key from a bad source (a State maybe) allows intercepted traffic to *be decrypted even if I was logging into for example my local supermarket. Is there a nice concise explanation anywhere to help me and a follow up with the more complex aspects. I'm looking for some context to start my understanding on a sound footing. Thanks.*

I occasionally receive Tweets from people who say that they don't understand all of what's said on this podcast, but they get enough from it that it's worth their while. Someone made a comment about sipping from a fire hose. So I thought that this person's honest question might be appreciated by others who might also feel a bit left behind. So here's how all that works:

When a browser connects to a web server, during that so-called "connection handshake", the server sends the browser a claim of its identity in the form of a digital certificate. A certificate is nothing more than a collection of information fields which, after having all been set correctly, are then signed. After the certificate has been signed it becomes tamper proof. Any change in its data will break its signature.

In the case of web servers, the certificate that the web server sends to the browser will have been signed by a "Certificate Authority". A "certificate authority" is nothing more than its name suggests: It's an organization the world and the world's browsers have all agreed to trust. These certificate authorities go to varying degrees of effort to independently verify the identity of the organizations whose certificates they sign.

So, if the certificate authority says that the entity presenting this certificate is "Joe", and if we trust the certificate authority to do its business properly, then we trust that this is, indeed, "Joe."

Okay. So now the question is, which certificate authorities should we trust? And that's where the "root store" comes in. Certificate Authorities also have their own certificates which attest to their identities. And since they trust themselves to be who they say they are, they each sign their own certificate. So what we refer to as the root store is the collection of certificates belonging to all of the certificate authorities whose signatures of website identity certificates we have decided to trust.

So, when a website presents the browser with a claim of its identity which will have been signed by some certificate authority, the browser looks in its root store for the matching certificate of the certificate authority that signed that identity claim. If it's found, that authority's certificate contains the public key that's then used to verify the signature of the web site's certificate.

Okay. So that's part one. That's how the system works. Part two is how traffic interception can occur...

We've often talked about network traffic interception. The traditional term for it is "man in the middle", meaning that some entity arranges to place their equipment in the path of communication so that they're able to monitor everything that flows through them. Normally, the use of HTTPS for web traffic, with its TLS for encryption, would render this ineffective since all of the traffic moving past would be encrypted. But if the "man in the middle" themselves was also a trusted certificate authority, the man in the middle could pretend to **be** the website at the far end of the connection.
So when the user was attempting to connect to, say, a racy dating site, the "appliance in the

middle" would create and sign a website certificate for that racy dating site on the fly and return it to the user's browser. If that appliance's certificate was trusted by the browser, no red flags would be raised, the padlock would be present, the connection would appear to be secure, and the user would 100% believe that they had connected directly to the racy dating site.

But they would have, instead, connected to the appliance in the middle where all of their traffic, their username and password and cookies and everything they did while at that sight would be decrypted and made available. In order to complete the illusion, the appliance itself would have made the connection to the dating site, pretending to be the user's browser. So it would pass along everything that transpired in both directions, while being privy to the entire dialog without any encryption.

Now, this is already exactly what **is** being done in corporations that deeply monitor all of their employee's communications. To do this, every web browser in the enterprise must receive and accept that "middlebox" appliance's certificate. This allows for full transparent "deep packet inspection" of all of the enterprise's network communications. And this is precisely the power that the EU is saying **it wants for itself and for all of its member countries.** Article 45 of the eIDAS v2.0 insists that without any oversight or governance, any member state can provide a certificate that all web browsers **MUST** – as a matter of European law – blindly and fully accept and trust without question.

There is only one possible reason for wanting this, which is to allow for on-the-fly Internet web traffic interception – exactly as happens inside of corporations. And that is acknowledged. Their claim, and I believe their hearts are in the right place, is that they want to have the ability to provide additional identity verification of websites for their citizenry. They want to be able to inject top-of-page identity assertion banners into EU website properties to provide an additional assertion of sites' trustworthiness.

It must be that some misguided techie, sometime in the past, cooked up this idea and sold it to the EU's legislators. This techie explained that if they could get their own certificates installed into the world's few web browsers and operating systems – there are not that many of them, after all – then they would have the ability to inject whatever they might wish to – like national emergency warnings or other public interest messages – into every one of their citizen's downloaded web pages.

