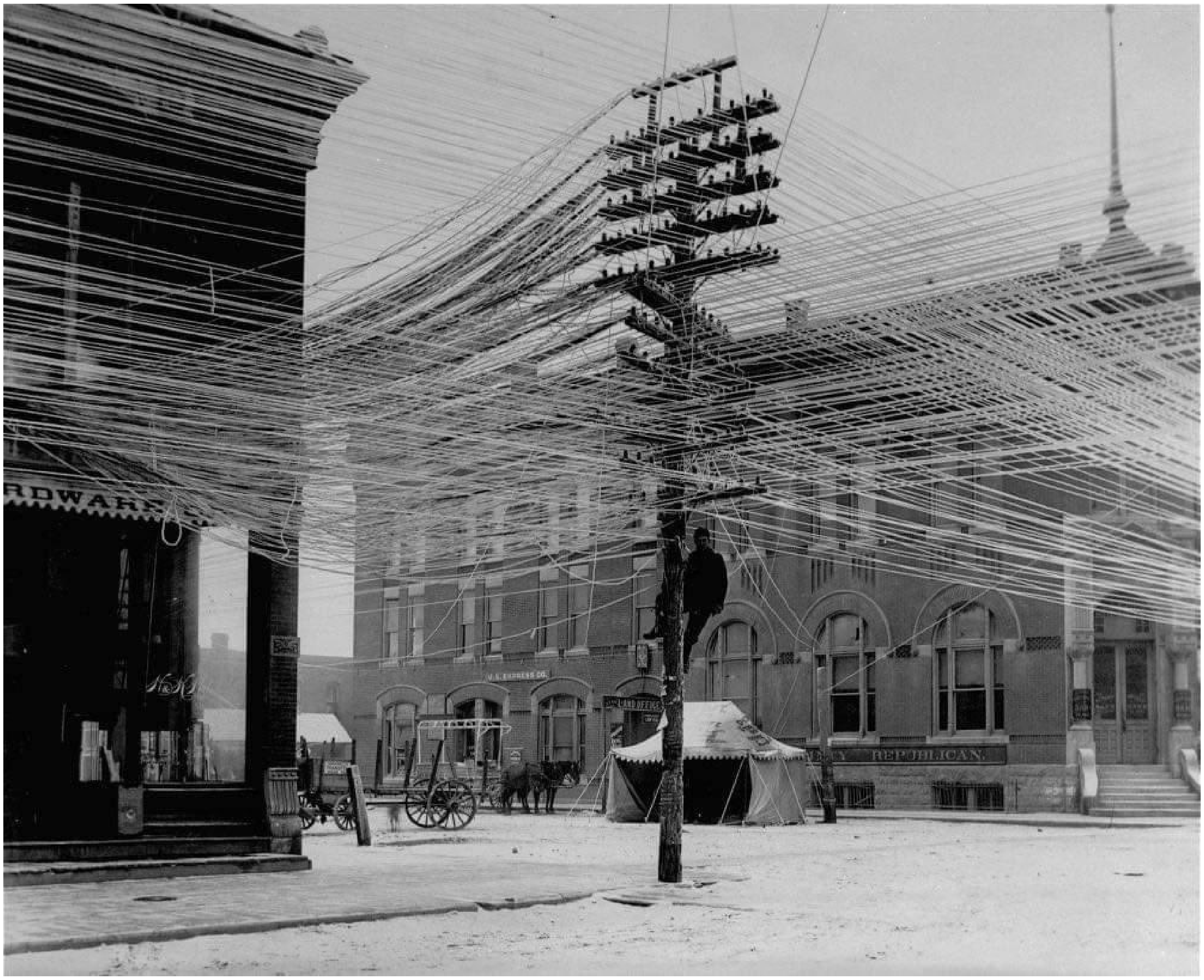
Security Now! #831 - 08-10-21
Apple's CSAM Mistake

This week on Security Now!

This week we look at a pervasive failure built into the random number generators of a great many, if not nearly all, lightweight IoT devices. We look at some old, new and returned critical vulnerabilities in major VPN products. And we encounter 14 fatal flaws in a widely used embedded TCP/IP stack. We look at a number of terrific bits of feedback from our listeners. Then we carefully examine the operation and consequences of Apple's recent announcement of their plan to begin reacting to the photographic image content being sent, received and stored by their iOS-based devices.

The more things change, the more they stay the same...



TomGerald / @TomGerald: @SGgrc as a follow up to last week's picture.

