Security Now! #749 - 01-14-20

Windows 7 - R. I. P.

This week on Security Now!

This week's Security Now! podcast is titled "Windows 7 R. I. P."... not because there's much that we haven't already said about the fact (we've already pretty much beaten that topic to death), but just that happens exactly **TODAY** and that, given the still massive install base of Windows 7, it remains significant that all of those machines will now be going without their clearly-needed security updates. The big news for this week **was** to be the event of the first successful chosen-prefix collision on the SHA-1 hash. But **that** news was preempted at the last minute by the much more immediately significant news of the remotely exploitable "Cable Haunt" vulnerability that's present in most of the world's cable modems... right now! So, we'll be talking about that, after we look at the FBI's recent request to have Apple unlock another terrorist's iPhone, update on the CheckRain jailbreak solution, examine the challenge of checking for illegal images while preserving privacy, look at some deeply worrying research into just how easy it is for bad guys to get SIMs swapped, examine the consequences of not patching a bad VPN flaw, deal with a bit of miscellany… and then, finally, look at the new "Cable Haunt" vulnerability. So… fasten your seatbelts, folks!

https://cablehaunt.com/

Security News

The FBI has asked Apple to unlock another iPhone

In a move that may signal another high-stakes clash over encryption, the FBI is asking Apple for help decrypting two iPhones believed to have belonged to Mohammed Saeed Alshamrani, the man who it suspected of the shooting attack that killed three people last month at the Naval Air Station in Pensacola, Florida.

FBI General Counsel Dana Boente issued the request in a letter sent to his Apple counterpart last week. Boente said that, although FBI investigators had obtained a search warrant to examine the phones, their investigators had so far been unable to guess the passcodes needed to unlock them and thus decrypt their stored contents. And further complicating any data retrieval is the fact that Alshamrani shot one of the phones, putting a bullet through it. So, depending upon where the bullet went, the data might well be physically destroyed even if it might have been otherwise decryptable.

When asked, an FBI spokeswoman confirmed that the letter was sent but declined to provide any detail as to its contents, citing the usual annoying dodge of it being part of an ongoing investigation.

Apple released a statement saying: "We have the greatest respect for law enforcement and have always worked cooperatively to help in their investigations. When the FBI requested information from us relating to this case a month ago, we gave them all of the data in our possession, and we will continue to support them with the data we have available." And Apple's representatives would not say whether Apple would comply with the government's request.

So we're back here again. Recall that it was in 2016 that the FBI obtained that court order requiring Apple to help unlock and decrypt the iPhone used by Syed Farook, who shot and killed 14 people and injured 17 others during his 2015 shooting rampage in San Bernardino, California.

What the FBI asked for then was for Apple to create custom firmware for the phone to remove the self-wiping protection after 10 failed attempts to guess the password. In court documents and congressional testimony at the time, the FBI said they had no other way to access the contents of the iPhone so that investigators could determine if Farook and his wife (who also participated and died in the shooting) acted in concert with others to carry out the deadly attack.

And, as we also know, the dispute was resolved or at least dropped 6 weeks after the government obtained the court order when government lawyers reported that FBI investigators had decrypted Farook's iPhone 5C and no longer needed Apple's help. Though it was never completely confirmed, the FBI Director at the time, James Comey, suggested that the agency paid more than $1.3 million to an unnamed company to crack the phone's encryption.

As our listeners know, during the intervening time very little has changed except that, notwithstanding the persistent CheckRain jailbreak (which we'll update in a moment), Apple's encryption is stronger, as is its insistence that, much as it would love to help, its customer's privacy is paramount and, consequently, the design of its systems renders it unable to comply with lawful government requests for information. And, as we know, other US companies, notably Facebook, are more recently adopting the same position.

So we're facing another standoff, and another very specific case over which the two sides may decide to fight. Sooner or later something is going to force a resolution to the increasingly pregnant question of whether future purveyors of commercial encryption technology will be required, by law, to incorporate some form of trapdoor technology into their systems' designs to allow them to respond to legally issued search warrants?

Note that in this case, I entirely believe that Apple is telling us the whole truth about being absolutely unable to unlock and decrypt a modern iPhone. So, unlike the question of compelling the disclosure of a password or a biometric, this is NOT something for our US Supreme Court to "decide." It's not about "deciding" anything since Apple's system design has already decided it's impossible. And I'm sure it is. Instead, this will need to be resolved through legislative action, with US law being changed -- and this would be such a radical change that I'll be surprised if it can happen -- to effectively outlaw non-compliant commercial encryption technology.

And just so we're clear, that's entirely different from my assertion that Apple could arrange to add a silent listener tap into someone's iMessage stream if they wished to. Doing that is clearly possible and does not necessitate the corruption of the integrity of encryption. But any legislation that mandates that purveyors of commercial encryption-at-rest technology be able to respond to a court order... well... that's an entirely different kettle of fish. And in many application settings I would agree with experts who argue that there's no way for it to be done without fundamentally undermining the encryption security.

I keep coming back to Apple. But Apple really is a special case due to the stranglehold they have over the design and operation of their devices. They are much less OUR iPhones and iPads than a device Apple allows us to use within their iron grip. So many of the solutions I have proposed for how this could be securely accomplished only apply to the Apple model of absolute platform hegemony.

