Security Now! #774 - 07-07-20
‘123456’

This week on Security Now!

This week we look at two new just released emergency Windows 10 updates, and the new and circuitous path they will need to take to get to their users. We look at a slick new privacy feature coming to iOS 14 and how it is already cleaning up prior behavior. We'll take our annual survey of the rapidly growing success of the HackerOne program, and also note the addition of a major new participant in their bug bounty management. We briefly note the latest American city to ban the use of facial recognition for law enforcement, but we mostly examine the result of NIST's analysis of demographic bias in facial recognition outcomes. We'll look at a new high-velocity vulnerability and exploitation, close the loop with a couple of listeners and I'll share an interesting bit of work on SpinRite's AHCI controller benchmarking. Then we’ll conclude by discussing the mysterious meaning of this week’s episode title: “123456.”

This is just **TOO** perfect!

Security News

US-CERT notes two Emergency Windows Updates:
US-CERT posted last Tuesday: On June 30, 2020, Microsoft has released information regarding vulnerabilities (CVE-2020-1425, CVE-2020-1457) in Microsoft Windows Codecs Library. This contains updates that are rated as "Critical". Remote attackers leveraging these vulnerabilities may be able to execute arbitrary code. For more information on the vulnerabilities, please refer to the information provided by Microsoft.

Both vulnerabilities have the same name, differing only in their CVE's:
CVE-2020-1425 | Microsoft Windows Codecs Library Remote Code Execution Vulnerability
CVE-2020-1457 | Microsoft Windows Codecs Library Remote Code Execution Vulnerability

And the disclosures are almost identical as well:
https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/CVE-2020-1425
https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/CVE-2020-1457

By this point our listeners are no longer surprised to learn of a fatal flaw in a media codec. As we know, codecs are complex interpreters of a compressing encoder's metadata. It's truly difficult to make a codec both screamingly fast and careful at the same time. Being super careful means checking everything, and checking everything takes precious time when a codec is, by its very nature, often racing the clock.

So what made these stand out, aside from the fact that they were once again patches for an out-of-cycle critical remote code execution vulnerability and an information disclosure vulnerability, was the fact that Microsoft indicated that the updates would **not** be available through Windows Update, nor through the Windows Update Catalog. No... these updates would be provided through the Microsoft Store.

https://support.microsoft.com/en-us/help/4026259/microsoft-store-get-updates-for-apps-and-games

Users are to click on the little white shopping bag on the taskbar (I'll note that none of my Windows 10 taskbars have little white shopping bags), then select: “More” > “Downloads and updates” > “Get updates”.

In their disclosure, Microsoft wrote:
“A remote code execution vulnerability exists in the way that Microsoft Windows Codecs Library handles objects in memory. An attacker who successfully exploited the vulnerability could execute arbitrary code.”

“A remote code execution vulnerability exists in the way that Microsoft Windows Codecs Library handles objects in memory. An attacker who successfully exploited this vulnerability could obtain information to further compromise the user's system.”

And in either case: “Exploitation of the vulnerability requires that a program process a specially crafted image file The update addresses the vulnerability by correcting how Microsoft Windows Codecs Library handles objects in memory.”

Microsoft writes that: “Affected users will be automatically updated by Microsoft Store. And, according to Microsoft, users who want to receive the update immediately can check for updates with the Microsoft Store App.”

I suppose it makes sense for Store apps and extensions, even when they are sourced by Microsoft, to be updated through the channel that was used for their original delivery. Especially in the case of third party apps being updated, Microsoft would not want to be hosting updates of those through its own operating system and app update channels. So the Store it is.

Both updates were privately reported and are not known to be used in the wild, so they were note 0-days. But now that they are out the race is on.

The problems exist in the HEVC video extensions available for $0.99 from the Microsoft Store. The HEVC extension rates 2.5 out of 5 stars and Microsoft’s description reads: “Play High Efficiency Video Coding (HEVC) videos in any video app on your Windows 10 device. These extensions are designed to take advantage of hardware capabilities on some newer devices—including those with an Intel 7th Generation Core processor and newer GPU to support 4K and Ultra HD content. For devices that don’t have hardware support for HEVC ”

For anyone who’s interested, Wikipedia explains about HEVC: “High Efficiency Video Coding (HEVC), also known as H.265 and MPEG-H Part 2, is a video compression standard designed as part of the MPEG-H project as a successor to the widely used Advanced Video Coding (AVC, H.264, or MPEG-4 Part 10). In comparison to AVC, HEVC offers from 25% to 50% better data compression at the same level of video quality, or substantially improved video quality at the same bit rate.”