Security News

“You're Doing IoT RNG”

In my favorite talk title of this summer’s DEF CON / BLACKHAT 2021, a pair of researchers from Bishop Fox Labs titled their talk: “You're Doing IoT RNG”. They dug into the source of entropy being used by some of today's most popular IoT platforms. And to say that they found it wanting, would be an understatement. To get their audience's attention they began by noting:

“There’s a crack in the foundation of Internet of Things (IoT) security, one that affects 35 billion devices worldwide. Basically, every IoT device with a hardware random number generator (RNG) contains a serious vulnerability whereby it fails to properly generate random numbers, which undermines security for any upstream use.”

Many years ago on this podcast, when we were laying the foundation of cryptography and the requirements for securely encrypting both data at rest and data in flight—which is to say, communications—we explained the importance of a high quality source of entropy. An example is the often used Diffie Helman key agreement system where each side picks a number at random, turns it into a key by performing some crypto math on it, then sends it to the other side. They exchange these keys. The cool thing about this, which makes it so valuable for securing communications, is that each side is able to take their originally chosen random number, along with the key they received from the other party, and arrive at a new key. And each end will arrive at that same new key. But the most critical aspect of this is that anyone eavesdropping on that initial setup conversation who is able to observe and obtain the keys that each side sends to the other, cannot recreate the key that the ends both share. This allows the ends to then start using that key they now share for truly private communications.

What was the first thing each side did? They chose a random number. And it's crucial to the security of the entire process that each side's chosen random number cannot be known or guessed by anyone else. It must be truly random. But it turns out that it's much easier to specify what we want from a random number than it is for a little computer sitting inside a light switch or webcam to actually generate such a number. A random binary number is any string of bits of some specified length, where the probability of each bit being a 0 or a 1 is exactly 50/50 and that each bit is chosen completely independently of any other of the random bits. So that seems easy to ask for. But software won't do that. It won't. Software is brutally deterministic. No matter how much you add, subtract, multiply and divide, if you start from the same starting point you're always going to arrive at the same ending point.

In order to break this deterministic cycle we need some external source of unpredictability. A little so-called system-on-a-chip (SoC) probably has a radio for WiFi. So, if it was clever it might hash the WiFi packets it's able to sniff buzzing around it, whether they're meant for it or not. And if it was extra clever it might even tune its radio to different channels, or maybe listen to the noise being generated by the cosmic background radiation—or what we commonly refer to as "static". It might periodically sample the value of its hashing system, dumping the results into a big pool of entropy. Then a CSPRNG—a cryptographically strong pseudo random number generator—would take the pool's contents and use it for that system's keys.

But that's not the way things turned out. For one thing, many of these SoC's are very resource constrained. They don't have enough extra RAM to even create a sufficiently large pool of entropy. So the designers of the hardware said "we can do this in hardware" and built-in their own source of entropy noise. One of the fundamental components of electronics is a diode. A diode uses the properties of semiconduction to only allow an electric current to flow through it in one direction. It strongly resists any current flow in the opposite direction. But there's a limit, known as the diode's breakdown voltage, above which a diode will start to leak in its reverse direction. And it turns out that as individual electrons valiantly tunnel their way through the diode to emerge victorious on the other side, this breakdown current is quite noisy. It doesn't have any sort of perfect 50/50 property. But it is utterly unpredictable when any given electron is going to emerge from the reverse-biased diode. So it can serve as a starting point for the SoC's full built-in hardware random number generator which will still need to do a lot of work on those random events to make the result satisfy that simple definition we began with.

And this brings us to a crucial limitation of any hardware random number generator which is operating in a resource constrained environment: It may be able to give you random bytes of data, but not necessarily at the rate that you might want or need for your application. Software generators can give you as much as you can take. But it won't be random. Hardware can generate true randomness, but not very quickly. And this is why, in mature well-designed systems, the two techniques are often combined. The hardware seeds and periodically re-seeds a high quality software pseudo random number generator to give us the best of both techniques.

But it turns out that we're back to that resource constraint problem again and IoT hardware has taken the shortcut of using a pure hardware generator with a limited rate of entropy generation.

Writing about the need for IoT devices to have entropy sources, Dan and Allan explain that "As of 2021, most IoT systems-on-a-chip (SoCs) have a dedicated hardware RNG peripheral that's designed to solve exactly this problem. But unfortunately, it's not that simple. How you use the peripheral is critically important, and the current state of the art in IoT can only be aptly described as "doing it wrong."

Get a load of this: The researchers found example after example where the code that was calling the hardware's built-in random number generator API always assumed that the API call succeeded in providing them with the valid random data they had requested and never bothered to check for the error status returned from the API stating that the request could not be met at this time.

Now, if you think about it, the really tricky thing about this is that if you're asking for random data, ANYTHING that is returned is valid, right? I mean, as unlikely as it might be, getting all 0's back from a request for a completely random number would not be, itself, an error, since while that would have been incredibly unlikely, it's possible.

