Security Now! #751 - 01-28-20

SHAmbles

This week on Security Now!

This week we look at some surprising revelations of Apple's cloud storage encryption (or lack thereof). We also cover a Microsoft cloud database mistake, some interesting legislation under consideration in New York, new attacks against a consumer router firmware, a rise of new attacks against our browsers, a welcome new publication from NIST on Privacy, a massive leakage of telnet usernames and passwords, a welcome micro-patch for this month's IE 0-day, a bit of miscellany and SpinRite news, and then some coverage of the final nail that was recently pounded into SHA-1's coffin.

Security News

Is Apple actually encrypting our iCloud storage backups?

I wanted to start out this week by discussing a significant bit of news that first came to my attention last week when "Jeff Root," a frequent participant in GRC's newsgroups, shared a link to some apparently exclusive coverage from Reuters. And since this is significant news, others in the tech press later picked up on it as well.

https://www.reuters.com/article/us-apple-fbi-icloud-exclusive/exclusive-apple-dropped-plan-for-encrypting-backups-after-fbi-complained-sources-idUSKBN1ZK1CT

Reuters story is titled: "*Apple dropped plan for encrypting backups after FBI complained - six sources*"

Six sources familiar with this matter, told Reuters that Apple had dropped its plans to give iPhone users the option to fully encrypt backups of their devices in the company's iCloud service after the FBI complained that the move would harm investigations. This reversal on Apple's part about two years ago had not been previously reported. Reuters stated that it showed how much Apple has been willing to help U.S. law enforcement and intelligence agencies, despite taking a harder line in high-profile legal disputes with the government and casting itself as a defender of its customers' information.

As we know from our reporting last week, responding to subpoenas from law enforcement in the matter of the shooting at the Pensacola, Florida naval base last month, Apple said that it rejected the characterization that it "has not provided substantive assistance" while turning over gigabytes of data, which was almost certainly the suspected shooter's iCloud phone backups.

But in addition to this, Reuter's says that behind the scenes Apple has provided the U.S. Federal Bureau of Investigation with more sweeping help, not related to any specific high-profile probe. When asked about this an Apple spokesperson declined to comment on the company's handling of the encryption issue, or any discussions it has had with the FBI. And neither did the FBI respond to requests for comment on any discussions with Apple.

Citing one current and three former FBI officials and one current and one former Apple employee, Reuters reported that more than two years ago Apple told the FBI that it planned to offer users end-to-end encryption when storing their phone data on iCloud. Under that plan, primarily designed to thwart hackers, Apple would no longer have a key to unlock the encrypted data, meaning it would not be able to turn material over to authorities in a readable form even under court order. During private talks with Apple soon after, representatives of the FBI's cyber crime agents and its operational technology division objected to the plan, arguing it would deny them the most effective means for gaining evidence against suspects using iPhones. And... when Apple later spoke privately to the FBI about its work on phone security the following year, the end-to-end encryption plan had been dropped, according to those six well-placed sources. Reuters said that it was unable to determine exactly why Apple had dropped their sweeping encryption plan.

Another former Apple employee said he was told, without any specific mention of why the plan was dropped or if the FBI was a factor in the decision: "Legal killed it, for reasons you can

imagine." That former Apple employee told Reuters the company did not want to be attacked by public officials for protecting criminals, sued for moving previously accessible data out of reach of government agencies, or used as an excuse for new legislation against encryption... All of which does sound pretty reasonable.

Referring to Apple's court battle with the FBI in 2016 over access to an iPhone used by one of the suspects in the San Bernardino, California mass shooting, this person said: "They decided they weren't going to poke the bear anymore."

Two of the former FBI officials, who were not, themselves, present in talks with Apple, told Reuters that it appeared the FBI's arguments that the backups provided vital evidence in thousands of cases were persuasive and had ultimately prevailed. One of them said: "It's because Apple was convinced. Outside of that public spat over San Bernardino, Apple gets along with the federal government." However, another former Apple employee said it was possible the encryption project was dropped for other reasons, such as concern that more customers would find themselves locked out of their data more often... So Apple needed a means for allowing them back in. Three people familiar with the matter told Reuters that once the decision was made the handful of experts who were assigned to the Apple encryption project - which carried the code names Plesio and KeyDrop - were told to stop working on that project. No need to encrypt iCloud, after all.

