Security Now! #829 - 07-27-21

SeriousSAM & PetitPotam

This week on Security Now!

This week we will plow into another two new serious vulnerabilities brought to the industry by Microsoft named SeriousSAM and PetitPotam. But we first look at how Chrome managed to hugely speed up its Phishing website early warning system (making it even earlier). We cover the striking news of Kaseya having obtained a universal decryptor which is effective for every one of their victims, we look at the massive HP printer driver mess and consider the larger lesson that it teaches, and then we look at the new security features GitHub is bringing to its support of the "Go" language. Then, after sharing one bit of listener feedback, we plow into SeriousSAM and PetitPotam.

"In the show, staged above Shanghai's scenic waterfront promenade, the Bund, 1,500 illuminated drones formed into the game's logo and characters before transforming into a floating QR code that links to its homepage."
https://www.vice.com/en/article/88n9vb/shanghai-drone-show-qr-code

Browser News

Faster and more efficient phishing detection in Chrome 92

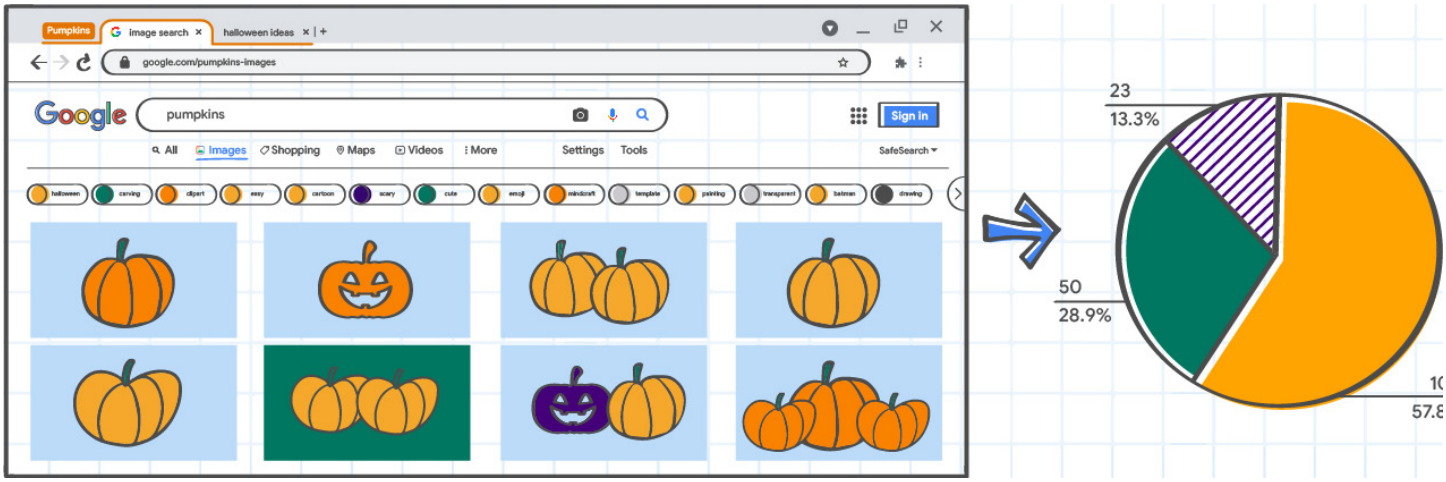
So, I'm reading this Chromium blog posting titled "Faster and more efficient phishing detection in M92" because that sounds like a good thing, and, depending upon what the posting's details revealed, I figured that it might be of interest to this podcast's listeners. You're hearing about it because that did turn out to be the case, though perhaps not for the reason you might imagine.

The posting starts out with an introductory market spiel:

> *"Keeping Chrome users safe as they browse the web is crucially important to Chrome; in fact, security has always been one of our four core principles. In some cases, security can come at the expense of performance. In this "The Fast and the Curious" series, we are excited to share how improvements to our phishing detection algorithms keeps users safe online. With these improvements, phishing detection is now 50 times faster and drains less battery."*

Then under the sub-heading "Phishing detection" they write:

> *Every time you navigate to a new page, Chrome evaluates a collection of signals about the page to see if it matches those of phishing sites. To do that, we compare the color profile of the visited page - that's the range and frequency of the colors present on the page - with the color profiles of common pages. For example in the image below, we can see that the colors are mostly orange, followed by green and then a touch of purple.*



Of course, those listening cannot see the diagram which I've included in the show notes. But it's a page with a bunch of orange pumpkins so that orange dominates the page, though they are all in light blue frames, which they're ignoring, except for the one that's green. But in any event, this says that to detect phishing they're looking at a page's color distribution. So my first thought was "What?! They're comparing the color profile of a page we visit to the color profiles of common pages? Really?" Turns out, yes, they are. That's what Chrome does.