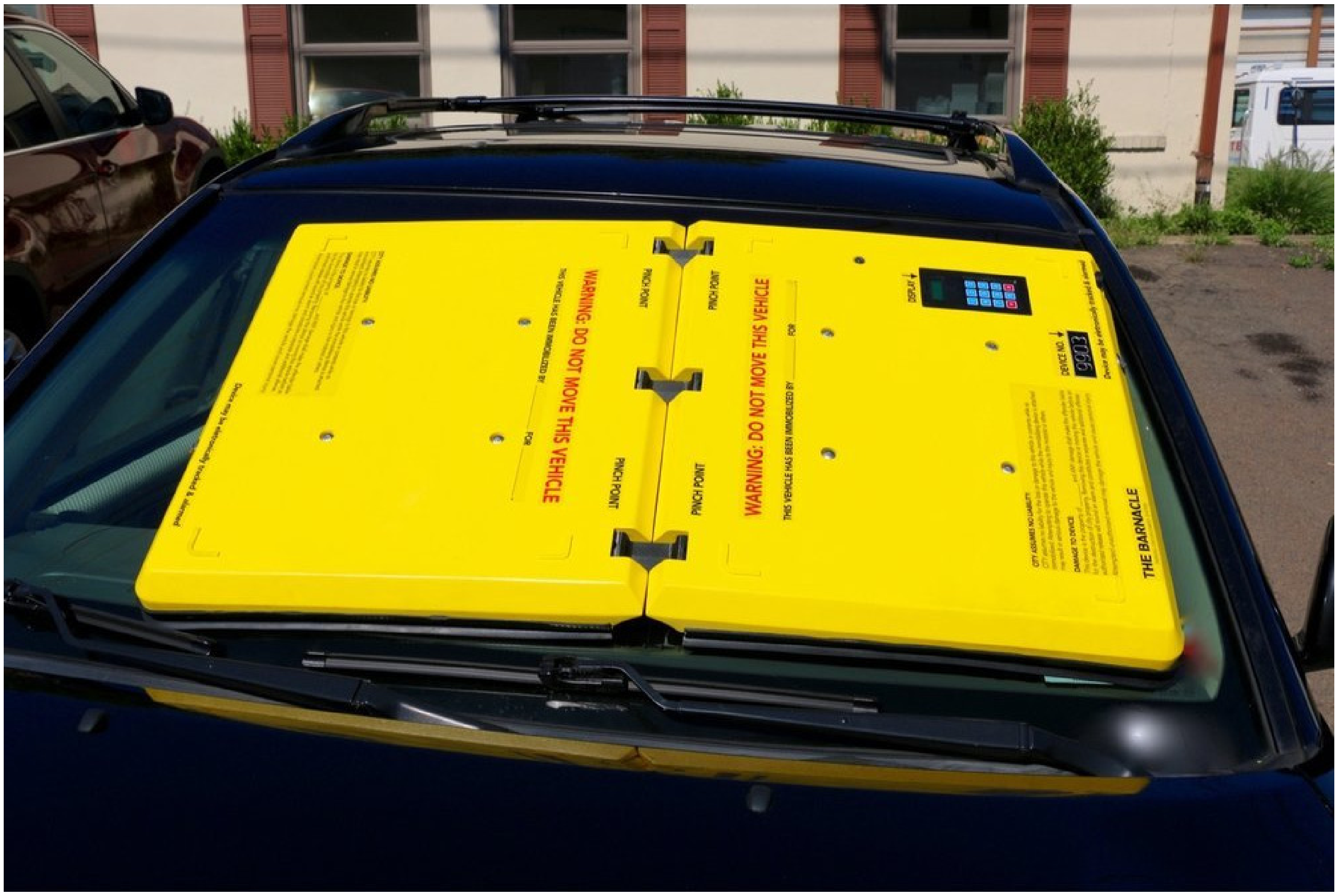
The trouble is that the EU and its member nations are very different from the employees of a private organization. Any time an employee doesn't want to be spied upon they can use their own smartphone to circumvent their employer's network. And, of course, an employer's private network is just that – a private network. The EU wants to do this for the entire public Internet from which there would be no escape.

It seems quite clear that this is not going to happen; but the exact way it doesn't happen should prove to be quite interesting.