And speaking of iPhone jailbreaking...
I thought we should check-in on CheckRain: https://checkrain.com/

The group's efforts are now up to beta 0.9.1. To give our listeners some sense for what's been going on and where things stand, it appears that the list of caveats to the operation and use of CheckRain are steadily dwindling. Under "What's New" the web page says:

This beta fixes multiple bugs, including:

- An issue where the loader app would crash when installing Cydia on iPads
- A crash when the macOS language was set to anything other than English
- An issue where iPad Minis would not work with the GUI
- An issue with the scp binary not working as expected

This version also improves the clarity of some errors, specifically the -20 error, which now offers guidance on how to resolve it.

Unsupported devices: “checkra1n supports all devices between the iPhone 5s and the iPhone 11, however we are having dependency issues with certain iPad models”

- iPad Air 2
- iPad 5th Gen
- iPad Pro 1st Gen

Support for these devices will be added in a later release. Support for the following devices is experimental, and may require more attempts than usual:

- iPhone 5s
- iPad Mini 2
- iPad Mini 3
- iPad Air

Reliability on these devices will be improved in future releases.

Unsupported platforms: This beta is available for macOS and Windows. Work is ongoing to support Linux, which will be added in a later release.

And while we're on the subject of Apple and the privacy of their users...
Speaking during last week's Consumer Electronics Show, Apple's chief privacy officer, Jane Horvath confirmed that Apple is automatically and continuously scanning images backed up to iCloud to ferret out child abuse images. Jane explained that this was the way they were working to help fight child exploitation, as opposed to breaking encryption.

Recall that it was last month when Senator Lindsey Graham asserted his belief that unbroken encryption provides a “safe haven” for child abusers, saying "You’re going to find a way to do this or we’re going to do this for you." He continued: "We’re not going to live in a world where a bunch of child abusers have a safe haven to practice their craft. Period. End of discussion."

So, at least as far as iCloud backed-up images are concerned, Apple has been doing something about the problem, and, according to Jane, in a privacy-preserving way. Jane Horvath did not elaborate on the specific technology Apple is using, nor whether the company is using its own tools, but it is not alone in using automatic scanning to find illegal images. Here's what we know...

We know that just as a hash can be used to compare passwords without ever knowing either of the passwords, it would be possible hash images. And, to this end, for the past 12 years the the National Center for Missing & Exploited Children (NCMEC) has made available a large file of hash values for known child sexual abuse images. This list enables companies to hash any images they are storing to check large volumes of files for matches without those companies themselves having to keep copies of offending images.

Unfortunately, even one pixel of an image being one shade darker would result in a hash having half of its bits inverted. In other words, simple hashing is powerfully discriminating. It virtually cannot produce false positives -- which we would call a hash collision -- but images are

inherently far less exacting that passwords. Crop, resize or recompress an image and you'll produce a radically different hash for an image that looks identical.

So we need a different solution to this problem. We need to tolerate a vastly higher false-positive rate in return for being able to match on inexact but identical appearing images.

What we need is "Image Hashing" and Microsoft has donated its technology known as "PhotoDNA" to this effort. PhotoDNA creates a unique signature for an image by converting it to black and white, resizing it, and breaking it into a grid. The technology scans the pixels within each grid cell to build a histogram of intensity gradient edges from which it derives its so-called DNA. Images with similar DNA are highly likely to be depicting the same image.

And, as with a hash, since the data reduction performed by this process produces a small "DNA imageprint", large data sets can be scanned quickly, enabling companies including Microsoft, Google, Verizon, Twitter, Facebook and Yahoo to find needles in haystacks and sniff out illegal child abuse imagery.

Henry Farid, one of the people who helped develop PhotoDNA was asked about the application of PhotoDNA to encrypted images. He explained that thanks to recent advances in partial or fully homomorphic encryption, it's possible to perform image hashing on encrypted data. This means that images in encrypted messages can be checked against known harmful material without Facebook or anyone else being able to decrypt the image. This analysis provides no information about an image's contents, thus preserving privacy, unless it is a known image of child sexual abuse.

So, during CES, Jane Horvath confirmed what others had picked up on months ago: that Apple had been conducting photo scanning for months, at least. In October, Mac Observer noted that it had come across a change to Apple's privacy policy that dates back at least as far as a 9 May 2019 update. In that update, Apple inserted language specifying that it is scanning for child abuse imagery:

"We may …use your personal information for account and network security purposes, including in order to protect our services for the benefit of all our users, and pre-screening or scanning uploaded content for potentially illegal content, including child sexual exploitation material.

So, in other words, despite the protestations of some in the US congress, the tech industry is responsibly scanning for child abuse materials, all the large tech firms are doing it while never reading messages nor examining human-observable photo content.