Then it hit me: Whatever they do to pull off this detection needs to be done entirely on the client, right? Chrome cannot be sending all visited-page URL's back to the Google mothership. That would be a privacy catastrophe. We'll ignore for the time being that loading our web pages full of image beacons and JavaScript includes is essentially doing exactly that.

And then they confirm the nature of this strategy by explaining:

If the site matches a known phishing site, Chrome warns you to protect your personal information and prevent you from exposing your credentials.

To preserve your privacy, by default Chrome's Safe Browsing mode never sends any images outside the browser. While this is great for privacy, it means that your machine has to do all the work to analyze the image.

I've lightly edited the posting for clarity without their graphics:

Image processing can often generate heavy workloads because analyzing the image requires an evaluation of each pixel in what is commonly known as a "pixel loop." Some modern monitors display upwards of 14 million pixels, so even simple operations on each of those pixels can add up to a lot of CPU use! For phishing detection, the operation that takes place on each pixel is the counting of its basic colors.

The counts are stored in an associative data structure called a hashmap. For each pixel, we extract its RGB color values and store the counts in one of 3 different hashmaps -- one for each color. Adding one item to a hashmap is fast, but we have to do this for millions of pixels. We try to avoid reducing the number of pixels to avoid compromising the quality of the analysis. However, the computation itself can be improved — and in M92 it has been. The code now avoids keeping track of RGB channels in three different hashmaps and instead uses only one to index by color. Three times less counting! And consecutive pixels are summed before being counted in the hashmap. For a site with a uniform background color, this can reduce the hashmap overhead to almost nothing. With the new approach there are significantly fewer operations on the hashmap:

As a result, starting with M92, Chrome now executes image-based phishing classification up to 50 times faster at the 50th percentile and 2.5 times faster at the 99th percentile. On average, users will get their phishing classification results after 100 milliseconds, instead of 1.8 seconds.This benefits you in two ways as you use Chrome. First and foremost, using less CPU time to achieve the same work improves general performance. Less CPU time means less battery drain and less time with spinning fans.

Second, getting the results faster means Chrome can warn you earlier. The optimization brought the percentage of requests that took more than 5 seconds to process from 16.25% to less than 1.6%. This speed improvement makes a real difference in security - especially when it comes to stopping you from entering your password in a malicious site!

Overall, these changes achieve a reduction of almost 1.2% of the total CPU time used by all Chrome renderer processes and utility processes.

At Chrome's scale, even minor algorithm improvements can result in major energy efficiency gains in aggregate. Here's to many more centuries of CPU time saved! Stay tuned for many more performance improvements to come!

Ransomware News

A Universal Decryptor for all Kaseya victims

The industry recently received some interesting news from Kaseya, who recently posted a pair of interesting updates. The first one was posted last Thursday on July 22 at 3:30 PM EDT: https://helpdesk.kaseya.com/hc/en-gb/articles/4403440684689-Important-Notice-July-22nd-2021

> *Kaseya has obtained a universal decryptor key.*
>
> *On 7/21/2021, Kaseya obtained a decryptor for victims of the REvil ransomware attack, and we're working to remediate customers impacted by the incident.*
>
> *We can confirm that Kaseya obtained the tool from a third party and have teams actively helping customers affected by the ransomware to restore their environments, with no reports of any problem or issues associated with the decryptor. Kaseya is working with EMSIsoft to support our customer engagement efforts, and EMSIsoft has confirmed the key is effective at unlocking victims.*
>
> *We remain committed to ensuring the highest levels of safety for our customers and will continue to update here as more details become available.*
>
> *Customers who have been impacted by the ransomware will be contacted by Kaseya representatives.*

Then, yesterday, on Monday July 26, 2021 at 1:00 PM EDT they updated that, writing:

> *Throughout this past weekend, Kaseya's Incident Response team and EMSIsoft partners continued their work assisting our customers and others with the restoration of their encrypted data. We continue to provide the decryptor to customers that request it, and we encourage all our customers whose data may have been encrypted during the attack to reach out to your contacts at Kaseya. The decryption tool has proven 100% effective at decrypting files that were fully encrypted in the attack.*
>
> *Kaseya has maintained our focus on assisting our customers, and when Kaseya obtained the decryptor last week we moved as quickly as possible to safely use the decryptor to help our customers recover their encrypted data. Recent reports have suggested that our continued silence on whether Kaseya paid the ransom may encourage additional ransomware attacks, but nothing could be further from our goal. While each company must make its own decision on whether to pay the ransom, Kaseya decided after consultation with experts to not negotiate with the criminals who perpetrated this attack and we have not wavered from that commitment. As such, we are confirming in no uncertain terms that Kaseya did not pay a ransom – either directly or indirectly through a third party – to obtain the decryptor.*