And OS, of course, Apple's decision not to implement full end-to-end encryption of iCloud backups makes the FBI's job easier. As we know, the FBI relies upon hacking software to exploit security flaws to break into iPhone. But that method requires direct access to the phone which would ordinarily tip off the user, who is often the subject of the investigation. Apple's iCloud, on the other hand, can be searched in secret.

During the first half of 2019 -- last year -- the period which is covered by Apple's most recent semiannual transparency report on requests for data it receives from government agencies, U.S. authorities armed with regular court papers asked for -- and obtained -- full device backups and other iCloud content in 1,568 cases, covering about 6,000 accounts. Apple's transparency report states that it turned over at least some data for 90% of the requests it received.

The report also shows that Apple receives more requests in the form of secret U.S. intelligence court directives of which it received and responded to more than 18,000 accounts during the first half of last year. If Apple had proceeded with its plan it would not have been able to turn over any readable data belonging to users who opted to enable end-to-end encryption.

So what's going on? Rather than protecting =ALL= iCloud-stored data with end-to-end encryption, Apple compromised and shifted their strategy to protect only the most sensitive user information, such as saved passwords and health data. But backed-up contact information and texts from iMessage, WhatsApp and other encrypted services remain available to Apple employees and authorities.

I was curious to see what Apple themselves have to say about the encryption of iCloud data and, if it is read with this information in mind, it's clear that they say this without saying if clearly:

End-to-end encrypted data
End-to-end encryption provides the highest level of data security. Your data is protected with a key derived from information unique to your device, combined with your device passcode, which only you know. No one else can access or read this data.

These features and their data are transmitted and stored in iCloud using end-to-end encryption:

- Home data
- Health data (requires iOS 12 or later)
- iCloud Keychain (includes all of your saved accounts and passwords)
- Payment information
- QuickType Keyboard learned vocabulary (requires iOS 11 or later)
- Screen Time
- Siri information
- Wi-Fi passwords
- To access your data on a new device, you might have to enter the passcode for an existing or former device.

Messages in iCloud also uses end-to-end encryption. If you have iCloud Backup turned on, your backup includes a copy of the key protecting your Messages. This ensures you can recover your Messages if you lose access to iCloud Keychain and your trusted devices. When you turn off iCloud Backup, a new key is generated on your device to protect future messages and isn't stored by Apple.

In other words… nothing else other than the above-listed subset of all iCloud data is encrypted in such a way that Apple cannot and will not supply it to authorities when served with a legally binding subpoena. And the mention that Messages (and not just iMessages) also use end-to-end encryption, but with the key stored alongside it.

At: https://www.apple.com/privacy/
"Messages are only seen by who you send them to. Apple can't read your iMessages while they're being sent between you and the person you're texting."

Reuters reminded us that well back in October of 2018 Google a system similar to Apple's now-dropped plan for secure backups. Google, whose Android OS now runs on about three-quarters of the world's mobile devices, said users could back up their data to its own cloud without trusting the company with the key. Two people familiar with THAT project said Google gave no advance notice to governments, and picked a time to announce it when encryption was not in the news. Google continues to offer the service but declined to comment on how many users have taken up the option. The FBI did not respond to a request for comment on Google's service or the agency's approach to it.

Android Central: *"Apple may have ditched encrypted backups, but Google hasn't. Google offers end-to-end encryption for Android backups, and it won't be able to decrypt the data if you lose access."*
https://www.androidcentral.com/apple-may-have-ditched-encrypted-backups-google-hasnt

Android Central: A bombshell report from Reuters suggests Apple ditched end-to-end encryption for iCloud backups at the behest of the FBI. Citing several former Apple employees and FBI officials, the publication notes that Apple planned to switch to end-to-end encryption for iCloud — putting it on the same level as iPhones and iPads — but reversed course after consulting with the FBI.

iCloud data is also encrypted by default, but Apple holds a key to decrypt it. So in a scenario where an iCloud user is locked out of their account for whatever reason, Apple has the ability to decrypt the contents of that iCloud library. It is this reasoning that Tim Cook gave in defence of the move last year:

> "We do this because some users lose or forget their key and then expect help from us to get their data back."

And while that is certainly true, it's another form of half-truth like "we cannot read your iMessages while they're being sent between you and the person you're texting." which is to say, the messages truly are encrypted end-to-end, so we store them to the cloud before they're encrypted and after they're decrypted. It's a bit slimy.