If you don't deliberately check the success or failure status of a request for random data, you'll still get data, but it might not be at all random. A Github search for this mistake being made in IoT code based upon the MediaTek 7697 SoC HAL returned 3,218 hits. And a similar Github search for mistaken use of the hugely popular FreeRTOS IoT operating system turned up 36,696 instances of improper use. As the authors wrote in their write-up where they show C-code snippets:

“Notice that the return code is pervasively not checked – though this isn’t unique to these two examples. This is just how the IoT industry does it. You’ll find this behavior across basically every SDK and IoT OS.”

So as not to put any words in their mouth, I'm going to quote from their write-up under “What's the worst that could happen?”

Okay, so devices aren’t checking the error code of the RNG HAL function. But how bad is it really? It depends on the specific device, but potentially bad. Very bad. Let’s take a look.

The HAL function to the RNG peripheral can fail for a variety of reasons, but by far the most common (and exploitable) is that the device has run out of entropy. Hardware RNG peripherals pull entropy out of the universe through a variety of means (such as analog sensors or EMF readings) but don’t have it in infinite supply. They’re only capable of producing so many random bits per second. If you try calling the RNG HAL function when it doesn’t have any random numbers to give you, it will fail and return an error code. Thus, if the device tries to get too many random numbers too quickly, the calls will begin to fail.

But that’s the thing about random numbers; it’s not enough to just have one. When a device needs to generate a new 2048-bit private key, as a conservative example, it will call the RNG HAL function over and over in a loop. This starts to seriously tax the hardware’s ability to keep up, and in practice, they often can’t. The first few calls may succeed, but they will typically start to cause errors quickly.

So… what does the HAL function actually give you for a random number when it fails? Depending on the hardware, one of the following:

- Partial entropy
- The number 0
- Uninitialized memory

As a consequence of this, the feeling of security we have is illusory. Oh yes, TLS. Yay! But if the TLS handshake is based upon all-0 or static unchanging keys, then TLS is just adding a bunch of connection overhead and not providing any true privacy.

As I've noted before, the entire world has rushed headlong into IoT without any standards, nor anything like an Underwriters Laboratories for device security. The outside of the box says "uses the most advanced military grade encryption"... but fails to mention that it also uses null crypto keys. Whoops. Think that might matter?

The Pulse Secure VPN remains in trouble

As we know, abuse of VPNs and Microsoft's Remote Desktop Protocol (RDP) are currently two of the most popular means for hackers to get into an enterprise's network. It's a problem when valid username and password login credentials are obtained. As we know, these days such credentials may find a ready market depending upon the value of the company they're protecting. So this allows for targeted attacks. But it's much worse when the product itself has exploitable vulnerabilities because then all users of the affected product will be open to exploitation until an update has been made available and installed.

The Pulse Secure VPN has been in the news all year due to continuing problems with vulnerabilities which are believed to have been leveraged as the way hackers conducted a number of recent attacks. The Pulse Secure VPN is back in the news today because it's trying again to fix a critical flaw that it first tried to fix last October. The trouble is a critical post-authentication remote code execution (RCE) vulnerability which exists in their “Connect Secure” VPN appliances. This flaw, CVE-2020-8260, was one of the four Pulse Secure flaws that were being actively exploited by bad guys in April in a series of intrusions targeting defense, government, and financial entities in the U.S. and beyond. Given the proven real-world exploitation, it's strongly recommended that anyone using Pulse's Connect Secure (PCS) 9.1R12 or later should be certain to be current with all relevant updates.

Richard Warren with the NCC Group said on Friday: “The Pulse Connect Secure appliance suffers from an uncontrolled archive extraction vulnerability which allows an attacker to overwrite arbitrary files, resulting in Remote Code Execution as root. This vulnerability is a bypass of the patch for CVE-2020-8260 [from last October]. An attacker obtaining such access will be able to circumvent any restrictions enforced via the web application, as well as remount the filesystem, allowing them to create a persistent backdoor, extract and decrypt credentials, compromise VPN clients, or pivot into the internal network.”

This all occurred after Ivanti, Pulse Secure's publisher, published an advisory for six security vulnerabilities on Monday, August 2nd. At that time Ivanti urged their customers to move quickly to update to version 9.1R12 to secure against any exploitation of those flaws.

And Cisco, too...
Ivanti is not alone with their VPN troubles. Last Wednesday, Cisco released patches for two serious flaws affecting many of their VPN products. The two security flaws are CVE-2021-1609 — which has a CVSS of 9.8/10 — CVE-2021-1602 — with CVSS of 8.2/10. They were discovered in the web-based management interfaces and are the result of improperly validated HTTP requests and insufficient user input validation. And, being in the web management interface they result in pre-authentication security vulnerabilities impacting multiple Small Business VPN routers and allowing remote attackers to trigger a denial of service condition or execute commands and arbitrary code on vulnerable devices.