Thanks to our podcast "REvil's Clever Crypto" two weeks ago, we know exactly how a single campaign-wide, multi-system universal decryptor could be provided. Thanks to the dual side-by-side encryption of the private half of each system-specific keypair, either the REvil affiliate themselves could have been induced to provide their private key for the entire Kaseya campaign, or the REvil gang themselves could have provided a universal decryptor which would

have been effective for all of that affiliate's victims, among which Kaseya would certainly have been the most prominent.

We'll likely never know what may or may not have transpired behind the scenes between Global political and law enforcement agencies, Russia's political and law enforcement agencies, and REvil and/or their Kaseya-attacking affiliate. But this supports the contention I voiced last week that we can expect to see future ransomware attacks and attackers working to deliberately remain under the radar. From the standpoint of REvil and their affiliate, the Colonial Pipeline, JBS Foods and Kaseya attacks have been catastrophic ... because they became far too public, and thus political. I'm sure that the lesson that's been taken away is to make smaller waves.

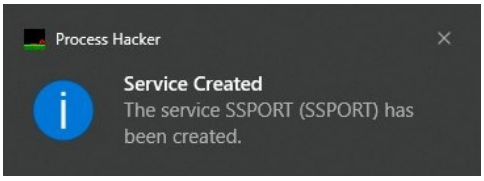
Security News

The printer driver used by millions of HP, Samsung and Xerox Printers is exploitable
A researcher by the name of Asaf Amir with Sentinel Labs gave HP, Samsung and Xerox a generous 5-month vulnerability pre-disclosure period of what he and his team discovered. So, they were notified last February, 5 months ago. And Microsoft has incorporated its update into one of recent Windows Updates. The Sentinel Labs' vulnerability disclosure was first published last Tuesday, containing four bullet points by way of its Executive Summary:

- *SentinelLabs has discovered a high severity flaw in HP, Samsung, and Xerox printer drivers.*
- *Since 2005 HP, Samsung, and Xerox have released millions of printers worldwide with the vulnerable driver.*
- *SentinelLabs' findings were proactively reported to HP on Feb 18, 2021 and are tracked as CVE-2021-3438, marked with CVSS Score 8.8.*
- *HP released a security update on May 19th to its customers to address this vulnerability.*

What Asaf and his team discovered was a trivial-to-exploit flaw affecting the printer drivers used by a large family of printers which have been continuously shipping since 2005, the year this podcast began. Here's the way Asaf describes what happened, and what they found:

Several months ago, while configuring a brand new HP printer, our team came across an old printer driver from 2005 called SSPORT.SYS thanks to an alert by Process Hacker once again.



This led to the discovery of a high severity vulnerability in HP, Xerox, and Samsung printer driver software that has remained hidden for 16 years. This vulnerability affects a very long list of ***over 380 different HP and Samsung printer models*** *as well as at least a dozen different Xerox products.*

Their disclosure page shows sample lists of just some of the printers. But, Leo, goto this page: https://support.hp.com/us-en/document/ish_3900395-3833905-16
And expand the "Affected Products" item to see what that closed expansion is hiding.

Just by running the printer software, the driver gets installed and activated on the machine regardless of whether you complete the installation or cancel. Thus, in effect, this driver gets installed and loaded without even asking or notifying the user. Whether you are configuring the printer to work wirelessly or via a USB cable, this driver gets loaded. In addition, it will be loaded by Windows on every boot. This makes the driver a perfect candidate to target since it will always be loaded on the machine even if there is no printer connected.

The vulnerable function inside the driver accepts data sent from User Mode via IOCTL (Input/Output Control) without validating the size parameter. This function copies a string from the user input using strncpy with a size parameter that is controlled by the user. Essentially, this allows attackers to overrun the buffer used by the driver.

Windows' IOCTL API is explicitly a means for allowing unprivileged User Mode code running in Ring 3 to communicate with drivers, services and other Kernel code in Ring 0. So what we have here, once again, is a trivia-to-exploit means for bypassing Windows' entire process and user security privilege model.