According to Reuters, Apple was considering switching to end-to-end encryption wherein it won't be able to recover data even when served with a court order. However, the company ultimately decided to not do so:

250 Million Microsoft Customer Support Records Exposed Online
https://thehackernews.com/2020/01/microsoft-customer-support.html
https://threatpost.com/microsoft-250m-customer-service-records-open/152086/
https://nakedsecurity.sophos.com/2020/01/22/big-microsoft-data-breach-250-million-records-exposed/

On December 5th of last year, Microsoft misconfigured the access controls of a database of more than 250 million technical support service records dating back 14 years to 2005. Although Microsoft claimed that automated tools has removed unneeded personally identifiable information, Bob Diachenko, the cybersecurity database sleuth who spotted the unprotected database and reported it to Microsoft said that there remained ample readable information, including email addresses, IP addresses, Locations, Descriptions of CSS claims and cases, Microsoft support agent emails, Case numbers, resolutions, and remarks, and Internal notes marked as "confidential." Microsoft fixed the problem in New Years Eve.

So… not an Earth shaking story, but I didn't want to deliberately skip over it without a brief mention since the tech press had all picked up on it.

New York state is aiming to ban the use of public funds for Ransomware
The state of New York is exploring whether outlawing the payment of ransomware ransoms might make the problem go away. I'm not hopeful.

Two legislative bills have been proposed that would require government agencies to tell ransomware attackers they cannot pay. The first bill, S7246, was proposed by Senator Phil Boyle

on 14 January. If enacted into law, would, for small cities or towns with populations under 1 million, restrict the paying of attackers with tax money. And, if passed, it would also set up a $5 million fund to help overhaul the IT infrastructures of such small towns. (Not that $5 million would even begin to make a dent in solving the problem.)

The second bill, S7289, introduced by Senator David Carlucci two days later, on 16 January, would prohibit government agencies from paying ransom in the event of a cyberattack against their critical infrastructures. At this point the bills are both under discussion in committee, and it's unclear which of the bills will make it to a vote in the state senate.

As we discussed previously, last summer in June of 2019, the US Conference of Mayors passed a non-binding resolution to eschew paying ransom to get municipal systems back online after attack. But their resolution was only that. It lacked any legal teeth. New York is the first state to move in that direction and to back up the intent not to pay with some proposed legislation.

The text for the second bill, S7289, referred to the attack in Albany, New York last month on Christmas day which paralyzed the Albany International Airport. The attackers demanded a ransom in exchange for the return of data and restoration of the airport's systems. The desperate airport complied, paying an unspecified amount less than six figures. In response, the legislation says "We don't want to keep rewarding these crooks for these attacks. From the bill's text: "When municipal corporations and government agencies comply with these ransoms, they incentivize cyber-attackers looking to make a quick buck. Prohibiting these entities from complying with ransom requests will remove this incentive and safeguard taxpayer dollars."

Uh huh… and if you believe that I have an airport I want to sell you. Oh, wait… that airport's broken!

New Muhstik Botnet Attacks Target Tomato Routers

Palo Alto Networks' Unit 42 researchers observed a variant of the wormlike botnet that adds scanner technology to brute-force Web authentication against Tomato routers on TCP port 8080 and bypasses the admin web authentication by default credentials bruteforcing. The defaults in this case are "admin:admin" and "root:admin." The researchers wrote that they "captured Tomato router web authentication brute-forcing traffic."

The Tomato firmware, a Linux-based non-proprietary firmware known for its stability, VPN-pass through capability, and advanced quality-of-service control, is typically used by multiple router vendors and also installed manually by end users.

We've not talked much about open source router replacement firmware, but it's a big deal. The top six are DD-WRT, OpenWrt, Gargoyle, Tomato, the LEDE Project and libreCMC.

What is Tomato?

Tomato is a small, lean, open source alternative firmware for Broadcom-based routers. It features a new user-friendly GUI, a new bandwidth usage monitor, more advanced QOS and access restrictions, new wireless features such as WDS and wireless client modes, a higher P2P maximum connections limit, the ability to run custom scripts, connect via telnet/ssh, reprogram the SES/AOSS button, perform wireless site survey, and much more.

What is AdvancedTomato?
A router's graphical user interface is the most important part of the system because most users are unable or unwilling to configure a router by any other means. Tomato comes with a dated web interface with the option to change the color scheme but for some of us that is not enough. The interface simply feels out of date, out of style and in need of an update. AdvancedTomato enables you to keep all of the features of Tomato by Shibby and also upgrade your router's GUI to a clean and contemporary flat design. Users who demand a modern feature-filled firmware like Tomato deserve to explore those features using modern intuitive GUI like AdvancedTomato.