For anyone who’s curious to know whether a system might have the HEVC Video Extensions installed, an if so which version, the following PowerShell command will tell you:

```
Get-AppxPackage -Name Microsoft.HEVCVideoExtension
```

The repaired version of the HEVC extensions is **1.0.31822.0** or **1.0.31823.0**. SInce I don’t have either extension, my PowerShell command just exited, returning nothing.

Some commentators have observed that this new channel for releasing critical updates outside of the normal Windows security update distribution channels, while as I noted makes sense and is understandable, can cause trouble in enterprise settings where certain Windows features -- and Windows Store probably first and foremost -- have been deliberately disabled. For such companies who have purposefully disabled the Microsoft Store and Microsoft Store automatic app updates, vulnerable computers will not receive the fixes without removing this policy.

ComputerWorld’s industry fixture Woody Leonard, over in his “AskWoody” column was far less patient and understanding than I am about this new source for Windows updates:
https://www.askwoody.com/2020/those-two-weird-microsoft-store-fixes-for-windows-security-flaws-keep-getting-stranger/

One of the replies to his posting notes that: “The optional HEVC codec exists by default in Windows Client editions since version 1809, except N and LTSC editions.” I have LTSC, so that’s probably why my PowerShell query came up blank. But assuming that's the case, it would be possible for any normal Windows 1809, 1903, 1909 & 2004 to have the vulnerable codec installed, yet to be unable to have it updated if a user or an enterprise had determined that they had no interest in the Windows Store. It's exactly the same as if we could uninstall Windows Update.

Woodly winds up his post by writing: "The distribution method is riddled with all sorts of obvious holes – I mean, anybody with any sort of updating experience should've been able to compile a list of a half dozen ways that this could go wrong. Yet another unholy mess."

A slick new iOS 14 feature catches Linked-In red handed!

When iOS 14 is officially released in the fall, some users may discover unexpected — and unwanted — behavior from some of their iOS apps. Apple has added a slick new privacy feature. It simply shows a popup notification when any app reads the content from their clipboard. It's so simple yet so powerful.

And the pre-release beta has caught a surprising number of other iOS apps quietly observing the user's global clipboard... which is **none** of their business!

News: ABC News, Al Jazeera English, CBC News, CBS News, CNBC, Fox News, News Break, New York Times, NPR, ntv Nachricten, Reuters, Russia Today, Stern Nachrichten, The Economist, The Huffington Post, The Wall Street Journal, Vice News

Games: 8 Ball Pool, AMAZE!!!, Bejeweled, Block Puzzle, Classic Bejeweled, Classic Bejeweled HD, FlipTheGun, Fruit Ninja, Golfmasters, Letter Soup, Love Nikki, My Emma, Plants vs Zombies Heroes, Pooking – Billiards City, PUBG Mobile, Tomb of the Mask, Tomb of the Mask: Color, Total Party Killer, Watermarbling

Social: TikTok, ToTalk, Truecaller, Viber, Weibo, Zoosk

Miscellaneous: 10% Happier: Meditation, 5-0 Radio Police Scanner, Accuweather, AliExpress Shopping App, Bed Bath & Beyond, Dazn, Hotels.com, Hotel Tonight, Overstock, Pigment – Adult Coloring Book to Color, Sky Ticket, The Weather Network

This all first came to light about two weeks ago when the Chinese app "TikTok" was caught reading the content of its users' clipboards at short and regular intervals. TikTok claimed that the feature was part of a fraud detection mechanism and that the company never stole the clipboard content, but they promised to remove the behavior nevertheless to put users' minds at ease. Yes, please.

And then last week, as developer/users continued experimenting with the pre-release iOS 14 clipboard access detection system, a developer from the portfolio-building portal Urspace.io discovered that the LinkedIn iOS app was doing this, too.