Last week we were introduced to the idea of signing a malicious printer driver for installation somewhere within an enterprise's network, whereupon Windows would auto-load it through their Point And Print facility, which Microsoft subsequently assured us was "vulnerable by design" (their exact words). Therefore it truly wasn't a bug, it was a feature.

But for 16 years, from 2005 until a month or two ago, for hundreds of millions of Windows systems worldwide, it was no longer necessary to bother with all of that malicious printer driver creation and signing muss and fuss. Just find any machine with the SSPORT.SYS device driver present and pass a specially crafted IOCTL API call to it, immediately taking over the system.

An interesting thing we noticed while investigating this driver is this peculiar hardcoded string: "This String is from Device Driver@@@@ ". It seems that HP didn't develop this driver, but copied it from a project in Windows Driver Samples by Microsoft that has almost identical functionality; fortunately, the MS sample project does not contain the vulnerability.

An exploitable kernel driver vulnerability can lead an unprivileged user to a SYSTEM account and run code in kernel mode (since the vulnerable driver is locally available to anyone). Among the obvious abuses of such vulnerabilities are that they could be used to bypass security products.

Successfully exploiting a driver vulnerability might allow attackers to potentially install programs, view, change, encrypt or delete data, or create new accounts with full user rights. Weaponizing this vulnerability might require chaining other bugs as we didn't find a way to weaponize it by itself given the time invested.

Generally speaking, it is highly recommended that in order to reduce the attack surface provided by device drivers with exposed IOCTL handlers, developers should enforce strong ACLs when creating kernel device objects, verify user input and not expose a generic interface to kernel mode operations.

As I mentioned above, the world has apparently dodged another bullet. We don't know of this vulnerability having been discovered and used to perform effortless elevation of privilege attacks on Windows systems during the previous 16 years. But the ability to do so has been there since the start of this podcast.

But we should take note that this highlights a fundamental and significant security weakness in the architecture of Windows which can never be remedied: We saw a strong hint of this design flaw last week when Mimikatz's Benjamin Delpy demonstrated how a malicious printer driver, once installed into a low-value machine, could be proactively pulled throughout an enterprise by Windows' PointAndPrint feature which was designed to silently and automatically install any needed drivers so that its users didn't need to be bothered.

But Windows' problem is actually much worse, as this serious problem with a widespread HP printer driver has just demonstrated. Windows drivers, like core services, run in the kernel. Userland applications can communicate with them two ways: through the normal data channel such as opening a device and sending output to it. We might call this "in band" use of the driver. But drivers can also be communicated with "out of band" through the use of the IOCTL API. A printer driver, for example, might provide IOCTL services to user mode applications for setting its page orientation, checking on toner level or communicating readiness and error conditions.

So, Windows provides for both in-band and out-of-band communication between applications and drivers. That's not unusual. But these drivers run with kernel privilege and the out-of-band IOCTL API deliberately crosses privilege boundaries. This means that **any** flaw existing in **any** driver can be used by **any** code running on the system to compromise that system. And as we have just seen, these drivers are not all written, vetted and provided by Microsoft. More often than not they're provided by third-parties to make their devices work with Windows. Therefore, **any** mistake made by **any** device driver vendor which is discovered by **anyone** malicious can potentially be used to compromise an entire system.

A long time ago, in a galaxy far far away, long before network security was even a thing, this design made sense because remote attackers operating from hostile foreign countries were not able to lurk around in our machines. But we've often used the apropos model of a chain of individual links, where the strength of the entire chain is limited by the strength of its weakest link. In Windows, the weakest links may be flaws lurking in the services and device drivers provided by well meaning developers whose code has never been thoroughly stress tested and security vetted because the moment it started to work without crashing it was declared finished, was packaged up and was shipped.

In that world, which is now unfortunately this world, it doesn't take much imagination to picture an attacker who gets into a machine as a low-privilege user, taking an inventory of the system's third-party services and device drivers, then cross-referencing that list against their internal stash of privately discovered 3rd-party service and device driver exploits.

For now, if you have an HP printer that's among the 380 models listed, and wish to check for any updates yourself just to be sure, you can use the page that I've linked to in the show notes: https://support.hp.com/us-en/drivers/printers to search for updates for your own printers.

Windows' Process Hacker

The Sentinel Labs guys discovered this whole HP printer driver mess when a tool they had running at the time, known as "Process Hacker" popped up a notification that a new "SSPORT" service had just been created as a result of something they were doing. I, for one, would love the idea of being proactively notified when something has just added a background service or driver to my system. So I wanted to take a moment to shine a light on the tool they used, known as "Process Hacker."