As our listeners know, I'm a fan of the FreeBSD-based pfSense Firewall router. So I don't have any first-hand knowledge of Tomato router configuration. But I know that the Tomato router firmware is very popular.

It's inconceivable that any modern router could or would have its administrative access enabled on its WAN interface, thus making it accessible to the Internet. Can you say "RDP?" But even assuming that the WAN interface binding must be manually enabled, in this day and age its unconscionable for firmware to allow =ANY= default login for the WAN-facing side. I'm encountering more and more software which refuses to do things like that. And that's a good thing.

So, in case we have any Tomato router enthusiasts in our midst, I wanted to mention that there's a nasty botnet out there scanning port 8080, the default Tomato WAN admin port, and poking at it to get in. I cannot imagine that any of our listeners would have enabled WAN admin access and left the credentials unchanged. (And, as I said, it s wrong that it's even possible, today.) So if you know anyone who's using the Tomato firmware, you might want to give them a heads-up.

Our web browsers are under attack... via their browser extensions:
The Chrome Web Store is experiencing a wave of fraudulent transactions and has temporarily suspended publishing and updating paid Chrome extensions due to a spike in fraudulent transactions

Get a load of THIS one!... The Google security team has indefinitely suspended the publishing or updating of **any** commercial Chrome extensions on the official Chrome Web Store following a spike in the number of paid extensions engaging in fraudulent transactions.

Let me repeat that: The Google security team has indefinitely suspended the publishing or updating of **any** commercial Chrome extensions on the official Chrome Web Store following a spike in the number of paid extensions engaging in fraudulent transactions.

Google said that the wave of fraudulent transactions began earlier this month. Google engineers described the fraudulent transactions as happening "at scale." And this ban on publishing or updating impacts all paid extensions including Chrome extensions that require paying a fee before installing, extensions that work based on monthly subscriptions, or Chrome extensions that use one-time in-app purchases to get access to various features. Any existing commercial extensions that are already in place are still available for download via the official Chrome Web Store but their developers are now forbidden from pushing any updates for fear that they might

be fraudulent.

Simeon Vincent, Developer Advocate for Chrome Extensions at Google said: “This is a temporary measure meant to stem this influx as we look for long-term solutions to address the broader pattern of abuse.”

Extension developers who try to publish a new paid Chrome extension, or push a new update of their commercial extensions, are currently receiving an automated message that reads: “Spam and Placement in the Store.”

Since this is a blanket ban, big-name extensions have been impacted by this ban, including password manager Dashlane and meeting planner app Comeet. The decision to ban publishing or updating Chrome extension was formally announced late last Friday night, January 24. However, Jeff Johson, the creator of the StopTheMadness Chrome extension, has told ZDNet that Google has been silently blocking updates for paid Chrome extensions for days.

And it's unclear how long the ban will last. Vincent said: “We are working to resolve this as quickly as possible, but we do not have a resolution timeline at the moment. Apologies for the inconvenience.”

And... it's not just Chrome!

Over the past two weeks, Mozilla has banned nearly 200 Firefox add-ons in a crack down on Firefox browser add-on misbehavior. In total, Mozilla's add-on review team banned 197 Firefox add-ons that were caught executing malicious code, stealing user data, or using obfuscation to hide their source code. The add-ons have been banned and removed from the Mozilla Add-on portal to prevent new installs and they have also been disabled in the browsers of the users who had already installed them.

129 of the 197 add-ons were developed by a single developer "2Ring", a provider of business-to-business software. In those 129 cases the ban was triggered because the add-ons were found to be downloading and executing code from a remote server. As we've discussed in the past, for obvious security reasons, Mozilla's updated add-on rules forbid add-ons from obtaining external code. Even if such code was benign now, there's no telling or controlling what any such code might do in the future, and it would represent a very tempting target for supply-chain attacks. So, Mozilla has started strictly enforcing this rule across its entire add-on ecosystem. Another developer had 6 add-ons doing the same thing, and a different developer had three add-ons banned that were determined to be fake premium products.

Bans were also levied against add-ons found to be improperly collecting user data including “WeatherPool”, “Your Social”, “PDFviewer - tools”, “RoliTrade”, and “Rolimons Plus”. And 30 add-ons were banned over various types of banned or malicious behavior. In those cases of misbehavior only the add-on IDs were published so that add-on developers could appeal the ban after curing the misbehavior.