The SIM Swapping Threat

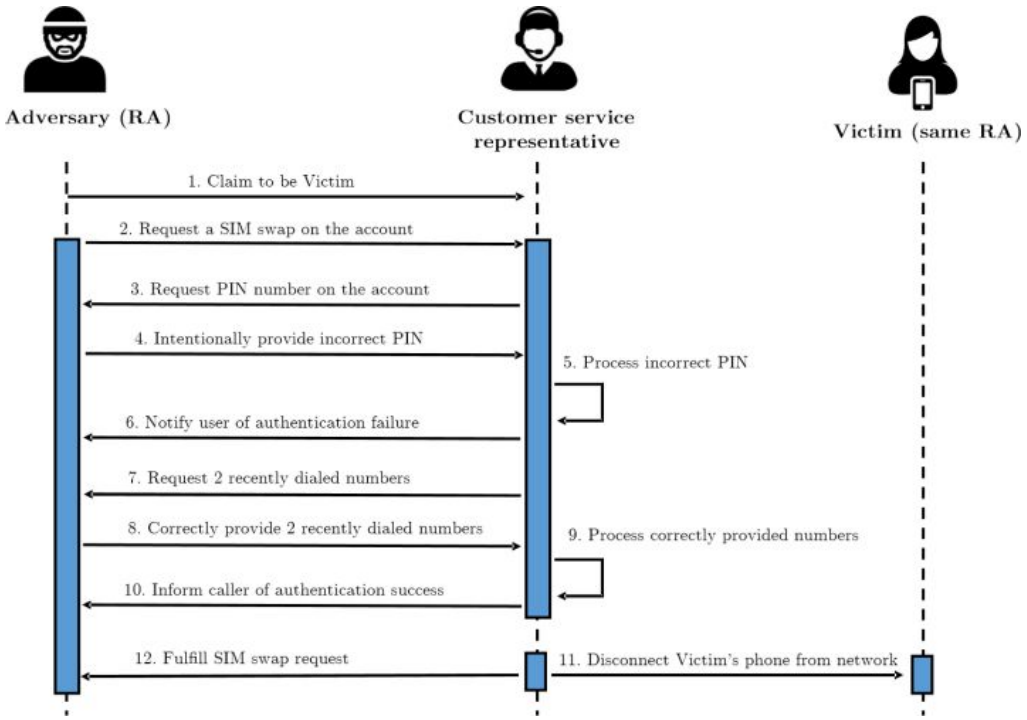
Robustly authenticating identity in an online world is difficult. As we know, I just invested six years of my life working to create a robust solution for identity authentication. The standard fallback is for the agency wishing to identify us to send a text message to the mobile phone associated with our account. The problem is, this ties our identity to our phone, yet we are not our phone.

On January 10th, a group of four researchers at Princeton published a whitepaper detailing their research into spoofing mobile carriers. Now, if they weren't from Princeton, the paper might have been titled something like "Holy Crap! You're not going to believe what we did!" But, instead, their paper carries the sufficiently dry and academic title: "An Empirical Study of Wireless Carrier Authentication for SIM Swaps"

https://www.issms2fasecure.com/assets/sim_swaps-01-10-2020.pdf

Abstract: We examined the authentication procedures used by five prepaid wireless carriers when a customer attempted to change their SIM card. These procedures are an important line of defense against attackers who seek to hijack victims' phone numbers by posing as the victim and calling the carrier to request that service be transferred to a SIM card the attacker possesses. We found that all five carriers used insecure authen-tication challenges that could be easily subverted by attackers.We also found that attackers generally only needed to target the most vulnerable authentication challenges, because the rest could be bypassed. In an anecdotal evaluation of postpaid [as opposed to prepaid] accounts at three carriers, presented in Appendix A, we also found—very tentatively—that some carriers may have imple-mented stronger authentication for postpaid accounts than for prepaid accounts.To quantify the downstream effects of these vulnerabilities,we reverse-engineered the authentication policies of over 140websites that offer phone-based authentication. We rated the level of vulnerability of users of each website to a SIM swap attack, and we plan to publish our findings as an annotated dataset. Notably, we found 17 websites on which user accounts can be compromised based on a SIM swap alone, i.e.,without a password compromise.

So, first of all, so as not to keep everyone in suspense, the bad news is that five mobile carriers found to be vulnerable were: AT&T, T-Mobile, Tracfone, US Mobile, and Verizon Wireless. In all five cases the authentication procedures were found to be vulnerable and allowing attackers to readily conduct SIM swapping attacks.



The diagram above shows one authentication bypass flow…

The researchers key findings were, #1: **That Mobile carriers use insecure methods for authenticating SIM swaps.** Specifically…

A. Last Payment:
We found that authenticating customers via recent payment information is easily exploitable. AT&T, T-Mobile, Tracfone, and Verizon use payment systems that do not require authentication when using a refill card. An attacker could purchase a refill card at a retail store, submit a refill on the victim's account, then request a SIM swap using the known refill as authentication.

B. Recent Numbers:
We also found that using information about recent calls for authentication is exploitable. Typically CSRs requested information about outgoing calls. Consider the hypothetical following attack scenario: Using only the victim's name and phone number, our simulated adversary could call the victim and leave a missed call or message that would prompt the victim into returning the call to a number known to the attacker. This call would then appear on the outgoing call log and the attacker could use it for authentication. CSRs appeared to also have the discretion to allow authentication with incoming call information, as this occurred four times between AT&T, T-Mobile, and Verizon. An attacker can trivially generate incoming call records by calling the victim.

C. Personal Information:
We found that Tracfone and US Mobile allowed personal information to be used for authentication. While our attacker did not use this information, it would likely be readily available to real attackers (e.g., via data aggregators) and is often public, so it offers little guarantee of the caller's identity. We note that for over a decade, FCC rules have prohibited using "readily available biographical information" to authenticate a customer requesting "call detail information."

D. Account Information:
We found that AT&T, US Mobile, and Verizon allowed authentication using account information. As with personal information, this information would often be readily available to an adversary. Receipts (whether physical or electronic), for example, routinely include the last four digits of a payment card number. We note that PCI DSS, the industry standard for protecting payment card information, does not designate the last four digits of a payment card as "cardholder data" or "sensitive authentication data" subject to security requirements. As for the activation date associated with an account, that information may be readily available from business records (e.g., via a data aggregator), inferable by website or mobile app logs (e.g., via UserAgent logs), or inferable via mobile app API access (e.g., via the usage stats API on Android or the health APIs on Android and iOS). We note that FCC rules also prohibit using "account information" to authenticate a customer requesting "call detail information."