Many of us are familiar with Mark Russinovich's excellent SysInternals tools. "Process Hacker" is immediately reminiscent of Mark's Process Explorer, so much so that I'm sure Mark's work was the inspiration behind Process Hacker. But Process Hacker has taken this far further. It looks looks like Process Explorer on steroids. It's open source, still at sourceforge, but also with a GitHub presence. It runs on anything from Win7 on and is showing a download count of 6.5 million. It shows 34 contributors, 814 forks, and is being actively developed and maintained. It bills itself as "A free, powerful, multi-purpose tool that helps you monitor system resources, debug software and detect malware." It's bullet pointed features

- A detailed overview of system activity with highlighting.
- Graphs and statistics allow you quickly to track down resource hogs and runaway processes.
- Can't edit or delete a file? Discover which processes are using that file.
- See what programs have active network connections, and close them if necessary.
- Get real-time information on disk access.
- View detailed stack traces with kernel-mode, WOW64 and .NET support.
- Go beyond services.msc: create, edit and control services.
- Small, portable and no installation required.
- 100% Free Software (GPL v3)

And it offers a plug-in architecture for extensions. Overall, it looks like a very nice piece of work. If it's able to sit quietly in the background and alert me when something is setting up permanent residence on any of my Windows machines, that's something I'd like to have. So, if you haven't completely given up on the idea that you might still have some remaining shred of control over the machine that's sitting in front of you, Google "Process Hacker" and you'll find it.

"GoLang" gains supply chain security features at GitHub

Last Thursday's GitHub blog posting was titled: "GitHub brings supply chain security features to the Go community"

> *The global Go community embraced GitHub from the beginning—both as a place to collaborate on code and a place to publish packages—leading to Go becoming one of the top 15 programming languages on GitHub today. We're excited to announce that GitHub's supply chain security features are now available for Go modules, which will help the Go community*

discover, report, and prevent security vulnerabilities.

Commenting on GitHub's announcement, Google's Go Language (or GoLang) product lead, Steve Francia, noted:

> Go was created, in part, to address the problem of managing dependencies in large-scale software. GitHub is the most popular host for open-source Go modules. The features announced today will help not just GitHub users but anyone who depends on GitHub-hosted modules. We are thrilled that GitHub is investing in improvements that benefit the entire Go ecosystem, and we look forward to more collaborations with them in the future.

The Go Language itself is rapidly gaining ground with 76% of respondents to a 2020 Developer Survey saying that Go is now used in some form in the enterprise.

Go's module system was introduced in 2019 to make dependency management easier and version information more explicit and Go moodule adoption is also increasing, with 96% of those surveyed saying that these modules are used for package management, a 7% increase from 2019; and 87% of respondents reported that only Go modules are used for this purpose. And an overall trend in the survey appears to suggest the use of other package management tools is decreasing.

GitHub's blog post detailed the four primary areas of improvement in supply chain security that are now available for Go modules. The first is GitHub's Advisory Database, which is an open source repository of vulnerability information containing over 150 Go advisories, a number which GitHub said is growing every day as they curate existing vulnerabilities and triage newly discovered troubles. The database also allows developers to request CVE IDs for newly-discovered security issues.

GitHub has also now provided its dependency graph, which can be used to monitor and analyze project dependencies though go.mod -- as well as to alert users when vulnerable dependencies are detected.

And GitHub has included Dependabot in this update, which will proactively send developers a notification when new vulnerabilities are discovered in Go modules. I'm sure our listeners all know how great I think that is. Developers will still need to seriously heed any notifications they receive. And it's not possible to heed notifications that are not received. So being proactive is a wonderful step. Automatic pull requests can be enabled to patch vulnerable Go modules and notification settings have been upgraded for fine-tuning. GitHub said that when repositories are set to automatically generate pull requests for security updates, dependencies tend to patch up to 40% faster than those which do not.

We've talked incessantly on this podcast about the power that automated and automatic background updates have to rapidly remediate security vulnerabilities. Whenever I do this I'm reminded by some of our intrepid listeners that with automation comes its risk of subversion. That's inarguably true.

But this month alone, Microsoft patched 117 flaws, 9 of which were 0-days with 4 of those being seen actively exploited in the wild. As it is, Windows is barely holding together. It's not possible any longer to conceive of a world without Windows automatic updates, where, as it was once upon a time, every individual end user was obligated to go get and manually install Windows updates onto their systems. And those old timers among us will recall the controversy that surrounded Microsoft's decision to take that job out of our hands. Cranky old guys who were still imagining that they had any real control, objected to the idea that their beloved hand-built machines might be changed without their prior approval.