Questionable behavior was also spotted in the FromDocToPDF add-on, which Mozilla engineers found to be loading remote content into Firefox's new tab page. A Firefox add-on named "Fake Youtube Downloader" was also banned for attempting to install malware in users' browsers.

Add-ons like EasySearch for Firefox, EasyZipTab, FlixTab, ConvertToPDF, and FlixTab Search were banned for intercepting and collecting user search terms -- a well-defined no-no and bannable offense.

Mozilla's security staff also banned 14 other add-ons that were caught using obfuscated code. As we've discussed before, while commercial add-on authors ought to have the right to protect their intellectual property, the safety of the browser ecosystem **must** take precedence. So add-on code **must** be auditable by browser repository curators so that they've able to protect their users.

This is all something of a mess, but browser add-ons have become a valuable adjunct to our web browsers. We're going to have to figure out how to make it work.

The NIST publishes a new Privacy Framework
https://www.nist.gov/privacy-framework

As we know, the US federal government's National Institute of Standards and Technology (NIST) has produced several standards and guides to aid organizations. Back in 2016 it published a set of password rules, and it also publishes a Cybersecurity Framework that has become a useful litmus test for those needing to secure their data.

NIST's latest such work is a new Privacy Framework laid out in a 43-page document aimed at helping to protect Internet users' personal privacy.

https://www.nist.gov/system/files/documents/2020/01/16/NIST%20Privacy%20Framework_V1.0.pdf

This Privacy Framework can be used when developing new products and services to ensure that the new work hasn't overlooked something important. And within an organization of any size, being able to state that your private information management complies with the guidance provided by the US government's NIST Privacy Framework is not a bad thing. But -- all bureaucracy aside -- having a checklist to check can be very useful.

In its Executive Summary, the Framework describes itself this way:

For more than two decades, the Internet and associated information technologies have driven unprecedented innovation, economic value, and improvement in social services. Many of these benefits are fueled by data about individuals that flow through a complex ecosystem. As a result, individuals may not be able to understand the potential consequences for their privacy as they interact with systems, products, and services. At the same time, organizations may not realize the full extent of these consequences for individuals, for society, or for their enterprises, which can affect their brands, their bottom lines, and their future prospects for growth.

Following a transparent, consensus-based process including both private and public stakeholders to produce this voluntary tool, the National Institute of Standards and Technology (NIST) is publishing this Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management (Privacy Framework), to enable better privacy engineering practices that support

privacy by design concepts and help organizations protect individuals' privacy. The Privacy Framework can support organizations in:

- Building customers' trust by supporting ethical decision-making in product and service design or deployment that optimizes beneficial uses of data while minimizing adverse consequences for individuals' privacy and society as a whole.

- Fulfilling current compliance obligations, as well as future-proofing products and services to meet these obligations in a changing technological and policy environment; and

- Facilitating communication about privacy practices with individuals, business partners, assessors, and regulators.

Deriving benefits from data while simultaneously managing risks to individuals' privacy is not well-suited to one-size-fits-all solutions. Like building a house, where homeowners make layout and design choices while relying on a well-engineered foundation, privacy protection should allow for individual choices, as long as effective privacy risk mitigations are already engineered into products and services. The Privacy Framework—through a risk- and outcome-based approach—is flexible enough to address diverse privacy needs, enable more innovative and effective solutions that can lead to better outcomes for individuals and organizations, and stay current with technology trends, such as artificial intelligence and the Internet of Things.

The Privacy Framework follows the structure of the Framework for Improving Critical Infrastructure Cybersecurity (Cybersecurity Framework) to facilitate the use of both frameworks together. Like the Cybersecurity Framework, the Privacy Framework is composed of three parts: Core, Profiles, and Implementation Tiers. Each component reinforces privacy risk management through the connection between business and mission drivers, organizational roles and responsibilities, and privacy protection activities.

It goes on at some length (for 43 pages) from there. But I wanted to bring it to the attention of our listeners since it could be extremely useful not only to quiet a bureaucracy by being able to assert that your work is compliant with the NIST Privacy Framework recommendations, but also as a truly useful checklist to make sure that you haven't forgotten anything important.