In a video that he shared via Twitter, the Urspace developer showed how LinkedIn's app was reading the clipboard content after every user key press, even accessing the shared clipboard feature that allows iOS apps to read content from a user's macOS clipboard.

He noted that LinkedIn was not only copying the contents of his clipboard with every keystroke, but that since iOS supports a cross-device copy and paste, LinkedIn was copying the clipboard contents of his MacBook Pro via his iPad Pro.

When LinkedIn was asked by the tech press what the heck was going on, LinkedIn's spokesperson claimed that the behavior was a bug, and was not intended behavior. And in a further effort to further quell the growing concern, Erran Berger, LinkedIn's VP Engineering of Consumer Products attempted to clarify the issue, writing on Twitter: “Appreciate you raising this. We've traced this to a code path that only does an equality check between the clipboard contents and the currently typed content in a text box. We don't store or transmit the clipboard contents. We will follow up once the fix is live in our app.” So, what’s interesting is that whatever it is that this was doing, apparently it’s not that necessary, such that if users were to be made aware of it — as is starting to happen now — it's possible to simply change that behavior. Hmmmm.

So the lesson here is that simply notifying users of something that's going on behind their backs without their knowledge, permission or understanding can go a long way toward cleaning up and eliminating that behavior.

Big props to Apple for this one. What a slick and welcome solution to a serious privacy threat.

HackerOne shares their top 10 public bug bounty programs
Last year we looked at HackerOne's top 10 bug bounty programs to see which companies were paying the biggest and/or most frequent bounties.

Now, a year later, we have HackerOne's update for 2020. Many of the names one the top 10 list are the same, some have moved, and a few new entrants have appeared.

Verizon Media held the first place position last year and they are again solidly — very solidly — in the top slot. Verizon Media runs, by far, the most active and successful bug bounty program. Compared to last year, Verizon increased their annual bounty payouts by $1.4 million from $4 million paid out last year to $5.4 million paid out in the most recent year. Moreover, just one of Verizon Media's bug bounties ranks among the top 5 largest payouts ever handed out through HackerOne: $70,000 handed to an enterprising researcher.

Everyone knows PayPal. And I'm delighted to see that they are maintaining an active bug bounty program. We talked a lot in the past about how difficult it is for in-house developers to discover their own problems. Bug hunting is inherently adversarial. PayPal is not a newcomer. Last year they took the #3 spot. But this year they have replaced Uber to take #2. Unlike Verizon whose HackerOne program launched in February of 2014, PayPal joined the game much more recently, in August of 2018. But nevertheless, PayPal quickly established itself as one of the most active companies on the platform. Over the past two years they have paid out a total of nearly $2.8 million, with a bit more than half of that, $1.62 million, in the past year.

Although Uber slipped from its #2 spot in the previous accounting with a significantly leaner most recent year, their strong early start back in December of 2014 has kept them near the top of the pack. In the most recent year Uber's security team awarded $620,000 in bug bounties, bringing the company's all time program total to $2,415,000. And Uber's bug bounty program ranks in the top 5 most thanked hackers, the top 5 most reports resolved, and the top 5 highest bounty paid rankings.

With all of their highly publicized recent troubles (and their very deep pockets), I suppose we should not be surprised to see Intel moving up two places from their #6 ranking in 2019 to the #4 slot today. But then, paying out more than $1 million in bug bounties to researchers in the past 12 months will have that effect.

And although the exact amount has remained a closely held secret, it is known that Intel has the sole distinction of having paid the highest bug bounty ever paid on the HackerOne platform. The single payout sum is assumed to fall somewhere between $100,000 and $200,000 and, you guessed it... it for a side-channel vulnerability affecting its CPU microarchitectures.

Twitter is, if nothing else, steady. They were #5 last years and #5 again this year. Running one of the older programs on HackerOne, starting back in May of 2014, Twitter has paid out over $1,288,000 in bounties to security researchers, with $118,000 of that sum distributed in the past 12 months.

GitLab has jumped from the 10th ranking in the previous line-up to hold this year's #6 spot. GitLab was also one of the early entrants, joining in June of 2014. So they've enjoyed a rather quiet start. Across all of the six previous years they paid out a total of $570,000 in bounties. But then in just this most recent 12 months they paid $641,000, $71,000 more than all of the previous years combined, bringing their total payouts to $1,211,000. And GitLab also has one of the fastest response times on HackerOne, amazingly responding to security researchers within an hour, on average, to new bug reports.