That day has long since passed... and I believe that we're headed much farther down that path. Because like it or not, warts and all, it's the right path for us to take. In time, I think it's going to become just as clear that the only way for complex multi-component and multi-sourced software to be built and maintained will be for similar dependency graph driven automated supply-chain management to become standard operating procedure.

Developers are just as busy as end users—if not more so. And, yes, it's true that automated supply-chain management brings risks of supply-chain attacks. But we're heading in with our eyes open and this is not our first rodeo. Automating the entire software lifecycle, from the developer's fingers on keyboard all the way out to code running on machines—creating the ability to write and publish incremental updates which then securely flow out as binary updates everywhere that code appears—anywhere in the world is where we need to eventually arrive.

And yes, it will create a mess. It will be a mess of overhead that we do not yet have today. As an analogy, look at what a mess the addition of "Secure Boot" has created. An incredible amount of overhead that's beginning to appear in every system, to address a problem that almost none of those systems will ever have. And what does it do? All it does (when it isn't bypassed) is attempt to assure that every step of the operating system boot process uses verifiably signed code. In the future, we're going to muck up our development process similarly. It's going to be a mess, too... but I think it's as inevitable as was allowing Windows to update itself.

Closing the Loop

JF / @jfparis

Hi @SGgrc I listened to you repeatedly trashing Qnap over several show.
Quality of their software is doubtful I agree but unlike many of their peers it is relatively easy to replace it with a clean Linux distro

SeriousSAM & PetitPotam

SeriousSAM

The first of these two new problems for Windows was just developing as last week's podcast was being produced. We noted last week that it had been preliminarily named "HiveNightmare" since we were all in a "nightmare" mode following the many recent printer nightmares. But since then, the name "SeriousSAM" appears to have taken root—SAM being the abbreviation for Windows' Security Account Manager. And I prefer this name since the trouble is entirely separate from any printer-related trouble. Unfortunately, "Serious Sam" is also the name of a 20 year old, and still relevant, first person shooter. So Googling just the phrase "Serious Sam" will return lots of gaming hits rather than security information. In any event, let's take a look at this problem first...

Last Monday, July 19, security researchers began reporting that the Security Account Manager (SAM) file on Win10 & 11 systems was READ-enabled for all local users. That was quite deliberately not true of Windows before v10, it's not good, and it's a critical security mistake.

The Security Account Manager SAM file—as its name suggests—stores sensitive security information including storing and caching all of the hashes for the user and admin passwords the system is aware of. Having this file's security access rights enabled for reading by everyone means that attackers with any access to the system can use this SAM file information to escalate privileges or access other data. It's a big no-no.

The next day, Tuesday the 20th, Microsoft issued an out-of-band advisory for this vulnerability, which is now being tracked as CVE-2021-36934. And as of last Thursday the vulnerability has been confirmed to affect Windows 10 version 1809 and later, as well as Windows Server 2019 and later.

A public proof-of-concept is available that allows non-admin users to retrieve all registry hives and researcher Kevin Beaumont (who tweets as @GossiTheDog) has released a demo that confirms that it's both possible and practical to obtain local hashes and pass them to a remote machine, achieving remote code execution as SYSTEM on arbitrary targets in addition to privilege escalation.

CERT's Coordination Center has published detailed vulnerability notes on CVE-2021-36934 titled: "Microsoft Windows gives unprivileged user access to system32\config files"

https://www.kb.cert.org/vuls/id/506989

Multiple versions of Windows grant non-administrative users read access to files in the C:\Windows\system32\config directory. This can allow for local privilege escalation (LPE).

With multiple versions of Windows, the BUILTIN\Users group is given RX permissions to files in the C:\Windows\system32\config directory.

If a VSS shadow copy of the system drive is available, a non-privileged user may leverage access to these files to achieve a number of impacts, including but not limited to:

- Extract and leverage account password hashes.
- Discover the original Windows installation password.
- Obtain DPAPI computer keys, which can be used to decrypt all computer private keys.
- Obtain a computer machine account, which can be used in a silver ticket attack.

DPAPI is Windows "Data Protection" API which has been part of Windows since Win2000. It offers Windows clients various simple and straightforward cryptographic services. Windows uses this DPAPI to protect various of its own sensitive local private keys, but, as always, the master key must be around somewhere. This flaw makes it available.