Miscellany

The Windshield Barnacle

Our frequent Tweeter, Simon Zerafa Tweeted this photo to @SGgrc which raised more questions than it answered:


WARNING: DO NOT MOVE THIS VEHICLE
WARNING: DO NOT MOVE THIS VEHICLE
PINCH POINT
DISPLAY
DEVICE NO.
9913
THE BARNACLE

For those who are not watching the video, what we see is a huge bright yellow slab which is covering most of a car's windshield. It's about an inch thick and is hinged in the center. In the upper right corner we see a membrane-style keypad and display and, along with the red warning "DO NOT MOVE THIS VEHICLE" in the lower right we see what is apparently the name of this thing – which is just too wonderful – since it's called "The Barnacle" ... and I love everything about this.

The reason it's "The Barnacle" is that underneath, two large full-contact suction cups, one on each half, are adhering this bright yellow eyesore to the windshield and, like any good barnacle, it will **not** let go. Its creator claims that it generates in excess of 1000 pounds of resistance to being removed. So, you return to find that this barnacle has attached itself to your presumably wrongly parked car. Either the meter ran out, you parked in an unauthorized location, you're taking up to two parking slots, or you have several years worth of unpaid parking tickets and "The Barnacle" has just caught up with you. Now what? This large yellow thing is stuck to your windshield, totally obstructing your view from the driver's seat and making it impossible to drive.

The view from the driver's seat:

So you are reduced to visiting the website that's shown on The Barnacle, or calling the toll free number. You identify the device instance by its very clearly posted serial number, and you are told how much you owe – which must be paid, in full, on the spot by credit card. Once you have entered in your credit card payment information to pay the parking fine, you're told that your account has also been charged a separate refundable fee for "The Barnacle" itself, whose return to a nearby receptacle you will now be responsible for, once the device has released itself from your windshield. After agreeing to this secondary contract you are given the device's current one-time release code which you enter into the membrane keypad. And then, presumably with a sigh of releasing suction, The Barnacle releases its grip on your windshield. You then refold it in half and slip it into the conveniently located nearby parking lot return stand.

Note that The Barnacle is equipped with GPS and will sound an extremely loud and piercing alarm if it detects that the car is moving after having been "Barnacled." The traditional way of dealing with scofflaws who ignore their parking tickets or other parking-related violations is to "Boot" a wheel of their car to prevent it from being driven. The trouble is, this requires two visits from the "booter" – one to affix the boot and another to remove it. So the Barnacle creates what is essentially "self-serve unbooting." A parking ticket slipped under a windshield wiper is easy to ignore. But affixing a Barnacle to a windshield can be done quickly and easily, it's going to generate far more revenue and as a side effect, everyone passing by can see it, too.

UC Davis Medical Center said: *"Our parking and Transportation department has been using* ***The Barnacle*** *for just over a year. It's lightweight, easy to maintain and store. Our parking enforcement team loves the fact that this device can be deployed in under 5 minutes. It has certainly captured the attention of scofflaws and on lookers. The Barnacle enabled our department to recoup over* ***$40,000*** *in outstanding parking citations!"*

The Cheyenne Police said: *"Thanks to **The Barnacle**, we increased collections by $30,000. In addition, people see The Barnacle on a vehicle which prompts many to pay their own outstanding tickets so they don't get Barnacled. The presence of The Barnacle is also causing more to abide by city parking rules in the first place, and we've written 1,000 fewer tickets. We collect more parking fines and write fewer tickets. It's a win-win for the City of Cheyenne."*

https://www.barnacleparking.com/

YouTube has a lot more for anyone who's interested. I thought it was a very clever use of technology. The device itself does not need any cellular radio. It just needs an inexpensive GPS receiver to detect whether the driver is somehow moving their car. Other wise, it just uses a combination unlock on a keypad and the driver who removes the device is responsible for its return and recycling in the facility where it was attached. I think it's genius.

Aeon 14
Last week I mentioned having finally landed upon a series of military hard science fiction that I was really liking. A couple of days later I decided that I needed to share my find with John Slanina (aka TWiT's Jammer B), Here's what I wrote to John:

John...

After having real trouble putting the 4th book in the LONG series down in order to sleep (or do anything else other than consume it) I'll be telling the podcast audience about it next week. But I wanted to tell you what I've found.