And for the first time, Mail.ru joins to top 10, moving inn a single year from the 14th slot into the #7 position. Much like GitLab, they have been a member of hackerOne since April 2014, but their most recent year's bounty payouts totaling $819,000 dwarfed the $300,000 paid out through all previous years. Mail.ru's bug bounty program also ranks in the top 5 most thanked hackers ranking (973 thanked hackers) and the top 5 most reports resolved (3,333 resolved reports).

Everyone's favorite GitHub is also making recent upward moves. This year's bounty payouts jumped them from last year's 11th rank into the number 8 slot. And the pattern repeats with the most recent year's payouts nearly matching the sum of all previous years. Github paid a total of $467,000 over the past 12 months to security researchers for their responsibly reported bugs. This brings GitHub's total since joining HackerOne in April of 2016 to $987,000.

And holding steady with its 9th place ranking we have Valve. It's been pretty steady in its payouts. In the most recent year Valve paid $381,000 in bounties to bug hunters which brought its lifetime program total to $951,000.

And last (and possibly least considering the effects of COVID-19) we have Airbnb. As the previous summaries have shown, the bug bounty market is rapidly growing and heating up. So despite having awarded more than $344,000 in bug bounties over the last 12 months, Airbnb's HackerOne competitive ranking slipped three rungs from its comfortable 7th spot last year.

Overall, Airbnb has awarded a respectable $944,000 in bug bounties since it initiated its program in February of 2015. And their software is all the better for it.

I think what these numbers reveal, as I just observed, is that bug bounties are finally becoming mainstream essential components of any mature business whose software creates a privacy or security exposure for a company, its employees or its customers.

And while we're on the topic of Bug Bounties... a couple of weeks ago...

Sony launches PlayStation bug bounty program with rewards of $50K+
Sony will pay security researchers for bugs in the PlayStation 4 gaming console, its operating system, official PS4 accessories, but also the PlayStation Network and related websites.

Following the now well established “responsible disclosure” model, Sony will reward security researchers who discover bugs in PlayStation-related devices and websites and report them to Sony's security team so that they can be patched before being exploited.

And unlike Microsoft and Nintendo who both top-out their bounties at only $20,000, Sony has said that it plans to pay security researchers between $100 and up to $50,000 (or even higher) for vulnerabilities reported in the company's products. Eligible targets of opportunity include the Sony PlayStation 4 gaming console, its operating system, official PS4 accessories, the PlayStation Network and related websites. And, yes, wisely, Sony's new vulnerability rewards program (VRP) will also be managed through HackerOne.

Prior to taking their VRP program public, Sony had been running a private in-house invite-only vulnerability rewards program since last year. And we know that the world of gaming has historically been a target rich environment. Hackers tend to heavily target gaming accounts, which are usually abused for fraud or put up for sale online, on underground hacking forums. And this past April hackers abused a vulnerability in an old Nintendo authentication mechanism to hijack more than 300,000 accounts. Sony also has the sort of deep pockets that makes the creation of a bug bounty program a “just do it” no-brainer. I'm delighted they are.

Boston bans facial recognition
The city of Boston just joined the Gronw number of cities to just say no to law enforcement's use of facial recognition. This makes Boston the second largest city, behind only San Francisco, to decide that the technology's current profound limitations and liabilities far outweigh its benefits.

Just what exactly are such system's limitations and liabilities? I'm annoyed by the idea that the results of tests to detect bias might themselves be biased. It's one thing to not like the idea and to find it to be creepy. But that alone doesn't mean that it doesn't work. The good news is, the US National Institute of Standards and Technology (our NIST) conducted their own analysis, the results from which were published late last year. And they are, indeed, quite eye opening.

NIST posed itself the question: “How accurately do face recognition software tools identify people of varied sex, age and racial background?” Not surprisingly, NIST found that the answer depends upon the algorithm at the heart of the system, the application that uses it and the data it's fed. However, they did also determine that the majority of face recognition algorithms exhibit strong demographic differentials. In this usage the term “differential” means that an algorithm’s ability to match two images of the same person varies between demographic groups.