Hacker Leaks More Than 500K Telnet Credentials for IoT Devices

Port 23 is the so-called "Well Known Port" for the Telnet service. As I'm sure our listeners know, Telnet is the original old-school remote console access service that's been around from the start of the Internet -- thus, its low port number. :) it is such a massive perennial security risk that modern systems won't even allow the Telnet service to be used remotely -- not even for an instant. I've been using FreeBSD UNIX as my preferred non-Windows UNIX platform for years. Back on FreeBSD v5 you had to really struggle to get the system to allow Telnet connections. But I was curious over the holidays when I was setting up GRC's replacement UNIX machine which is running FreeBSD v12.1. There's no way to do it any longer. The OS simply and wisely refuses. The only way now to connect is over port 22 using the Secure Shell -- SSH.

But, not all operating systems are designed correctly. And various IoT devices are the worst offenders. Consequently, news was made last week when the long-time operator of a DDoS For Hire (so-called "booter' service because its boots target off the Internet) published the remote login Telnet username and password combinations -- along with the target IP addresse -- of more than 515,000 devices, of all kinds, on the Internet.

In other words... yeah... the brain-dead Telnet service, providing what is effectively full remote console access, is still actively supported and being served by a huge number of insecure -- and now suddenly much less secure -- Internet-connected devices.

When he was asked why he published such a massive list of "bots," the leaker said he upgraded his DDoS service from working on top of IoT botnets to a new model that relies on renting high-output servers from cloud service providers. In other words... he had apparently moved on to commandeering bigger game and so he thought he'd just drop off a little gift, curbside.

Last week, Leo, you were mentioning your occasional use of GRC's ShieldsUP service. So I'll note that since Telnet runs over TCP, and ShieldsUP checks for and reports on listening TCP ports, anyone can quickly use ShieldsUP to verify that no device on their internal network has surreptitiously used UPnP to open a Telnet port through their routers. As I've said from the start, UPnP is an extremely mixed blessing. In order to work, it needs to be UI-free. That's its whole point. But that also allows anything to open ports without any permission.

A Welcome "Micro Patch"
... for the Windows IE jscript.dll 0-day vulnerability

Since fully disabling the jscript.dll **does** cause problems for some systems, notably Windows Media Player (WMP), I was hoping that we might get a micropatch since this is a perfect use for the micro-patching service being offered at https://0patch.com

https://blog.0patch.com/2020/01/micropatching-workaround-for-cve-2020.html

Their blog posting about this is titled "Micropatching a Workaround for CVE-2020-0674"

Last Friday, Microsoft published an advisory about a remotely exploitable memory corruption vulnerability (CVE-2020-0674) that was reported to them by Qihoo 360 as being exploited in the wild. These attacks were reportedly limited so Microsoft decided not to rush with issuing a patch but will rather provide one as part of February's Patch Tuesday. They did, however, provide a workaround.

Because the provided workaround has multiple negative side effects, and because it is likely that Windows 7 and Windows Server 2008 R2 users without Extended Security Updates will not get the patch at all (their support ended this month), we decided to provide a micropatch that simulates the workaround without its negative side effects.

The vulnerability is in jscript.dll, which is the scripting engine for legacy JScript code; note that all "non-legacy" JScript code (whatever that might be), and all JavaScript code gets executed by the newer scripting engine implemented in jscript9.dll.

Microsoft's workaround comprises setting permissions on jscript.dll such that nobody will be able to read it. This workaround has an expected negative side effect that if you're using a web application that employs legacy JScript (and can as such only be used with Internet Explorer), this application will no longer work in your browser.

There also several other negative side effects:

- Windows Media Player is reported to break on playing MP4 files.
- The sfc (Resource Checker), a tool that scans the integrity of all protected system files and replaces incorrect versions with correct Microsoft versions, chokes on jscript.dll with altered permissions.
- Printing to "Microsoft Print to PDF" is reported to break.
- Proxy automatic configuration scripts (PAC scripts) may not work.

Microsoft's advisory states that the provided workaround will have to be reverted when they issue a patch for jscript.dll. However, note that some additional side effects may result from changing the ownership on jscript.dll (from TrustedInstaller to the user you're implementing the workaround as).

Test Case

Finding a test case that triggers the loading of jscript.dll was not difficult - Google Project Zero has one published in an advisory from November last year. (It won't be at all surprising if the vulnerability at hand is either a clone of that one, or a bypass for its patch - but we'll know soon enough.)