E. Device Information:
We found that all carriers except for T-Mobile use device information for authentication. These authentication methods included the customer's IMEI (device serial number) and ICCID (SIM serial number). Both the IMEI and ICCID are available to malicious Android apps, and IMEIs are also available to adversaries with radio equipment.

F. Security Questions:
We found that Tracfone used security questions for authentication. We also found that T-Mobile, Tracfone, and Verizon prompted users to set security questions upon sign up. Recent research has demonstrated that security questions are an insecure means of authentication, because answers that are memorable are also frequently guessable by an attacker.

#2. Some carriers allow SIM swaps without authentication.
Tracfone and US Mobile did not offer any challenges that our simulated attacker could answer correctly. However, customer support representatives at these carriers allowed us to SIM swap without ever correctly authenticating: 6 times at Tracfone and 3 times at US Mobile.

#3. Some carriers disclose personal information without authentication, including answers to authentication challenges.

- AT&T. In 1 instance, the representative disclosed the month of the activation and last payment date and allowed multiple tries at guessing the day. They also guided us in our guess by indicating whether we were getting closer or further from the correct date.

- Tracfone. In 1 instance, the representative disclosed the service activation and expiration dates. Neither are used for customer authentication at Tracfone.

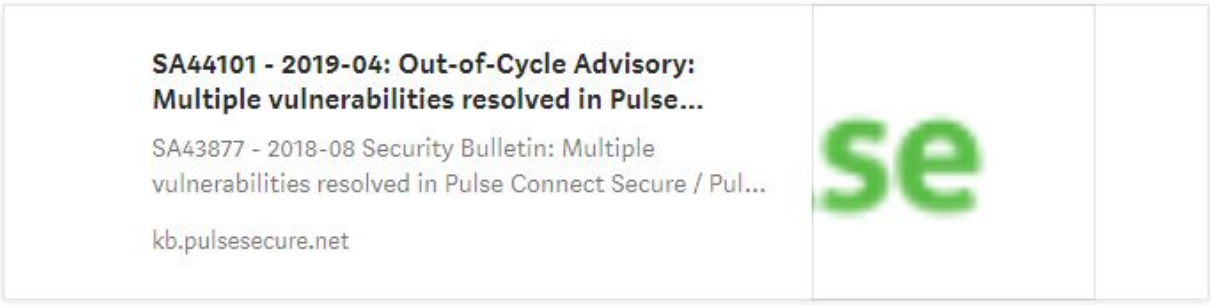
- US Mobile. In 3 instances, the representative disclosed the billing address on the account prior to authentication. In 1 instance, a portion of the address was leaked. In 1 instance, part of the email address was disclosed. In 3 instances, the representative disclosed portions of both the billing address and email address.

So… we're kind of in some deep trouble here where, as an industry, we are relying so much on the possession of our phone, whose communication can rather easily be commandeered and rerouted to an adversary.

Anatomy / Timeline of the exploitation of an unpatched VPN bug.
Okay, so first of all, this is another of those "If you, your organization, or anyone you know or care about is using a VPN by the name of "Pulse Secure" stop listening right now and verify that they have patched their VPN server endpoint anytime since last April 2019, when an emergency patch to close a serious remote access vulnerability was patched.

Because... THIS is the story of the nearly 15,000 Pulse Secure VPN server endpoints that did NOT and have not patched:

Our story begins in April 2019, when Pulse Secure issued an advisory for their apparently aptly named "Zero Trust" VPN product. (Though I'm sure that's not the way they meant the name to be taken.) At the time they warned organisations of an out of cycle patch which fixed a vulnerability in their product Pulse Connect Secure:



With organizations being so negligent about the application of Windows OS patches, where virtually all the work is done for them, you can imagine how often patching occurs outside of the Windows ecosystem. In other words… next to never for the most part.

This particular vulnerability is as bad as it gets for a VPN. Our listeners could probably write the next few sentences themselves: It allows people **without** valid usernames and passwords to remotely connect to the corporate network the device is supposed to protect, turn off multi-factor authentication controls, remotely view logs and cached passwords in plain text (including Active Directory account passwords). Yes... a full remote authentication bypass for a widely deployed corporate and government VPN.

About four months later on 14th August 2019 someone posted an exploit for the issue on Kevin Beaumont's forum, OpenSecurity.global: (https://opensecurity.global/) Then, a few short days later, a public exploit by Justin Wagner and Alyssa Herrera appeared.

And on August 22nd 2019, with the help of BinaryEdge.io (a sort of commercial version of Shodan, an Internet scanning service), Kevin determined that someone was scanning the Internet for the Pulse Secure vulnerability.

Kevin said that he sent up a flare that organisations needed to urgently patch this 4 month old CRITICAL vulnerability. On August 25th 2019, Bad Packets scanned the Internet and found nearly 15,000 endpoints across the world had the issue directly exploitable:

Country	Count of Vulnerable Hosts
United States	5,010
Japan	1,511
United Kingdom	830
Germany	789
France	626
Netherlands	420
Israel	406
Switzerland	307
Canada	296
South Korea	281
All Other Countries	4,052

Those results included networks at governments across the world and many incredibly sensitive organisations included. And what was essentially a list of the world's largest companies. It was clear organisations were not patching.

Since then, Bad Packets has been working with the various CERTs, CSIRTs and ISACs and other related government groups to share the scan data and encourage organisations to patch.