NIST's results, which were captured in the report, “Face Recognition Vendor Test (FRVT) Part 3: Demographic Effects (NISTIR 8280)”, were and are intended to inform policymakers and to help software developers better understand the performance of their algorithms.

The report quoted Patrick Grother, a NIST computer scientist and the report’s primary author. Patrick stated: “While it is usually incorrect to make statements across algorithms, we found empirical evidence for the existence of demographic differentials in the majority of the face recognition algorithms we studied. While we do not explore what might cause these differentials, this data will be valuable to policymakers, developers and end users in thinking about the limitations and appropriate use of these algorithms.”

The study was conducted through NIST’s Face Recognition Vendor Test (FRVT) program, which evaluates face recognition algorithms submitted by industry and academic developers on their ability to perform different tasks.

The NIST study evaluated 189 software algorithms from 99 developers which represents a large majority of the industry. It focuses on how well each individual algorithm performs one of two different tasks that are among face recognition’s most common applications. The first task, confirming a photo matches a different photo of the same person in a database, is known as “one-to-one” matching and is commonly used for verification work, such as unlocking a smartphone or checking a passport. The second, determining whether the person in the photo has any match in a database, is known as “one-to-many” matching and can, in theory at least, be used for identification of a person of interest.

To evaluate each algorithm’s performance on its task, the team measured the two classes of error the software can make: false positives and false negatives. A false positive means that the software wrongly considered photos of two different individuals to show the same person, while a false negative means the software failed to match two photos that are of the same person.

These distinctions are obviously important because the class of error and the type of search performed can carry vastly different consequences depending on the real-world application.

Patrick said: “In a one-to-one search, a false negative might be merely an inconvenience — for example, you can’t get into your phone, but the issue can usually be remediated by a second attempt. But a false positive in a one-to-many search puts an incorrect match on a list of candidates that warrant further scrutiny.”

NIST's description of this noted that the thing that sets the publication apart from most other face recognition research is its concern with each algorithm’s performance when considering demographic factors. For one-to-one matching, only a few previous studies have explicitly explored demographic effects; for one-to-many matching, none have.

So, here are some details:

To evaluate the algorithms, the NIST team used four collections of photographs containing 18.27

million images of 8.49 million people. All came from operational databases provided by the State Department, the Department of Homeland Security and the FBI. The team did not use any images “scraped” directly from internet sources such as social media or from video surveillance.

The photos in the databases included metadata information indicating the subject’s age, sex, and either race or country of birth. Not only did the team measure each algorithm’s false positives and false negatives for both search types, but it also determined how much these error rates varied among the tags. In other words, how comparatively well did the algorithm perform on images of people from different groups?

Tests showed a wide range in accuracy across developers, with the most accurate algorithms producing many fewer errors. In other words, there's highly accurate facial recognition and quite crappy facial recognition. The study focused upon the performance of individual algorithms and was able to reach five broad findings:

1. For one-to-one matching, the team saw higher rates of false positives for Asian and African American faces relative to images of Caucasians. The differentials often ranged from a factor of 10 to 100 times, depending on the individual algorithm. False positives might present a security concern to the system owner, as they may allow access to impostors.

2. Among U.S.-developed algorithms, there were similar high rates of false positives in one-to-one matching for Asians, African Americans and native groups (which include Native American, American Indian, Alaskan Indian and Pacific Islanders). The American Indian demographic had the highest rates of false positives.

3. However, a notable exception was for some algorithms developed in Asian countries. There was no such dramatic difference in false positives in one-to-one matching between Asian and Caucasian faces for algorithms developed in Asia. Patrick Grother emphasized that the NIST study did not explore the relationship between cause and effect, one possible connection, and area for research, is the relationship between an algorithm’s performance and the data used to train it. He wrote: “These results are an encouraging sign that more diverse training data may produce more equitable outcomes, should it be possible for developers to use such data.”

4. For one-to-many matching, the team saw higher rates of false positives for African American females. Differentials in false positives in one-to-many matching are particularly important because the consequences could include false accusations. (In this case, the test did not use the entire set of photos, but only one FBI database containing 1.6 million domestic mugshots.)