So we took Google's example, simplified it to only cause the loading of jscript.dll for executing a single alert call. Note the important part is to specify JScript.Encode as language. (By the way, read the Wikipedia article on JScript.Encode for a primer on how to underestimate intelligent adversaries.)

```
<html>
<head>
 <meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
</head>
<body>
<script language="Jscript.Encode">
        alert("jscript.dll was loaded");
</script>
</body>
</html>
```

The test was then simple: opening this file in IE without mitigation or patch should result in a popup saying "jscript.dll was loaded". With a patch, we would want the loading of jscript.dll to fail and consequently no popup to appear. Ideally, in a graceful manner without any ugly error messages or crashed processes.

[[They then go into instruction-level detail about which 18 bytes they patched to cause the jscrip.dll to refuse to load. Note that this micro-patch does not fix the problem, but it does prevent its abuse within IE while not interfering with its other legacy uses within the system.]]

They then explain that they have ported this patch to Windows 7 and 10 workstation and server editions. And, promoting their free personal and non-profit service, they add…

If you're a 0patch user, you already have this micropatch downloaded to all your online computers with 0patch Agent, and - depending on your settings - already automatically applied to all processes using the Internet Explorer 11 engine for rendering content. This includes Internet Explorer (obviously), Microsoft Word, Microsoft Outlook, and a variety of other applications.

As all our micropatches, you can switch this micropatch on or off and have it instantly applied to, or removed from running applications - effectively making it a "kill switch" for jscript.dll.

Frequently Asked Questions

Q: Why would we apply your micropatch instead of Microsoft's recommended workaround?

Our micropatch is designed to avoid negative side effects of Microsoft's workaround (see above). It can also be easily reverted (un-applied) with a switch of a button without leaving any traces, while the workaround changes the ownership on jscript.dll.

Q: Will Microsoft provide a patch for CVE-2020-0674 to Windows 7 and Windows Server 2008 R2 users without Extended Security Updates?

We don't know but these systems are now officially out of support and Microsoft has historically only issued security patches for unsupported systems in extreme cases (e.g., the Shadow Brokers leak, or BlueKeep / CVE-2019-0708).

We at 0patch have committed to provide post-end-of-service security micropatches for Windows 7 and Windows Server 2008 R2 for three additional years, which is why we're also issuing this micropatch for these platforms. (Read FAQ for more information.)

Q: What will happen if I apply your micropatch, and then apply Microsoft's patch for CVE-2020-0674 when it comes out?

When Microsoft issues a patch for this vulnerability, we'll inspect it and decide whether to replace our current micropatch (which resides in mshtml.dll and disables jscript.dll entirely) with a more targeted micropatch in jscript.dll (which will only fix that vulnerability but keep jscript.dll available). It might happen that we do so on supported Windows platforms but keep the current micropatch on Windows 7 and Windows Server 2008 R2. This also depends on user feedback (i.e., whether our micropatch causes anyone problems).

In any case, you won't have to do anything - Microsoft's patch will have precedence over our micropatch.

Miscellany

- Star Wars
- Picard

SpinRite

- https://grc.sc/roadmap

SHAmbles

https://sha-mbles.github.io/
https://eprint.iacr.org/2020/014.pdf

This is the duo, Gaëtan Leurent and Thomas Peyrin, who have been methodically pounding on the aging SHA-1 hash for several years. They have come to know it extremely well. And they have finally taken SHA-1 from weakened and worrisome to demonstrably broken.

Their just-released paper is titled: "SHA-1 is a Shambles. First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust."

Abstract. The SHA-1 hash function was designed in 1995 and has been widely used during two decades. A theoretical collision attack was first proposed in 2004, but due to its high complexity it was only implemented in practice in 2017, using a large GPU cluster. More recently, an almost practical chosen-prefix collision attack against SHA-1 has been proposed. This more powerful attack allows to build colliding messages with two arbitrary prefixes, which is much more threatening for real protocols.

In this paper, we report the first practical implementation of this attack, and its impact on real-world security with a PGP/GnuPG impersonation attack. We managed to significantly reduce the complexity of collisions attack against SHA-1: on an Nvidia GTX 970, identical-prefix collisions can now be computed with a complexity of 2^61.2 rather than 2^64.7, and chosen-prefix collisions with a complexity of 2^63.4 rather than 2^67.1.

When renting cheap GPUs, this translates to a cost of 11k US$ for a collision, and 45k US$ for a chosen-prefix collision, within the means of academic researchers.

Our actual attack required two months of computations using 900 Nvidia GTX 1060 GPUs (we paid 75k US$ because GPU prices were higher, and we wasted some time preparing the attack).