Note that VSS shadow copies may not be available in some configurations, however simply having a system drive that is larger that 128GB in size and then performing a Windows Update or using Windows setup to install an MSI package file will ensure that a VSS shadow copy will be created automatically to allow a system change roll-back. To see whether a system has VSS shadow copies available, issue this command from a privileged command prompt:

`vssadmin list shadows`

To check if a system is vulnerable, issues this command from a **non**-privileged command prompt: **`icacls %windir%\system32\config\sam`**

If a bunch of stuff is printed, ending with *"Successfully processed 1 files; Failed processing 0 files"* from a **non**-privileged command prompt, **this system IS** vulnerable. But if you are told:

C:\Windows\system32\config\sam: *Access is denied.*
Successfully processed 0 files; Failed processing 1 files

Then this system is **not** giving read access to the system's SAM—that's good!

There is currently no patch from Microsoft for the trouble, though I'll be surprised if we don't see something soon. In the meantime, Microsoft did release remediation guidance for Windows 10 and 11 users to mitigate the risk of immediate exploitation. For these measures to be effective it's necessary to **FIRST** restrict future access **AND THEN** delete any existing shadow copies. A replacement shadow copy can then be immediately recreated if needed. And note that the CERT Coordination Center (and everyone else) recommends applying the remediations on an emergency basis.

1. Restrict access to the contents of %windir%\system32\config:

   Open Command Prompt or Windows PowerShell as an administrator.
   Run this command: **`icacls %windir%\system32\config\*.* /inheritance:e`**

2. Delete Volume Shadow Copy Service (VSS) shadow copies:

   Once the ACLs have been corrected for these files, any VSS shadow copies of the system drive must be deleted to protect a system against exploitation. This can be accomplished with the following command: **`vssadmin delete shadows /for=%systemdrive% /Quiet`**

Confirm that VSS shadow copies were deleted by running **`vssadmin list shadows`** again.

3. Create a new System Restore point if desired.

Okay. So that was **SeriousSAM**. Let's now turn to **PetitPotam**...

A Paris-based French security researcher whose GitHub handle is "topotam" and who appears to have a thing for hippopotamuses (we'll get to that in a second), recently discovered and went public with another (what day is it?) serious security flaw in Windows which can be exploited to force remote Windows servers to authenticate with an attacker, thus sharing their NT LANMAN authentication details and certificates.

Given that this researcher's GitHub page shows a bunch of apparently quite happy and sort of adorable hippopotamuses. It appears that the "Potam" of "PetitPotam" is meant to put us in mind of a small hippopotamus.

https://github.com/topotam/PetitPotam

So, should the question of the naming of this Windows security fiasco du jour arise at a cocktail party, all of our listeners will be fully equipped to duly impress their friends with their erudition.

Okay, so what **is** this new problem?

The trouble surrounds a means of abusing Microsoft's MS-EFSRPC protocol. "EFS" as in Encrypted File System and "RPC" as in Remote Procedure Call. So, MS-EFSRPC is the network protocol which enables Windows machines to perform operations on encrypted file system data stored on remote encrypted NTFS-based systems. But an encrypted file system is not required for this attack to work.

The PetitPotam attack proof-of-concept code allows an attacker to send SMB requests to a remote system's MS-EFSRPC endpoint interface to cause the target victim computer to initiate an authentication procedure and thus share its authentication details. Attackers can collect this data and abuse it as part of a NTLM relay attack to gain access to remote systems on the same internal network.

PetitPotam cannot be exploited remotely across the Internet, It's an attack that's designed to be used inside large corporate networks, where attackers could use it to force domain controllers to cough up their NTLM password hashes or authentication certificates. This could then, in turn, lead to the complete takeover of a company's internal network. So we can see why Microsoft responded with surprising speed to the new threat created by this public proof of concept.

There's a related exploit using MS-RPRN (that's their Print System Remote Protocol), though the developer of the PetitPotam exploit tweeted Sunday before last back on the 18th, when it first pointed the world to his discovery:

> *Hi all,*
> *MS-RPRN to coerce machine authentication is great but the service is often disabled nowadays by admins on most orgz. Here is one another way we use to elicit machine account auth via MS-EFSRPC. Enjoy!! :)* https://github.com/topotam/PetitPotam

In response to that, four days later, another researcher replied:

> *Finally finished testing it, it's quite brutal! Network access to full AD takeover… I really underestimated the impact of NTLM relay on PKI #ESC8 ?? The combo with PetitPotam is awesome! Everything is already published to quickly exploit it.*
> *— Rémi Escourrou (@remiescourrou) July 22, 2021*

Tests carried out by multiple security researchers have shown that disabling support for MS-EFSRPC did not stop the attack from working. It has been tested against Windows Server 2016 and Windows Server 2019 systems, but security researchers believe PetitPotam impacts most Windows Server versions supported today.