5. However, not all algorithms give this high rate of false positives across demographics in one-to-many matching, and those that are the most equitable also rank among the most accurate. This last point underscores one overall message of the report: Different algorithms perform differently.

So not all facial recognition algorithms are created equally, nor do they currently treat everyone equally. This suggests that it's dangerous to lump all facial recognition into a single performance category. It also means that if at some future time we decide that the technology has improved to the point where it is no longer mostly a liability, any solution that is proposed will need to be carefully tested for bias. So, overall, I think an important lesson was learned.

F5 Networks patches a highest-severity vulnerability
Last Friday the 3rd, F5 Networks released a patch for a super-critical vulnerability with a CVSS rating of 10 out of 10. All of their so-called "BIG-IP" networking devices running application security servers are vulnerable to a remote takeover.

BIG-IP devices are used in government networks, on the networks of internet service providers, inside cloud computing data centers, and they're widely deployed across enterprise networks. The devices are so powerful and popular that on its website, F5 claims that 48 of the 50 companies included in the Fortune 50 list rely on BIG-IP systems.

According to Mikhail Klyuchnikov (kly-ooch-ni-kov), a security researcher at Positive Technologies who discovered the flaw and reported it to F5 Networks, the issue resides in a configuration utility called Traffic Management User Interface (TMUI) for BIG-IP application delivery controller (ADC).

This Application Delivery Controller is used by large enterprises, data centers, and cloud computing environments since it allows them to implement application acceleration, load balancing, rate shaping, SSL offloading, and a web application firewall. In other words... it's the Internet-facing big iron hardware that all of an organization's traffic passes through. And, since one of its jobs is SSL offloading, that means it's performing the TLS connection decryption and can see plaintext on the non-Internet edge.

So what we have learned is that an unauthenticated attacker can remotely exploit this vulnerability by sending a maliciously crafted HTTP request to the vulnerable server hosting the Traffic Management User Interface (TMUI) utility for BIG-IP configuration.

And I'll just say once again, never, never, never, expose any sort of privilege-requiring management interface to the public Internet. Never, never, never. Find some way not to. Unfortunately, a Shodan search reveals that about 8500 major organizations and governments have not heeded that advice and were wide open to this remote exploitation at the time of its disclosure last Friday.

And, successful exploitation of this vulnerability could allow attackers to gain full admin control over the device, eventually making them do any task they want on the compromised device without any authorization.

(Kly-ooch-ni-kov) said that "The attacker can create or delete files, disable services, intercept information, run arbitrary system commands and Java code, completely compromise the system, and pursue further targets, such as the internal network. Remote Code Execution in this case results from security flaws in multiple components, such as one that allows directory traversal exploitation."

As for where the devices are, 40% reside in the United States, 16% in China, 3% in Taiwan, 2.5% in Canada and Indonesia and less than 1% in Russia.

So that was last Friday. Apparently, security researchers were intrigued enough by the possibility of this exploitation to skip their 4th of July holiday (if they were in the US) and instead spent and weekend developing working proof-of-concept exploits showing just how easily these devices could be exploited ... because those PoCs began appearing on the Internet by Sunday.

And the next day, the attacks began. The cyber-security community did expect that this bug would come under active attacks as soon as hackers figured out how they could exploit it.

One such researcher tweeted: "The urgency of patching this cannot be understated. I worked for F5 for a decade; they power cell carriers, banks, Fortune 500 and many governments. If deployed correctly the management interface should not be exposed to the Internet, but @binaryedgeio returns 14k hits for 'tmui' so YMMV."

And the NCC Group's security researcher, Rich Warren, who is currently operating BIG-IP honeypots, said he detected malicious attacks coming from five different IP addresses.

So this is a bad one, but it's a great case in point: Never never never expose management interfaces to the public Internet. Make certain to have open lines of communication to the vendors of the critical devices you use so that you can and will receive notifications of critical vulnerabilities. And take action on them with all possible speed. The game has changed over the course of the past decade. It's no longer "are there any vulnerabilities?" It's now: "Who is quicker: The patcher or the exploiter?"