Being the voracious Sci-Fi reader that you are, you may have already run across the author "M.D. Cooper" (Malorie Cooper) who has created an extensive future universe for humankind under the banner of "Aeon 14."

In terms of the overall timeline, where I began (with "Outsystem") was not the earliest in the timeline -- but it is the first book Malorie wrote, so she and her co-authors went back and filled-in the earlier timeline later. Here's what she explained:

Whether you're a new reader, or you have a hundred titles under your belt, one thing is for sure... Aeon 14 is huge. Hi, my name is Malorie Cooper (M. D. Cooper) and I created Aeon 14 as an imagining of a possible future history for humanity. The story covers hundreds of characters, sweeping across over ten thousand years. If you're just getting started, I recommend beginning where I did, with Outsystem. You can also check out the primer on the 42nd century to get yourself acquainted with the setting.

https://aeon14.com/books/outsystem

Everything is "Kindle Unlimited" and it's possible to grab a single "Omnibus" that contains the first 4 books in a single download:

https://amazon.com/dp/B01HWU85LQ

Oddly, this "The Intrepid Saga" includes a 4th book. It's the next book in the next series and *this is the book that really hooked me. Things have really come together.*

The follow-on books can be found here:

https://aeon14.com/books/the-lost-colony-ship-(destiny-lost)

And on Amazon: https://amazon.com/dp/B075RZZX5P

I finished book #4 and I'm into the next one. And it has not disappointed. John replied that the timing of my recommendation was perfect because he had just finished the book for Stacy's book club and needed something next. I have always shared my Sci-Fi reading discoveries, and boy we have had some wonderful reading, I wanted to share this one too. I just asked John if he had any feedback. I cannot share all of what he wrote since I don't want any spoilers, but part of what he replied was:

"They've made it to Ceres and they're planning their visit to Calisto - quite enjoying it! Love the AIs some of them have in their heads... and so much cool tech"

So it's still early for him, and a LOT more ultra-cool tech is in store. But I would say that it's a hit for John, too.

This series may not be for everyone. It's more pulpy, campy, space operatic, and in some ways a lot more fun and frolicy than any of Peter Hamilton's wonderful works, and it offers a lot of action, lots of stuff gets "blowed up" and there's a never-ending stream of very cool new tech. It is by no means "high brow" – so I won't attempt to defend it on that basis – but I've read enough to be able to heartily recommend it and I'm having a lot of fun with it!

It's available under Amazon's Kindle Unlimited plan and Audible versions of the books are also available. We now know that Kindle Unlimited plan authors are remunerated based upon the page counts that are actually read... it's too bad that re-reading pages probably doesn't create a bonus since there are many passages I've enjoyed more than once.

The link to Malorie's Aeon14 web site is this week's shortcut of the week: https://grc.sc/948

But the site is easy to find, it's just www.aeon14.com ... and speaking of 14, if that's your age, you'll also probably enjoy all of her book's cover art.

What if a Bit Flipped?

We have a classic case of attacks never getting weaker, only ever getting stronger. In this case a theoretical weakness that was first posited 27 years ago, back in 1996, has been found to be practical. Blessedly rare, but still practical. What this means for us today is that the private keys protecting some SSH servers using RSA keys can leak to an entirely passive observer if, for any reason, a mistake is made during the generation of the RSA signature.

The first of two papers from back then was titled *"Memo on RSA signature generation in the presence of faults."* A ways into this paper its author writes: *"This implies that, even if there is the tiniest probability that an error occurs during RSA signature generation, the generator of an RSA-signature must make sure that each signature generated is indeed correct."*

The following year, in 1997, another paper was published titled: *"On the Importance of Eliminating Errors in Cryptographic Computations"*. The Abstract of this paper paints the picture very clearly and starkly:

We present a model for attacking various cryptographic schemes by taking advantage of random hardware faults. The model consists of a black-box containing some cryptographic secret. The box interacts with the outside world by following a cryptographic protocol. The model supposes that from time to time the box is affected by a random hardware fault causing it to output incorrect values. For example, the hardware fault flips an internal register bit at some point during the computation. We show that for many digital signature and identification schemes these incorrect outputs completely expose the secrets stored in the box. For example, the secret signing key used in an implementation of RSA is completely exposed from a single erroneous RSA signature.