Florian Roth, the Head of Research at Nextron Systems, was quoted in The Record, saying: *"The problem with this type of attack is that it will take a considerable amount of time and considerations to develop appropriate countermeasures. These are design flaws that are more difficult to fix. It's much easier to just patch a vulnerable font driver DLL or Internet Explorer library."* In other words, we're here again with a fundamental flaw in a core Windows protocol where whatever fix is found must also not break existing facilities. Again, it's not a bug, it's a feature... but it's a bad feature.

https://blog.truesec.com/2021/07/25/mitigating-ntlm-relay-attacks-on-active-directory-certificate-services-ad-cs-adv210003-kb5005413-petitpotam/

The TrucSec describes exactly how this happens:

This advisory is related to the recent Certified Pre-Owned whitepaper discussing the possible abuse of the Active Directory Certificate Services AD CS role in combination with Credentials Relay Attacks such as MS-RPRN and the more recent MS-EFSRPC aka PetitPotam.

The MS-EFSRPC protocol can be used to coerce any Windows host including Domain Controllers to authenticate to a specific destination. The designated destination then forwards the NTLM credentials to another service that is configured to accept [the Domain Controller's certification] resulting in an abuse of the services.

An attacker can target a Domain Controller to send its credentials by using the MS-EFSRPC protocol and then relaying the DC NTLM credentials to the Active Directory Certificate Services AD CS Web Enrollment pages to enroll a DC certificate. This will effectively give the attacker an authentication certificate that can [then] be used to access domain services as a DC and compromise the entire domain.

AD CS is especially interesting as it offers role services that by default accept NTLM based authentication. The Certificate Authority Web Enrollment and the Certificate Enrollment Web Service can be abused to issue certificates by performing NTLM Relay Attacks using MS-EFSRPC, MS-RPRN or other API that offer a similar behavior.

Microsoft has released ADV210003 and KB5005413 in response to the published POC.

So we'll conclude with a couple of points and observations:

First, this is of no concern for end Windows users. This is entirely a high-end enterprise worry when an organization is running with Domain Controllers and Active Directory Certificate Services. So if you don't already understand that you are vulnerable, you're not.

Second, when exploited, it provides a means for an attacker who is already present on the victim's network to fully compromise and take over the entire operation. That's obviously not good because it completely collapses all security containment. But it is, at least, constrained to be a local-only attack.

In response to all of this, Microsoft immediately blamed their old, but still widely used and enabled-by-default NTLM (NT LAN Manager) protocol:

https://support.microsoft.com/en-us/topic/kb5005413-mitigating-ntlm-relay-attacks-on-active-directory-certificate-services-ad-cs-3612b773-4043-4aa9-b23d-b87910cd3429

Microsoft is aware of PetitPotam which can potentially be used in an attack on Windows domain controllers or other Windows servers. PetitPotam is a classic NTLM Relay Attack, and such attacks have been previously documented by Microsoft along with numerous mitigation options to protect customers.

To prevent NTLM Relay Attacks on networks with NTLM enabled, domain administrators must ensure that services that permit NTLM authentication make use of protections such as Extended Protection for Authentication (EPA) or signing features such as SMB signing.

PetitPotam takes advantage of servers where the Active Directory Certificate Services (AD CS) is not configured with protections for NTLM Relay Attacks. The mitigations below outline to customers how to protect their AD CS servers from such attacks.

And then Microsoft goes on to describe what needs to be changed, disabled and blocked to thwart this attack.

Note that Microsoft calls it a "**classic** NTLM relay attack" as if everyone should be aware that NTLM, which they once believed was fabulously secure, has since been so completely broken that attacks on it are considered to be "classic" (like Coke) and that as a result of NTLM's fully acknowledged crap security, it should never be used unless there's really no other choice.

Except for this, quoting an imaginary, fully forthcoming Microsoft:

*"You just bought a brand new server for your enterprise and we've just installed a bunch of old and insecure crap, on all of your servers, just in case they might need to connect to something else that you may have lying around that's also old and insecure. To eliminate **any** confusion about why all this complex stuff might not be working, we turned **everything** on and it's fully enabled, so that your shiny new systems would just work out of the box without you needing to learn anything about them, or even wonder for a minute why you couldn't just plug everything in and have it all go! So it does! ... because we're Microsoft "Vulnerable by Design."*