Miscellany

Goodbye YouTube TV
Hello SlingTV. Google's price jump put me off. I watch not-a-one of the newly added channels, so I was getting zero value. I do use YouTube's DVR feature, but for an additional $5 on top of Sling's $30 per month I get 50 hours of DVR, which is all I need.

Closing The Loop

Kevin Morris | @KevMorris_Today | Monday, 8:50pm
"Hi Steve. I've been experimenting with various flavors of Linux and once I burned an ISO to my Flash drive, I always had a devil of a time being able to use it again. InitDisk solved the problem! Thank you for that. Kevin Morris, Santa Clara, CA"

InspClousseau | @InspClousseau | Saturday, 4:48pm
I just found out about this: Thinkst has an open source version of its Canary which runs on a Raspberry Pi. I'm trying to install it now.

https://github.com/thinkst
https://github.com/thinkst/opencanary
"OpenCanary is a daemon that runs several canary versions of services that alerts when a service is

(ab)used.”

Prerequisites:

- Python 2.7, 3.6
- [Optional] SNMP requires the python library scapy
- [Optional] RDP requires the python library rdpy
- [Optional] Samba module needs a working installation of samba

Install:

- For updated and cleaner documentation, please head over to http://opencanary.org
- Installation on Ubuntu:

-
  ```
  $ sudo apt-get install python-dev python-pip python-virtualenv
  $ virtualenv env/
  $ . env/bin/activate
  $ pip install opencanary
  $ pip install scapy pcapy # optional
  ```

SpinRite

AHCI driver performance

As we know, the next release of SpinRite will be bypassing the BIOS and bringing its own maximum performance mass storage device drivers to bear. With all of our AHCI testers reporting initial success with the driver seeing their drives, a couple of days ago I implemented its first use of the bulk data transfer that SpinRite will be using. It transfers maximum-size 32 megabyte blocks of 65,536 sectors from the drive being used. For the initial benchmarking test I wanted to measure the time required to transfer 1 gigabytes of data, so that's 32 of these 32 megabyte blocks. I would initiate a transfer with the AHCI controller, wait for it to issue a hardware interrupt signaling its completion, then immediately initiate the next transfer.

The code worked right off the bat, which always makes me a bit suspicious. But I posted it for the gang to test. Right away we began noticing that the numbers weren't making sense. They were lower than they should be, especially for the faster devices, such as SSDs.

Then one of our testers switched his system from AHCI back to IDE mode and ran the earlier benchmark I had completed back in 2013 before I interrupted this work for SQRL. And THAT benchmark definitely maxed-out his drive and SATA link. That benchmark was doing exactly the same thing — initiating a transfer, waiting for a completion hardware interrupt, then starting the next transfer. But, it was NOT using the super-fancy AHCI controller. In IDE (sometimes referred to as compatibility mode) it was bypassing the AHCI controller's features and performing what's known as a “Bus Mastering” DMA transfer. And that blazed.

So I turned my attention back to the AHCI solution. The first thing I did was to try timing the transfer of just a single 32 megabyte block. That's problematic due to device caching. A 1 gigabyte transfer is guaranteed not to be cached. But 32 megabytes might be. But I was doing all of this work with an SSD, so that didn't matter.
What I found was that a single 32 megabyte block was giving MUCH higher calculated byte

transfer rates. That suggested that the trouble was inter-transfer overhead — the time being taken from the end of one block transfer to initiating the next one. And this fit the symptoms, since it was the faster devices, the SSDs, where we seemed to be returning values that were the most off. That would be because the interblock interval would be consuming a larger percentage of the whole when the transfers themselves were super-short.

So the next thing I did was to transfer only one block, but to also eliminate the interrupt service overhead by “spinning” on the AHCI controller's completion status. Rather than halting the thread and waiting for a hardware interrupt to reawaken, I “spun” (as it called) in a tight loop doing nothing by waiting for the completion status to be signalled. That way, the INSTANT it flipped to “done” I would stop the timer and calculate the time required. And, sure enough, once again I got a better result. So then I switched back to a multi-block transfer while still using an interrupt-free spin loop, and I got the best results so far.

But I was still not getting the performance that I was getting if I switched back to much simpler IDE Bus Mastering. And now I know why: The secret is queuing.

[...]