So our primary takeaway to this point is the somewhat surprising news that, terrific as the RSA public key cryptosystem is, it is also quite fragile to information disclosure. If ***any*** mistake is made for ***any*** reason during the computation of an RSA signature, that mistaken signature along with any valid signature made with the same private key can be used to completely expose the server's private key. From there, this allows the server to be transparently impersonated by anyone who can arrange to place themselves in the information flow.

RedHat considered the implication of this in September of 2015 with regard to TLS connections. Their piece was titled: *"Factoring RSA Keys With TLS Perfect Forward Secrecy"* and this is what they found. They wrote:

Back in 1996, Arjen Lenstra described an attack such that if a fault occurred during the computation of a signature, an attacker might be able to recover the private key from the signature. At the time, use of cryptography on the Internet was uncommon, and even ten years later, most TLS (or HTTPS) connections were immune to this problem by design because they did not use RSA signatures. This changed gradually, when forward secrecy for TLS was recommended and introduced by many web sites. We evaluated the source code of several free software TLS implementations to see if they implement hardening against this particular side-channel attack, and discovered that it is missing in some of these implementations. In addition, we used a TLS crawler to perform TLS handshakes with servers on the Internet, and *collected evidence that this kind of hardening is still needed, and missing in some of the server implementations: We saw several RSA key leaks, where we should not have observed any at all.*

This issue has been quietly lurking in the backs of the minds of cryptographers for decades, which brings us to today and the recently discovered and confirmed vulnerability of many SSH servers.

A group of four researchers from UC San Diego at La Jolla have just published their research under the title: *"Passive SSH Key Compromise via Lattices"* which will be presented two weeks from now during the upcoming ACM SIGSAC Conference on Computer and Communications Security (CCS 2023), this November 26–30, in Copenhagen, Denmark.

We demonstrate that a passive network attacker can opportunistically obtain private RSA host keys from an SSH server that experiences a naturally arising fault during signature computation. *In prior work, this was not believed to be possible for the SSH protocol because the signature included information like the shared Diffie-Hellman secret that would not be available to a passive network observer. We show that for the signature parameters commonly in use for SSH, there is an efficient lattice attack to recover the private key in case of a signature fault. We provide a security analysis of the SSH, IKEv1, and IKEv2 protocols in this scenario,* ***and use our attack to discover hundreds of compromised keys in the wild from several independently vulnerable implementations.***

The best way to get a better feeling for what's going on here is to look at how they summarized their overall findings. They wrote:

RSA digital signatures can reveal a signer's secret key if a computational or hardware fault occurs during signing with an unprotected implementation using the Chinese Remainder Theorem and a deterministic padding scheme. This attack requires only a single faulty signature, the public key, and a single GCD (greatest common divisor) computation, and it has been exploited extensively in the cryptographic side channel literature on active fault attacks.

In a 2015 technical report, Weimer observed that this same vulnerability could be exploited in the context of TLS by an attacker without physical access to a machine, simply by connecting to machines and waiting for a fault to occur during computation. He traced several of the failures he observed to failing hardware.

In 2022, Sullivan et al. observed that this flaw remained exploitable on the open internet, and used passive network measurement to compute TLS private keys from vulnerable implementations that appeared to experience hardware failure.

However, this vulnerability was not believed to be realistically exploitable in the context of other popular network protocols like IPsec or SSH because the signature hash includes a Diffie-Hellman shared secret that a passive eavesdropper would be unable to compute, thus ruling out the single-signature GCD attack. Because a passive adversary can typically collect significantly more data than an active adversary who must participate in every Diffie-Hellman exchange, this belief represented a significant underestimate of the cryptanalytic capabilities of such passive adversaries against SSH and IPsec compared to TLSv1.2.

In this paper, we show that passive RSA key recovery from a single faulty signature is possible in the SSH and IPsec protocols using a lattice attack. In this context, a passive adversary can quietly monitor legitimate connections without risking detection until they observe a faulty signature that exposes the private key. The attacker can then actively and undetectably impersonate the compromised host to intercept sensitive data. We cast the key recovery problem as a variant of the partial approximate common divisor problem, and we show that this problem is efficient to solve for the key sizes and hash functions used for SSH and IPsec.