Next it became clear that Advanced Persistent Threat (APT) groups were using the vulnerability to gain access to networks. Last October warnings to patch were sent out via US CISA, the US NSA and the UK's National Cyber Security Centre.

Since Kevin works in corporate cybersecurity he follows ransomware and general cyberattacks because he needs to know what attackers are up to. He had seen the correlation between companies being successfully attacked with those who used the Pulse Secure VPN. And he saw from Internet vulnerability scanning that many of the organisations either hadn't patched their Pulse Secure system at the time of the incident, or had only patched recently. As part of the vulnerability it's possible to backdoor Pulse Secure systems, and gain access even after the VPN has itself been patched.

ANd, finally, last week he spotted two notable incidents where recently breached companies had a strong reason to believe that Pulse Secure was the entrypoint for the breach, and was then used to deliver Sodinokibi (REvil) ransomware. In both cases the organisations had unpatched Pulse Secure systems, and the footprint was the same — access was gained to the network, domain admin was gained, VNC was used to move around the network (they installed VNC via psexec, as java.exe), endpoint security tools were disabled and Sodinokibi was pushed to all systems via psexec. Kevin wrote that he had now seen an incident where they can prove Pulse Secure was used to gain access to the network.

Today there remain more than 1,000 unpatched WIDE OPEN systems on the Internet, and in Kevin's final update he added that he had just learned that "Travelex" had been hit by a Sodinokibi attack. Kevin noted that they had 7 unpatched Pulse Secure servers.

I think it's clear that as an industry we still haven't solved the problem of communicating the urgency of patching and getting systems patched.

And speaking of patching right away…

China's Qihoo 360 security company spotted a serious (as in CRITICAL) type confusion bug in Firefox's IonMonkey JavaScript Just-In-Time (JIT) compiler being abused in the wild. And, as we know, that makes it a 0-day vulnerability.

Two days after releasing Firefox 72, Mozilla issued an update to patch and resolve this critical zero-day flaw.

Attacking a JIT compiler is both clever and common because compilers, like interpreters, tend to be finicky. But also because it's their job to create new executable instructions on-the-fly. That means that they cannot be constrained by the safeguards provided by DEP -- Data Execution Protection -- which would otherwise prevent the execution of data. Executing freshly-compiled data, as instructions, is precisely what JIT compilers must do.

When Qihoo 360 notified Mozilla of the trouble they indicated that they discovered the flaw after observing targeted attacks occurring in the wild.

My Firefox updated itself over the weekend to v72.0.1, which is what everyone should now have.

But I've seen Firefox wait to be asked. So it might be worthwhile to check under Help / About Firefox just to be sure. If you're not yet current that will wake it up.

Miscellany

Star Wars
OMG!!! The sheer horror of the "4DX" experience!!! I would never have imagined that I would walk out of a Star Wars movie! My first indication that perhaps I'd made a serious mistake was when I spotted the control on the armrest to turn off being sprayed with water. What!?!?!?!?!

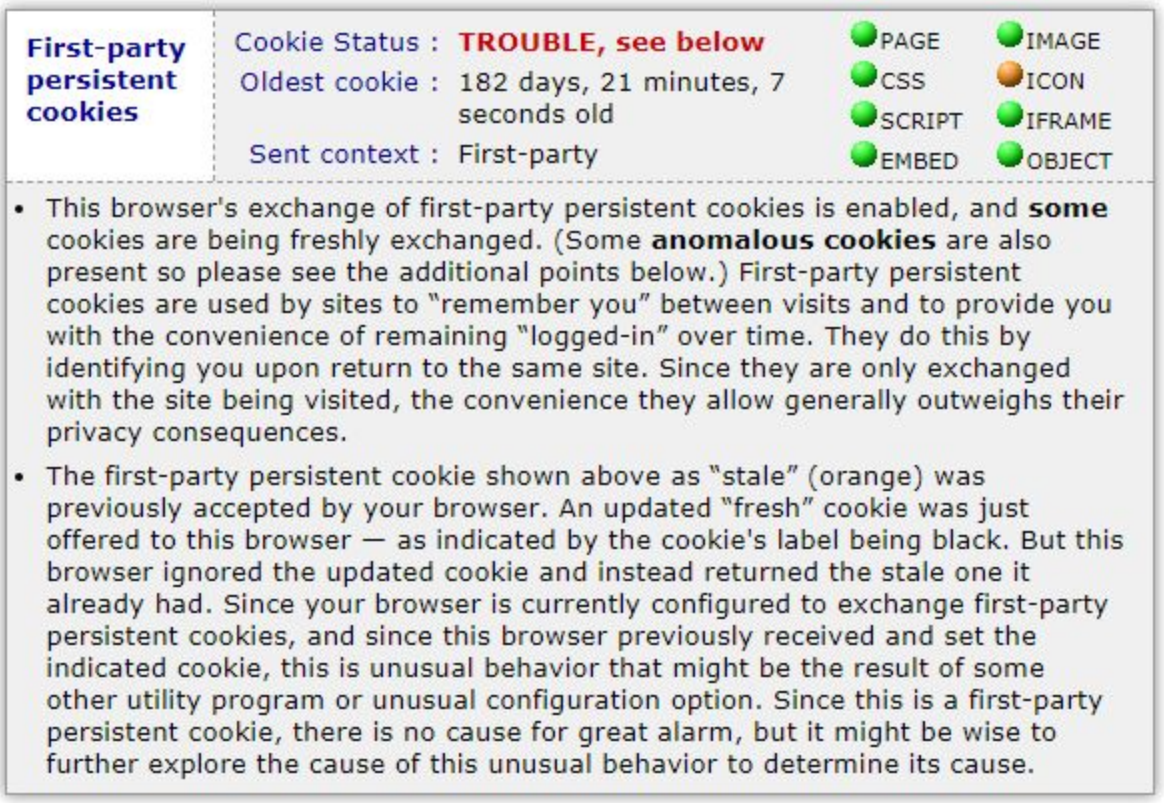
The 4DX tickets were $26.70 each. $26.70!! Fortunately, we were able to get our $80 refunded for the three tickets!