The first flaw impacts RV340, RV340W, RV345, and RV345P Dual WAN Gigabit VPN routers.
The second flaw impacts RV160, RV160W, RV260, RV260P, and RV260W VPN routers.

When the Internet-facing remote management interface is enabled, both bugs are exploitable remotely without requiring authentication as part of low complexity attacks that don't require user interaction. Attackers could exploit the vulnerabilities by sending maliciously crafted HTTP requests to the affected routers' web-based management interfaces. The good news is, those remote management interfaces are disabled by default on all of those routers. On the other hand, Cisco said that the web-based management interface for these devices is available through local LAN connections by default and cannot be disabled there. So if a malicious attacker were to briefly gain access to the internal network they might be able to exploit this vulnerability on the LAN side web interface to, for example, create a persistent VPN account for themselves that might go unnoticed until it was needed.

Cisco has released software updates to address these vulnerabilities and says no workarounds are available to remove the attack vectors. Anyone having any of these Cisco VPN models should immediately, at the very least, verify that WAN-side web management is disabled. It's under “Basic Settings” > “Remote Management.” You'll want to be sure it's set to OFF!

Flaws found in another popular embedded TCP/IP library
14 newly disclosed vulnerabilities, collectively referred to as INFRA:HALT were discovered by the work of a joint research effort by security teams at Forescout and JFrog. These vulnerabilities impact an extremely popular TCP/IP library which is commonly used in industrial equipment and Operational Technology (OT) devices manufactured by more than 200 vendors.

NicheStack is a proprietary TCP/IP stack developed originally by InterNiche Technologies and acquired by HCC Embedded in 2016. The earliest copyright messages indicate that the stack was created in 1996, although InterNiche was founded in 1989. The stack was extended to support IPv6 in 2003. In the last two decades, the stack was distributed in several “flavors” by OEMs such as STMicroelectronics, Freescale (NXP), Altera (Intel) and Microchip for use with several (real-time) operating systems or its own simple RTOS called NicheTask. It also served as the basis for other TCP/IP stacks.

Operational Technology or OT is not a term we've used before. NIST defines it as: “Programmable systems or devices that interact with the physical environment (or manage devices that interact with the physical environment). These systems or devices detect or cause a direct change through the monitoring or control of devices, processes, and events. Examples include industrial control systems, building management systems, fire control systems, and physical access control mechanisms.” So what has traditionally been referred to as SCADA (Supervisory Control And Data Acquisition) has now been broadened and formalized into “Operation Technology” or OT for short.

The point is, these OT systems are at the heart of industrial control and monitoring. They are increasingly networked. And now we learn that more than 200 vendors of these gadgets have all been relying upon this “NicheStack” which provides a library of now known-to-be-vulnerable TCP/IP networking functions. Forescout and JFrog have collectively named these vulnerabilities INFRA:HALT because they allow for remote code execution, denial of service, information leakage, TCP spoofing and DNS cache poisoning. These are not features we want in the networked critical infrastructure monitoring and management devices being produced by more than 200 different vendors.

And unlike our personal PCs and smartphones, these random faceless boxes buried behind crates and in the closed and locked closets of industrial manufacturing facilities, are not accustomed to be being updated regularly, if at all. If ever. The disclosing researchers explained that “the nature of these vulnerabilities could lead to heightened risk and expose national critical infrastructure at a time when the industry is seeing an increase in OT attacks against global utilities, oil and gas pipeline operators as well as healthcare and the supply chain.”

All versions of NicheStack prior to v4.3 (the latest at the time of research), including NicheLite, are affected. The patches released by HCC Embedded are available upon request. Rows are colored according to the CVSS score: yellow for medium or high and red for critical.

In the DNSv4 component the researchers found that "The routine for parsing DNS responses does not check the "response data length" field of individual DNS answers, which may cause OOB-R/W." This received a difficult-to-obtain CVSS score of 9.8.

In the HTTP module, "A heap buffer overflow exists in the code that parses the HTTP POST request due to lack of size validation." Okay, that's just difficult to excuse. That's SO basic. And it's worth a CVSS of 9.1.

Also in the DNSv4 code, "The routine for parsing DNS domain names does not check whether a compression pointer points within the bounds of a packet, which leads to OOB-R." I've written a bunch of DNS parsing code since I have the DNS Benchmark and the DNS spoofability system. So I know it pretty well. The guys who designed the DNS on-the-wire format were clever. They realized that DNS packets would naturally contain a lot of redundancy. For example, when you get a DNS answer it contains the DNS query too. So a single DNS packet might have answers for grc.com, www.grc.com, forums.grc.com, sqrl.grc.com. Instead of wasting space for all of those redundant "grc.com's", the DNS packet definition allows a DNS name prefix and then a pointer to the rest of the DNS name. Since using a short pointer to point to a longer string compresses the size of the packet, it's referred to as a compression pointer. So, in this example the string "grc.com" would only occur once in the DNS packet and everything else would contain a pointer to that one instance rather than repeating it. So now we learn that this widespread and pervasively used TCP/IP stack's DNS parser "does not check whether a compression pointer points within the bounds of a packet." Again, how dumb is that? It's just unconscionable.