‘123456’

What is '123456'?
It's not a fibonacci sequence, that would be 1, 1, 2, 3, 5, 8.
It's not a linear regression, but it IS a linear transgression... Because, believe it or not, still, in this day and age, “123456” comprises one out of every 142 passwords found on the Internet.

Yes, in one of the largest password re-use studies of its kind, the password “123456” was found to occur 7 million times across a massive data trove containing one billion leaked credentials.

As we know, the number of leaked credential database collections continues to grow as new companies continue getting hacked and their databases exfiltrated. They are eventually made available online at GitHub or GitLab, or distributed on hacking forums and file-sharing portals.

And some good use is being made of them. Responsible tech companies like Google, Microsoft and Apple have collected leaked credentials to create in-house alert systems that warn users when they're utilizing a "weak" or a "common" password. And as we know, Troy Hunt's “Have I been Pwned” online service relies upon submissions of these leaked credential databases.

So... Last month a Turkish student studying at a university in Cyprus, decided to download and analyze more than one billion of these leaked credentials.

His primary discovery was that the 1,000,000,000+ credential dataset contained a startlingly high count of duplicates. Or stated another way, among more than 1 Billion passwords, only 168,919,919 unique passwords occurred. And of those nearly 169 million unique passwords, more than 7 million were: “123456.”
This means that one out of every 142 passwords included in the analyzed sample was the #1

weakest password known today. Additionally, the research also revealed that the average password length is 9.48 characters. That's not great, but at least it's not six. Since I invariably use LastPass to synthesize my passwords, I typically have mine set to 32, then reduce it as required when a brain dead site sets a lower upper limit.

The Turkish researcher also observed that password complexity was another problem. Only 12% — 1 in 8 — of the passwords contained any special character. While that's not good for them, that's great news for us. Since the bad guys who might be attempting to brute force our credentials also know this, they will certainly expend all of their time not bothering to brute force special characters. There's just no payoff in that. So when =WE= use special characters we're HUGELY reducing the likelihood that our password will be cracked through brute force hashing.

Most of the passwords (29%) used only letters or (13%) used only numbers. The full research is up on GitHub for anyone who's curious to know more...
https://github.com/FlameOfIgnis/Pwdb-Public

But we have some nice bullet-point takeways:

- From 1.000.000.000+ lines of dumps, 257.669.588 were filtered as either corrupt data (gibberish in improper format) or test accounts.
- 1 Billion credentials boil down to 168.919.919 unique passwords, and 393.386.953 unique usernames.
- As we titled this podcast, the most common password is "123456." It covers roughly 0.722% of all the passwords. (Around 7 million times per billion)
- The top 1,000 recurring passwords cover 6.607% of all the passwords.
- Within the most common 1 million passwords the hit-rate against the whole collection is 36.28%, and within most common 10 million passwords the hit rate is at 54.00%. This means that any test of those top 10 million passwords would have a better than 50/50 chance of succeeding against any one of those randomly selected 1 billion credentials.
- The average password length is 9.4822 characters.
- Only 12.04% of passwords contain special characters.
- 28.79% of passwords are letters only.
- 26.16% of passwords are lowercase only.
- 13.37% of passwords are numbers only.
- 34.41% of all passwords end with digits, but only 4.522% of all passwords start with digits.

And, finally, in an observation that constitutes a real contribution, this researcher noticed and then researched an unsuspected pattern among the data...

He wrote: "During my research, I've noticed a handful [of] apparently high entropy passwords (all 10 characters, uppercase-lowercase-digits) that were being reused. These passwords had a very low occurrence rate, but far more than one would expect." Specifically:

- They all start and end with uppercase characters
- None of them seem to have a keyboard pattern or meaningful word in them.
- They are all 10 characters long.
- They don't contain special characters.
- Some of them occurred up to 1 per 100 million credentials (meaning i have around 10 reuses

of it currently)

- Most recent occurrence for these: 86 of these were found in a 55623 credentials from a leak in June 2020

He found 763 thousand passwords matching this pattern and he observed:

“I have no idea what this uncovers and what it implies, but I'm suspecting a password manager out there is creating passwords with low entropy, causing repetitions over a lot of users. All the ideas about this are welcome and appreciated.”

Next week: “**Glupteba**” a deep dive into the operation of a state-of-the-art malware.