Also, the movie was 3D, but it wasn't the cool RealD 3D utilizing the clockwise and counter-clockwise polarization we've talked about here before. We were actually given Red/Green tinted glasses! What year is it?

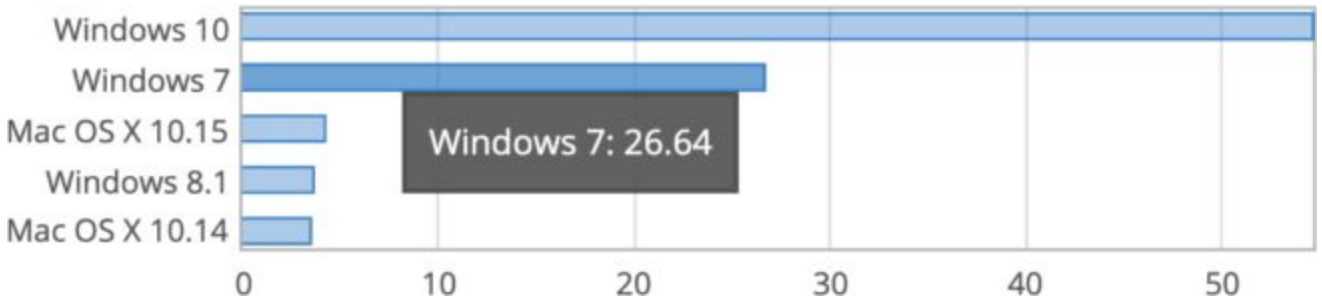
Credit Bureau Locking and Unlocking
Medicare, Social Security & an Amazon Store Card.

- Equifax: 1-800-685-1111 (NY residents 1-800-349-9960)
- Experian: 1-888-397-3742
- TransUnion: 1-888-909-8872

Cookies: https://grc.sc/cookies
Firefox came out perfectly… but to my surprise, Chrome is NOT handling cookies correctly!:



Windows 7 - R. I. P.



As of today, Windows 7 stops receiving its much-needed security patches -- unless they are purchased. Microsoft has them, but they are no longer being included as part of the purchase price. And I say "much needed" because it's not as if anything has been slowing down recently due to all of the bugs and security flaws which riddle Windows 7 having finally been fixed. No. Yet Windows 7 continues to command an overall desktop market share of 26.64% -- or more than 1 out of every 4 desktop OSes currently in use. Over the past 5 years I've pretty much made peace with Windows 10. So I won't be setting up any new Win7 machines. But it's too bad... since I otherwise MUCH prefer Windows 7. **Windows 7 is a tool. Windows 10 is a toy.** I noticed that the Win10 download shortcut is seeing a lot of use **https://grc.sc/Win10**

Cable Haunt

Hundreds of millions of popular Broadcom-based cable modems are vulnerable to a newly discovered vulnerability which has been dubbed "Cable Haunt"

https://cablehaunt.com/
https://github.com/Lyrebirds/Cable-Haunt-Report/releases/latest/download/report.pdf

Okay… so what new horror have we here?

"Cable Haunt" is the name given to a critical vulnerability found in cable modems from various manufacturers around the world. The reason so many various brands share the same problem is the very worrisome tendency we're seeing toward a monoculture. Broadcom is by far the dominant supplier of cable modem core technology, and Broadcom published some reference firmware -- which everyone copied.

The vulnerability ultimately enables remote attackers to execute arbitrary code on vulnerable modems. And at this point it looks like pretty much all modems are vulnerable. This exploitation is accomplished indirectly through an endpoint on the **internal** local network. Through malicious communication with this internal endpoint, a buffer overflow can be exploited to gain control over the cable modem.

How widespread is the problem?
The researchers write: "There are an estimated 200 million cable modems in Europe alone. With almost no cable modem tested being secure without a firmware update, the number of modems initially vulnerable in Europe is estimated to be close to this number. However, it is difficult to give a precise estimate of the reach of Cable Haunt. The reason for this is that the vulnerability originated in reference software, which has seemingly been copied by different cable modems manufacturers when creating their cable modem firmware. This means that we have not been able to track the exact spread of the vulnerability and that it might present itself in slightly different ways for different manufacturers. We have contacted as many of the largest ISPs and manufacturers as we could ahead of time, to give them time to fix the issue, but with varying success. Some of the contacted ISPs have informed us that they have or are rolling out firmware updates; however, we are still missing updates from several, and some have wished not to be listed on this website. The ISPs that have confirmed their modems are secure can be found below.

The affected component is the cable modem's core OS which is eCos which is a widely popular embedded multi-threaded realtime OS. Being an embedded OS, it is meant to be small and fast and lightweight. Since it also only runs trusted compiled-in code, it completely lacks any of the now common anti-malware preventions such as address space layout randomization, protections against stack smashing or execution, thus freely allowing execution of code on the stack. The result is that exploits are unusually easy to write, implement, and run with high reliability. Code will always be at fixed known addresses and the OS never performs any stack sanity-checking, making it entirely vulnerable to buffer overflow attacks. This made finding the Cable Haunt exploit much easier, just as it makes its further exploitation also far easier.