And everyone's using this code because, why not. It works, don't it?

But wait, there's more: "The routine for parsing DNS responses does not check whether the number of queries/responses specified in the packet's header corresponds to the query/response data available in the DNS packet, leading to OOB-R." I don't think we've probably ever encountered on this podcast a clearer example of code that was shipped because it worked with no apparent thought whatsoever—not even one heartbeat—given to that code's security.

And there's 10 more vulnerabilities just like those.

If the earliest copyright carried by the stack is 1996 the code is 25 years old. So I could cut it some slack. Afterall, the Internet's original RFC's don't ever mention security at all. They specify that a compression pointer will point to somewhere within the packet. It only makes sense and will only work if it does. But it's certainly insecure if it's allowed to point anywhere it wants to. That's just nuts in today's world.

The researchers found a legacy website listing the main customers of InterNiche. According to the website, most of the top industrial automation companies in the world use the stack. And in addition to those, the website mentions another nearly 200 device vendors.

So the researchers queried Shodan looking for devices showing some evidence of NicheStack, for example, application-layer welcome banners. On March 8th of this year they found more than 6400 devices NicheStack. Of those devices, the large majority (6360) run an HTTP server while the others run FTP, SSH or Telnet servers.

Talk about a target-rich environment. It's good that this research was done. Now the NicheStack has been significantly cleaned up. So all new code built using it will be much better than ever before. But a massive amount of damage has already been done over the course of the past 25 years. There are countless "things" out in the world containing that stack, even if they aren't exposed to the public Internet. Conscientious state actors will have added these datums into their comprehensive device vulnerability databases for the day when they have a need to penetrate, or screw with, any of those millions of vulnerable gadgets having an old NicheStack connected to the network.

Browser News

Microsoft Edge gets "Super Duper Secure Mode"

Yes, that's actually what they're calling it because all of the serious names were apparently taken. Actually, they realize full well that this is a humorous name and they do plan to find sometime more befitting its seriousness if it takes hold.

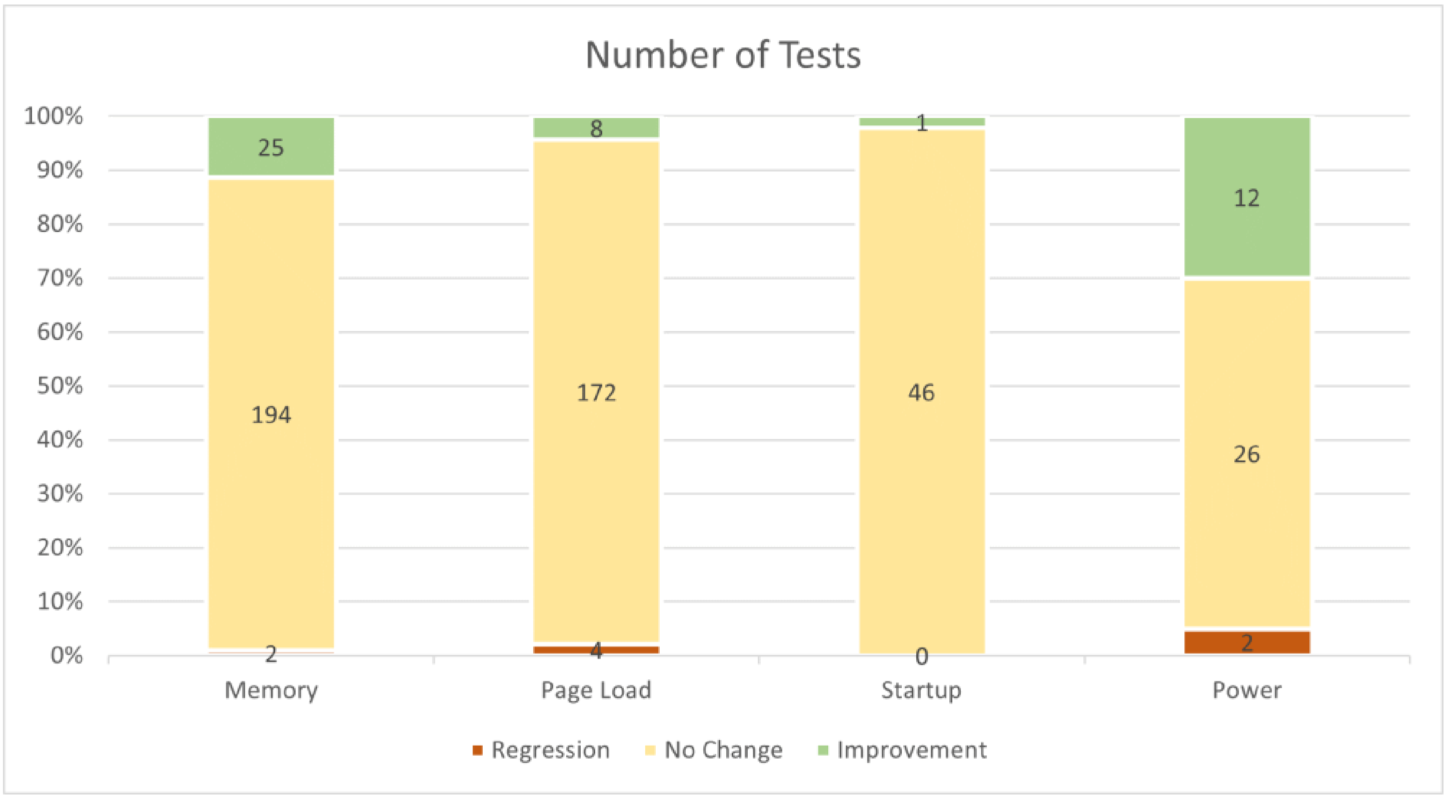
So here's the sixty-four thousand dollar question which Microsoft's experimental Edge browser "Super Duper Secure Mode" asks: Would you be willing to sacrafice some web browser performance in return for potentially significantly greater security?