Therefore, the same attacks that have been practical on MD5 since 2009 are now practical on SHA-1. In particular, chosen-prefix collisions can break signature schemes and handshake security in secure channel protocols (TLS, SSH). We strongly advise to remove SHA-1 from those type of applications as soon as possible. We exemplify our cryptanalysis by creating a pair of PGP/GnuPG keys with different identities, but colliding SHA-1 certificates. A SHA-1 certification of the first key can therefore be transferred to the second key, leading to a forgery. This proves that SHA-1 signatures now offers virtually no security in practice. The legacy branch of GnuPG still uses SHA-1 by default for identity certifications, but after notifying the authors, the modern branch now rejects SHA-1 signatures (the issue is tracked as CVE-2019-14855).

1 Introduction

Cryptographic hash functions are present in countless security applications and protocols, used for various purposes such as building digital signature schemes, message authentication codes or password hashing functions. In the key application of digital signatures for example, hash functions are classically applied on the message before signing it, in order to improve efficiency and provide security guarantees. Informally, a cryptographic hash function H is a function that maps an arbitrarily long message M to a fixed-length hash value (we denote n its bit size). Collision resistance is the main security property expected from a hash function: it should be hard for an adversary to compute a collision, aka two distinct messages M and M0 that map to the same hash value H(M) = H(M0), where by “hard” one means not faster than the generic 2^n/2 computations birthday attack.

A cryptanalyst will try to find a collision for the hash function at a reduced cost, but ad-hoc collision attacks are hard to exploit in practice, because the attacker has then usually little control over the value of the actual colliding messages (in particular where the differences are inserted, which are the interesting parts when attacking a digital signature scheme for example). Thus, one can consider stronger and more relevant variants of the collision attack in practice, such as the so-called chosen-prefix collision or CP collision: two message prefixes P and P0 are first given as challenge to the adversary, and his goal is to compute two messages M and M0 such that H(P+M) = H(P0+M0). With such ability, the attacker can obtain a collision even though prefixes can be chosen arbitrarily (and thus potentially contain some meaningful information). A CP collision can also be found generically with 2^n/2 computations (thus 2^80 for a 160-bit hash function like SHA-1), but ad-hoc CP collision attacks are much more difficult to find than plain collision attacks, because of the random and completely uncontrolled internal differences created by the prefixes. Yet, a CP collision attack was found for the MD5 hash function, eventually leading to the creation of colliding X.509 certificates, and later of a rogue Certificate Authority (CA). CP collisions have also been shown to break important internet protocols, including TLS, IKE, and SSH, because they allow forgeries of the handshake messages.

Largely inspired by MD4 and then MD5, SHA-1 is one the most famous cryptographic hash functions in the world, having been the NIST and de-facto worldwide hash function standard for nearly two decades. It remained a NIST standard until its deprecation in 2011 (and disallowed to be used for digital signatures at the end of 2013). Indeed, even though his successors SHA-2 or SHA-3 are believed to be secure, SHA-1 has been broken by a theoretical collision attack in 2004 [WYY05]. Due to its high technicality and computational complexity (originally estimated to about 2^69 hash function calls), this attack was only implemented in practice in 2017, using a large GPU cluster. Because it took more than a decade to compute an actual collision and

because plain collisions are difficult to use directly to attack a protocol, the on-field SHA-1 deprecation process has been quite slow in practice and one can still observe many uses of SHA-1 in the wild unfortunately, migration being expensive. Web browsers have recently started to reject certificates with SHA-1 signatures, but there are still many users with older browsers, and many protocols and software that allow SHA-1 signatures. As observed in, it is still possible to buy a SHA-1 certificate from a trusted CA, many email clients accept a SHA-1 certificate when opening a TLS connection, and SHA-1 is also widely supported to authenticate TLS handshake messages.

Very recently, a CP collision attack against SHA-1 has been published, which requires an estimated complexity between 2^66.9 and 2^69.4 SHA-1 computations. It works with a two-phase strategy: given the challenge prefix and the random differences on the internal state it will induce, the first part of the attack uses a birthday approach to limit the internal state differences to a not-too-big subset. From this subset, reusing basic principles of the various collision search advances on SHA-1, one slowly adds successive message blocks to come closer to a collision, eventually reaching the goal after a dozen blocks. Even though these advances put the CP collisions within practical reach for very well-funded entities, it remains very expensive to conduct and also very difficult to deploy as the attack contains many very technical parts.

1.1 Our Contributions

In this article, we exhibit the very first chosen-prefix collision against the SHA-1 hash function, with a direct application to PGP/GnuPG security.