Our cable modems all have a built-in spectrum analyzer which can be used to identify connection problems with the cabling system. Although the port which exposes the spectrum analyzer varies by modem make and model, it is readily discoverable with any port mapping tool such as nmap. And the proof of concept Python script performs this port scan to locate the analyzer.

Requests to the spectrum analyzer are sent JSON-formatted through a websocket. However, the JSON deserializer used in the spectrum analyzer allocates a predefined amount of memory for each parameter, but will keep reading input parameters until a comma (,) is reached. Not surprisingly, this can be easily exploited by a malicious request. In their example, the fStartHz parameter has a value bigger than the allocated memory, and will therefore overflow and overwrite the registers. To validate this, a JSON package with 200 A's as the fStartHz parameter can be sent through the serial port to the CM. This will crash the modem and all register values will be displayed, showing that the program counter has changed to 0x41414141.

```
{
   "jsonrpc":"2.0",
   "method":"Frontend::GetFrontendSpectrumData"
   ,"params":{
     "coreID":0,
     "fStartHz":0,
     "fStopHz":1000000000,
     "fftSize":1024,
     "gain":1,
     "numOfSamples":256
     }
   ,"id":"0"
  }
```

This is the expected Spectrum Analyzer query format.

```
{
   "jsonrpc":"2.0",
   "method":"Frontend::GetFrontendSpectrumData"
   ,"params":{
     "coreID":0,
     "fStartHz":"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA....",
     "fStopHz":1000000000,
     "fftSize":1024,
     "gain":1,
     "numOfSamples":256
     }
   ,"id":"0"
  }
```

This is a malformed query that shows a bogus 'fStartHz' query value.

The eCos OS saves the caller's registers on the stack and restores these before returning. Therefore, if the variable registers S0-S7 are overwritten and the return address register is saved on the stack, it's trivial to run any existing code in the system, with the attacker's desired input variables. Although they did not bother to execute their own code, using Return Oriented Programming (ROP) they were able to execute existing code on the system in a turing-complete manner and manipulate the system extensively. This can be used, for example, to open a telnet server for **external** root access to the cable modem, allowing remote access to the entire system. Through this telnet connection they were able to access a range of methods, including reading and writing memory addresses, and executing code from any memory address, including ones just written to. They noted that the last steps vary from modem to modem, but they provide a complete example in Appendix B of their report.

The attack can be executed by having the victim run malicious javascript. A common avenue of attack would be a link that is opened in a browser, but could for example, also be done through ads on a trusted website or email clients. The exploit starts when the malicious code has been sent to the client, and is executed.

The javascript running in the browser establishes a websocket connection, directly to the modem via the local IP address . Websocket is not explicitly covered by CORS (Cross Origin Resource Sharing) and SOP (Same Origin Policy). Therefore, browsers should not be relied upon to prevent this, and some will not. While browsers will set the origin parameter on all websocket requests, it is the server's responsibility to reject the request as needed. None of tested modems reject these types of requests. And using an external domain with a short DNS expiration allows a DNS rebinding attack to succeed by redefining the domain's resolution to the local IP.

All of this means that attacks a wide range of attacks and abuses are enabled, including:

- Change default DNS server
- Conduct remote man-in-the-middle attacks
- Hot-swap code or even the entire firmware
- Upload, flash, and upgrade firmware silently
- Disable ISP firmware upgrade
- Change every config file and settings
- Get and Set SNMP OID values
- Change all associated MAC Addresses
- Change serial numbers
- Be exploited in botnet

Unlike our routers, consumers do not update our own cable modems. This is only done from the broadband WAN side. Thanks to this podcast, I have a friend at COX who's deep into the technology. In the past, he's pushed firmware updates to my COX-connected modems, so I'm sure that can be done. It causes a brief outage while the modem reboots and relocks to the network… but I think it's entirely foreseeable that pretty much everyone who has a cable modem is going to be seeing a brief outage as soon as our cable companies get their acts together and catch up.

Threatpost updated their earlier coverage this morning by adding: "As far as U.S. ISPs are concerned, a Cox spokesperson told Threatpos: "we are rapidly testing all our in-home broadband equipment, determining any vulnerability and the best steps to mitigate, as needed."

And a spokesperson with Charter told Threatpost that Charter is "currently working with each of our vendors to determine if their equipment is vulnerable and when we could expect to see a firmware upgrade."

The problem is, these sorts of large, world-wide and super-juicy targets are rare. And compared with other working exploits, the virtually defenseless nature of the embedded eCos OS means that this one is almost sure to be targeted just as quickly as the bad guys can assemble their tools. And once firmware updates are blocked, there's nothing that can be done.

So what do WE listeners of Security Now do in the mean time.