This is really interesting. Based upon an analysis of all Chrome/Chromium browser CVE's issued since 2019, 45% of the vulnerabilities appearing in the Chromium project's V8 JavaScript and WebAssembly engine were related to its Just-In-Time (JIT) component. And over half of ALL of Chrome's bugs found being exploited "in the wild" are abusing JIT bugs.

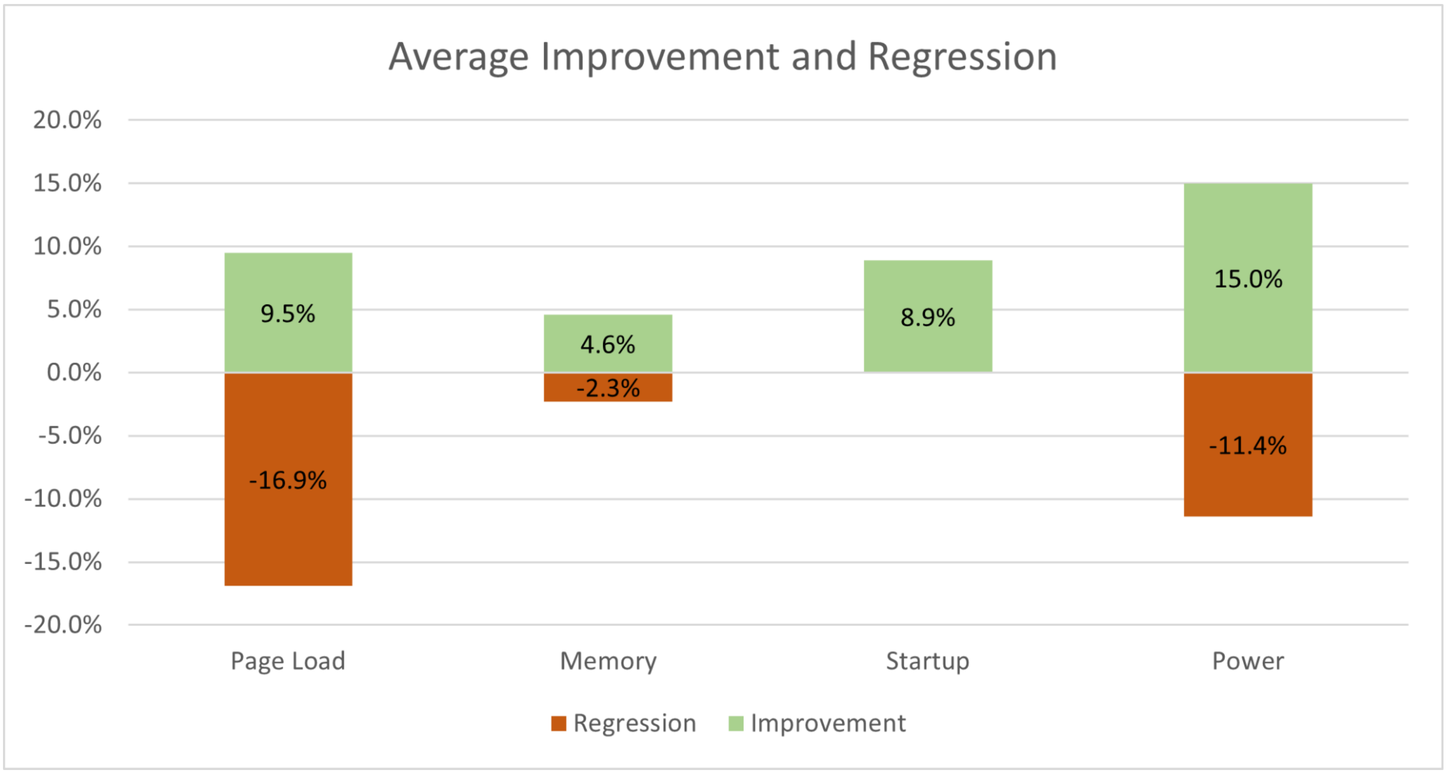
This led Microsoft's Edge Vulnerability Research team to wonder what the impact would be of just saying no to Chromium's JIT engine. And the result is Edge's new experimental "Super Duper Secure Mode." Johnathan Norman, Microsoft Edge's Vulnerability Research team Lead observed that "This reduction of attack surface has potential to significantly improve user security; it would remove roughly half of the V8 bugs that must be fixed. This reduction in attack surface kills half of the bugs we see in exploits, and the bugs it doesn't kill become more difficult to exploit. To put it another way, we lower costs for users but increase costs for attackers."