We then carry out internet-wide scans for SSH and IPsec to measure the prevalence of vulnerable signatures in the wild. We find multiple vulnerable implementations that appear to be due to different classes of hardware flaws. We also carry out a retrospective analysis of historical SSH scan data collected over the course of seven years, and find that these invalid signatures and vulnerable devices are surprisingly common over time. Our combined dataset of around 5.2 billion SSH records contained more than 590,000 invalid RSA signatures. We used our lattice attack to find that more than 4,900 revealed the factorization of the corresponding RSA public key, giving us the private keys to 189 unique RSA public keys. We also analyze passively collected SSH network data. In addition to the signature vulnerabilities we were searching for, our analysis gives us a window into the state of the SSH, IKEv1, and IKEv2 deployment landscape. We observed a number of vulnerable and non-conformant behaviors among IPsec hosts in particular.

So, by passively examining SSH handshake traffic – and even recordings of previous handshake traffic – they were able to obtain the secret private keys belonging to 189 different devices. The other thing that had to say that was interesting was what they learned about the SSH devices which were occasionally generating invalid and thus vulnerable signatures. They wrote:

Our research identified four manufacturers of devices susceptible to this key recovery attack. We disclosed the issue to Cisco on February 7, 2023 and to Zyxel on March 1, 2023. Both teams investigated promptly, although limitations in the historical scan data made it challenging to identify the software versions that generated the vulnerable signatures and reproduce the issue. Cisco concluded that Cisco ASA and FTD Software had introduced a suitable mitigation in 2022, and was investigating mitigations for Cisco IOS and IOS XE Software.

Zyxel concluded that the issue had affected ZLD firmware versions V3.30 and earlier, which have been end-of-life for years. By the time of our disclosure, the ZLD firmware had begun using OpenSSL, which mitigates this issue.

Our attempts to contact Hillstone Networks and Mocana were unsuccessful, and we submitted the issue to the CERT Coordination Center on May 18, 2023 for assistance with disclosure. We received no additional information from CERT/CC during the 45-day disclosure period.

We considered notifying operators of affected devices whose keys we had recovered, but we determined this would be infeasible. Even after combining publicly available data with the historical scanning data, we were unable to determine which organization was responsible for a particular device, whether the device was still in use today, or up-to-date contact information for the device's current operator. We also lacked practical and actionable advice for owners of affected devices until the manufacturers completed their investigation.

So now we knew that communication protocols using RSA signature generation which have not been explicitly hardened against the possibility of mistaken signature computation are prone to having their private keys stolen if anyone is watching and paying attention. And in today's world it would be difficult to imagine that with papers about this dating back to 1996, 1997 and 2015, the NSA might not have already done the math and may have been keeping quiet about this in case it might have come in handy. We don't know what we don't know, which is why I think that this sort of academic poking at the protocols we have in use can be so important.

I titled this podcast: *"What if a Bit Flipped?"* And now we know what if a bit flipped. What we still don't know is WHY that bit might have flipped. The literature doesn't have a lot to say about it, but it boils down to cosmic rays just happening to hit a microscopic transistor at the right instant to cause an unrepeatable event. Or a system's power supply at the beginning of its death spiral. I had this happen to me recently with my NetGate SG-1100 router. On the surface, when it was working it was working. But was it really? What if I'd been using it as an SSH endpoint. Might it have glitched at just the wrong moment and caused a signature mistake?

I think that the best takeaway is that we all sometimes see our computers do something weird and inexplicable. Since software bugs are clearly by far the most common cause, we would rarely think that we just experienced a cosmic ray event. But who knows? And the cause of a mistaken signature computation doesn't have to be hardware. A device driver might not have properly restored the machine's registers following a hardware interrupt that happened to occur right in the middle of that critical signature computation. Not that long ago we tracked down weird one-off problems in SpinRite to exactly that cause. After I wrapped SpinRite in protection we have never had another such error.

The best news is that the crypto industry has learned that some cryptographic processes are highly sensitive to mistakes occurring from any source, even cosmic background radiation. And the systems that were late to protecting themselves have been identified. Getting them fixed in the field may never happen. But new systems won't be shipping with those cosmic ray vulnerabilities.