First, you want to see whether your cable modem is accessible from your LAN. Mine was. I'm able to bring up its configuration webpage at 192.168.100.1. And while I have a maximum-entropy maximum-length 32-character password protecting its web interface, it's not the web interface that needs protection. So I ran a 65535 ports scan against the IP and it revealed that ports 80 and 8080 we both open:

192.168.100.1

Status: Alive
Operating system:
IP: 192.168.100.1
MAC:
Manufacturer:
NetBIOS:
User:
Type:
Date:
Comments:

Service	Details
HTTP	
Port **80** (TCP)	
Port **8080** (TCP)	

Windows users can easily do this using the free "Advanced Port Scanner" utility which doesn't even need to be installed: **https://grc.sc/aps** (APS = Advanced Port Scanner)

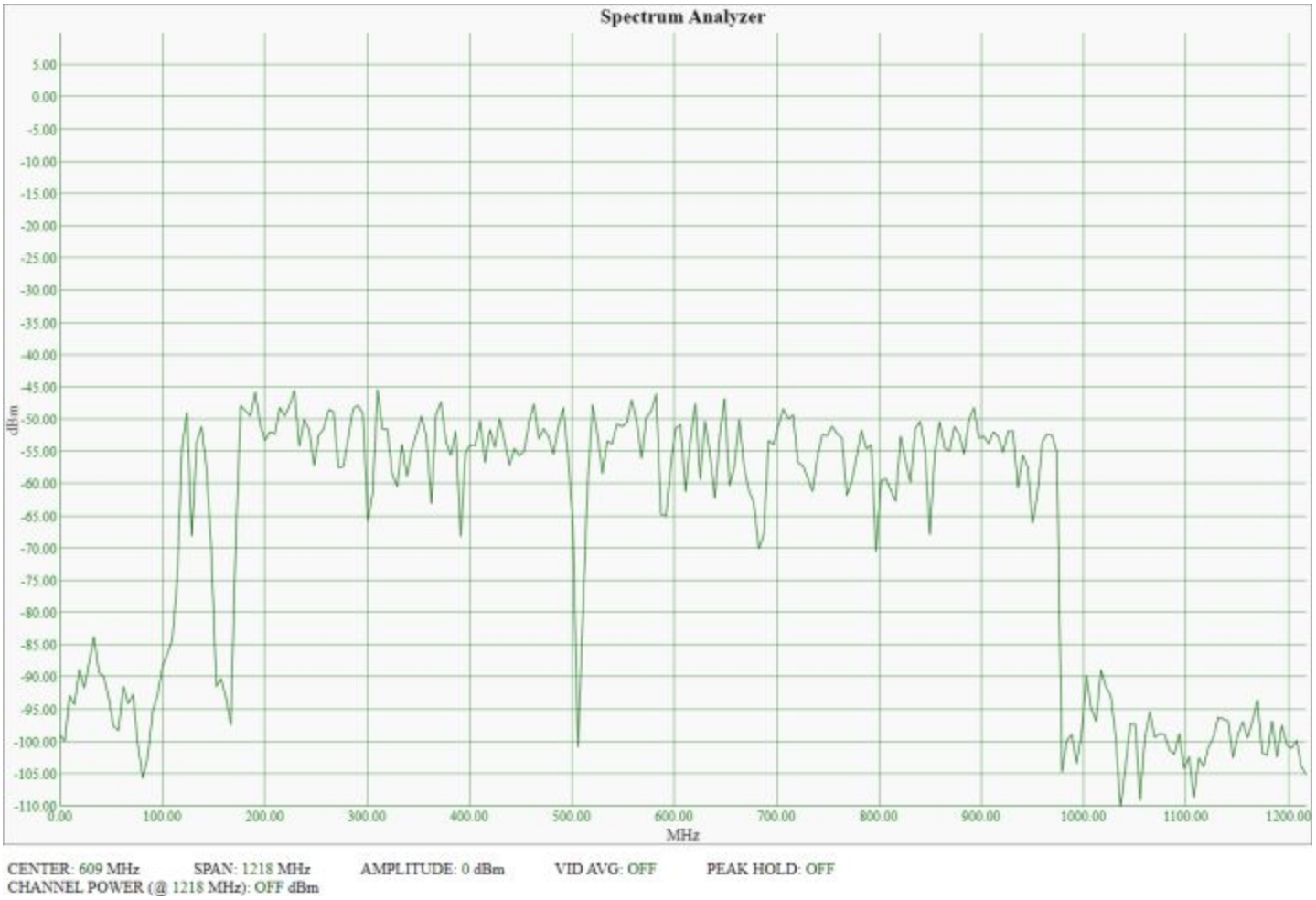
https://www.advanced-port-scanner.com/
The same company also offers a free IP scanner: **https://www.advanced-ip-scanner.com/**

To my surprise, the Advanced Port Scanner showed me that my cable modem not only has port 80 open for HTTP web management, but also port 8080, which I never knew. So I pointed by browser to 192.168.100.1:8080 and was greeted with the interesting notice:

"Spectrum Analyzer not supported in this browser. Please use Safari or Chrome."

Huh! While researching this I did see a mention that this cable modem nightmare could not be exploited from Firefox. This is apparently why. The spectrum analyzer doesn't support Firefox, so Firefox cannot be used to launch these attacks. But... anything else could -- like a rogue IoT hub or camera... so this still needs to be fixed.

So next, I did the same 192.168.100.1:8080 from Chrome and was greeted with an amazing spectrum analysis of my cable modem's connection:



How cool is that? I never knew that was in there! However, this also proved what I already knew, which was that my lovely Netgear DOCSIS 3.0 cable modem is, indeed, almost certainly vulnerable -- it has a browser-accessible spectrum analyzer!

However, that cable modem feeds directly, and only, into my wonderful pfSense firewall router. And, since I virtually never need to access my cable modem's web interface, it's trivial to add a firewall rule to block all access to my cable modem's web and port 8080 interface. This will prevent any possibility of exploitation until COX has the chance to push a firmware update.