And Microsoft acknowledges that this challenges some conventional assumptions held by many in the browser community. Since the JIT engine is a Just In Time compiler implemented to optimize performance, the immediate question that arises is "Okay, so more security. But at what cost in performance?" That's really the question for us to answer. Johnathan said that recent tests carried out by the Edge team have shown that despite its pivotal role in speeding up browsers in the early and mid-2010s, JIT no longer appears to be a crucial feature for Edge's performance.

I think that's some very interesting out-of-the-box thinking. And it makes sense. Our systems' processors have become so much more capable over the past 10 to 15 years that it's reasonable to ask whether our browser's what have grown so fast that only a modest cost in performance could buy us significantly greater security.



Microsoft said of the diagram above that most tests see no changes with JIT disabled. There are a few improvements and regressions, but most tests remain unchanged. And they said that anecdotally they found that users running with JIT disabled rarely noticed any difference in their daily browsing behavior or speed. But how much variation did they see in the tests that did so change? The chart below shows the average percentage improvement or regression in performance.



Closing the Loop

BobSouthwell / @bobsouthwell

The 'Bobiverse'? Oh yes!!!!! But listen to the audiobooks, The voice characterizations are wonderful. Very geeky cultural references!.

BobSouthwell @bobsouthwell — 12 Jun 2018

Last lines from Netflix movie "Anon": "I don't have anything to hide. I just don't have anything I want you to see." Brilliant summary of the whole privacy issue.

On TailScale:

Deacon D / @SoCalVistas
Hi Steve, I just want to send a BIG THANK YOU! For turning me on to Tailscale. Wow!!! I have it running on 2 Synology NAS', 2 Windows boxes, and my Android phone. It's unbelievably easy, and like they say, it just works. :-)

Matt Vest / @mvest20
@SGgrc Thank you so much for bringing @Tailscale to my attention! I just set it up - it took about 15 minutes to get it working the way I want, no firewall or port configuration effort, and it works flawlessly. Exactly what I've been looking for.
James P (he/him) / @jpancoast
.@SGgrc Setting up @Tailscale was so damn easy it was scary.

Marko Simo / @m_simo
Many thanks to @SGgrc for letting us SN listeners to know about @Tailscale. I have now replaced my OpenVPN with it and it blows my mind. I think this must be the future of private networking, not just virtual, but ALL private networking. 🙌

Philip Hofstetter / @pilif
@SGgrc re the hosts file entry of you eCommerce provider: You talked about adding that entry on the show back in 2016 ?? https://www.grc.com/sn/sn-583.htm

October 25th, 2016 — We were covering a major DNS outage I wrote:

My own intersection with last week's problem was when I received, at about 7:30 in the morning, an iMessage from Sue, my bookkeeper. She had been away from the office for about a week, traveling with her laptop and checking-in constantly to deal with any sales grc-related mail stuff. And she sent me a note saying that Eudora - yes, we're all still using Eudora - was returning "GRC.com address not resolved" error. And I thought, whoa, that's odd. And so I shot her a text note back and said that doesn't really sound like our problem, but I'll look into it.

And then, like maybe two hours later, I got an alert saying that one of GRC's eCommerce transactions had failed. So then I started seeing the news about a major DNS-related outage. And that put all the pieces together for me. That explained why Sue, wherever she was, whatever DNS server her location was using, was unable to obtain the IP for GRC. And suddenly I thought, ah, I'll bet that's what's happening because of the coincidence that GRC's eCommerce system was unable to obtain the IP for the merchant gateway. But I was able to get it from here as a Cox cable subscriber.

So I looked up the IP, jumped over to GRC's server to drop an entry into the hosts file. And what I found, interestingly, was I had already commented out the line I was going to put in. In other words, this had happened previously. So all I did was remove the pound sign from that line because the IP had not changed from whenever it was I had done that before. And then immediately eCommerce transactions started to process again.

(This was the large DYN DNS outage.)

Apple's CSAM Mistake

"CSAM Detection" vs "Communication safety in Messages"
There are two completely different systems here and because they've been announced together they are being muddled and confused. There's one system called "CSAM Detection" and an entirely different system which Apple refers to as "Communication safety in Messages". I'm going to first briefly describe each of the two systems, then Leo, let's talk about the various social controversies surrounding them.

Okay. The intent of Apple's "CSAM Detection" system is expressly and only for keeping Child Sexual Abuse Material — C.S.A.M. — out of Apple's iCloud facility. This is done by cross checking the image hashes of any image bound for iCloud against an archive of known illegal and child abusive material. Since the user's iOS device encrypts images before they're uploaded to iCloud, this image hashing and known image archive comparison must all occur before the encryption, in other words, on the user's device.

Some people have freaked out, wrongly believing that Apple will be loading the abusive images onto everyone's phones for comparison. But on this podcast we understand what it means to "hash" something. It's an extremely information lossly process that creates a fingerprint of something. In this case it's a fingerprint of a known illegal and abusive image. But in no way is it the image itself. Apple calls this form of image hashing "NeuralHash" and describes it this way:

The hashing technology, called NeuralHash, analyzes an image and converts it to a unique number specific to that image. Only another image that appears nearly identical can produce the same number; for example, images that differ in size or transcoded quality will still have the same NeuralHash value.

So, here are the properties of the system as Apple describes them:

"CSAM Detection" enables Apple to accurately identify and report iCloud users who store known Child Sexual Abuse Material (CSAM) in their iCloud Photos accounts. Apple servers flag accounts exceeding a threshold number of images that match a known database of CSAM image hashes so that Apple can provide relevant information to the National Center for Missing and Exploited Children (NCMEC). This process is secure, and is expressly designed to preserve user privacy.

CSAM Detection provides these privacy and security assurances:

- *Apple does not learn anything about images that do not match the known CSAM database.*
- *Apple can't access metadata or visual derivatives for matched CSAM images until a threshold of matches is exceeded for an iCloud Photos account.*
- *The risk of the system incorrectly flagging an account is extremely low. In addition, Apple manually reviews all reports made to NCMEC to ensure reporting accuracy.*
- *Users cannot access or view the database of known CSAM images.*
- *Users cannot identify which images were flagged as CSAM by the system.*

The technical details of how Apple pulls this off are very cool and they are not at all controversial. But this week, Leo, I want to discuss the sociological controversies surrounding Apple's stated plans. For now, we're going to assume simply that Apple is able to do this and to provide the various blinding guarantees that they claim. I'm sure they are. And once I have dug into them and understand how they're pulling this off, if it still seems relevant next week, we'll do a technical deep dive into the technology.

Okay. So that was "CSAM Detection." The other completely separate new feature is known as "Communication safety in Messages":

Communication safety in Messages gives parents and children new tools to help protect children from sending or receiving sexually explicit images through any Apple messaging. It's a pre- and post- message image filter. It operates only on images sent or received over a messaging channel and only for child accounts set up by parents in Family Sharing. Anthony Weiner's sexting would not have set off this system—but only because he wasn't a minor at the time. The new system performs the image analysis using algorithms on the device itself, pre- or post-encryption, so it does not change the privacy assurances of Messages. When a child's account sends or receives an image, which the iOS device detects and believes to be sexually explicit, the image will be blurred and the child will be warned that their device believes that the image might be inappropriate. And the child will be reassured that it is okay if they do not want to view or to send the photo. So this will dissuade children from sending images that the device believes to be sexually explicit, and it will warn children on the receiving end of such images before they are seen. As an additional precaution, younger children of age 12 or below can also be told that, to make sure they are safe, their parents will get a message if they DO choose to proceed to view or send the image. Older children of ages 13 to 17 will be warned, but no parental notification will occur.

And this entire system is only available for accounts set up as families in iCloud and the parent or guardian must opt-in to enable the feature for their family group. And as I noted above, parental notifications can only be enabled by parents or guardians for child accounts age 12 or younger.

Okay. So that's the system. What problems and concerns does this create?