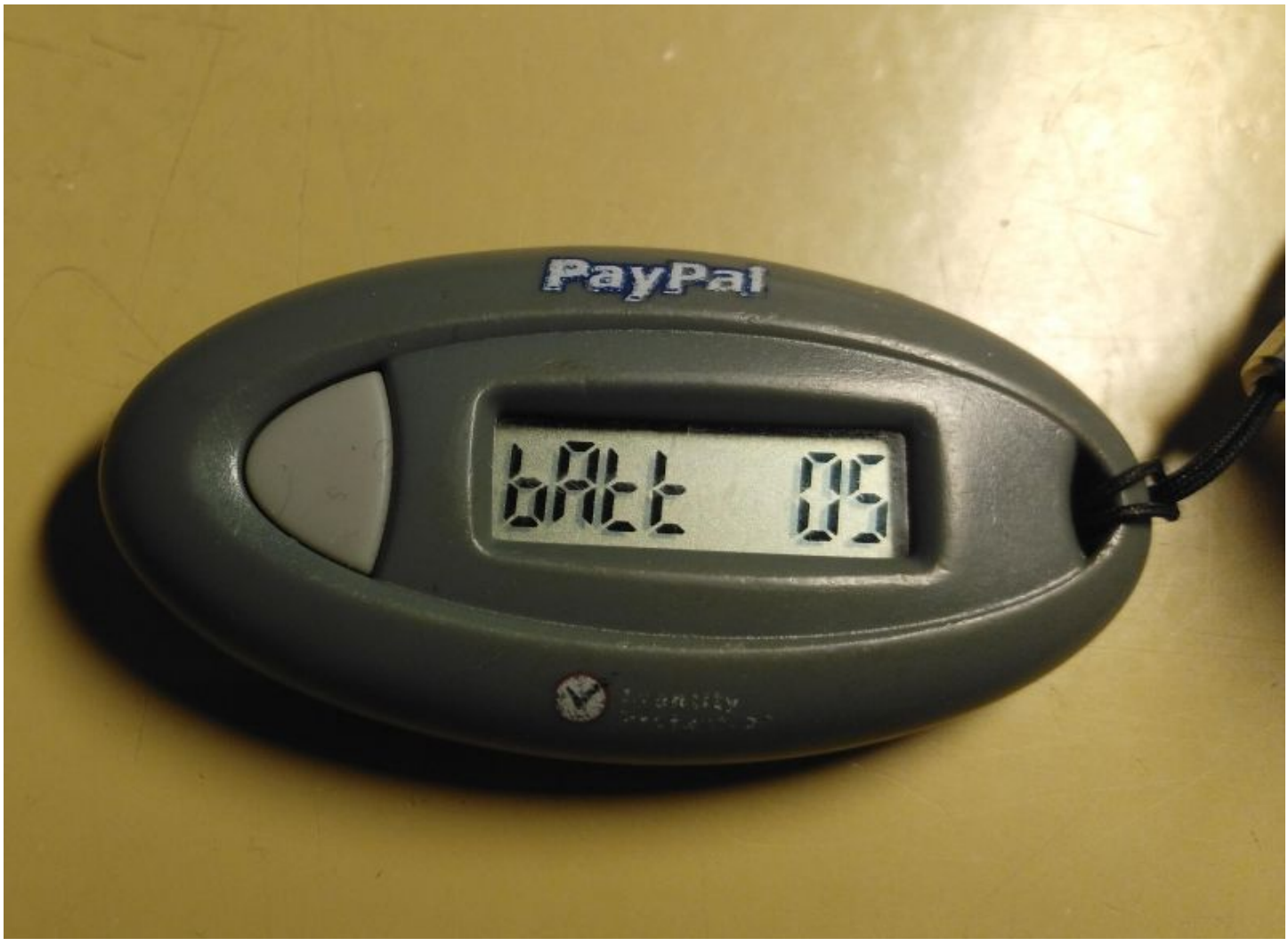
Security Now! #800 - 01-05-21

SolarBlizzard

This week on Security Now!

This week we open the New Year taking a longer look at fewer topics since the bad guys were apparently enjoying their New Year holiday, too. So we look at an interesting kludge that's been forced upon Chrome by ill-mannered antiviral scanners. We need to warn all enterprise users of Zyxel network border security products of another recently-discovered built-in backdoor. We look at the rise in IoT compromise SWATting attacks and a series of new flaws & vulnerabilities in the PHP Zend and Yii frameworks. We have a quick bit of miscellany to share, then I want to explain a lot about the value of TRIMming SSDs and newer SMR drives. And we'll conclude by catching up with what will hopefully be the last news, for a while at least, of the disastrous SolarWinds breach and intrusions.

The Time-based One Time PIN (TOTP)
has proven to be a workable solution



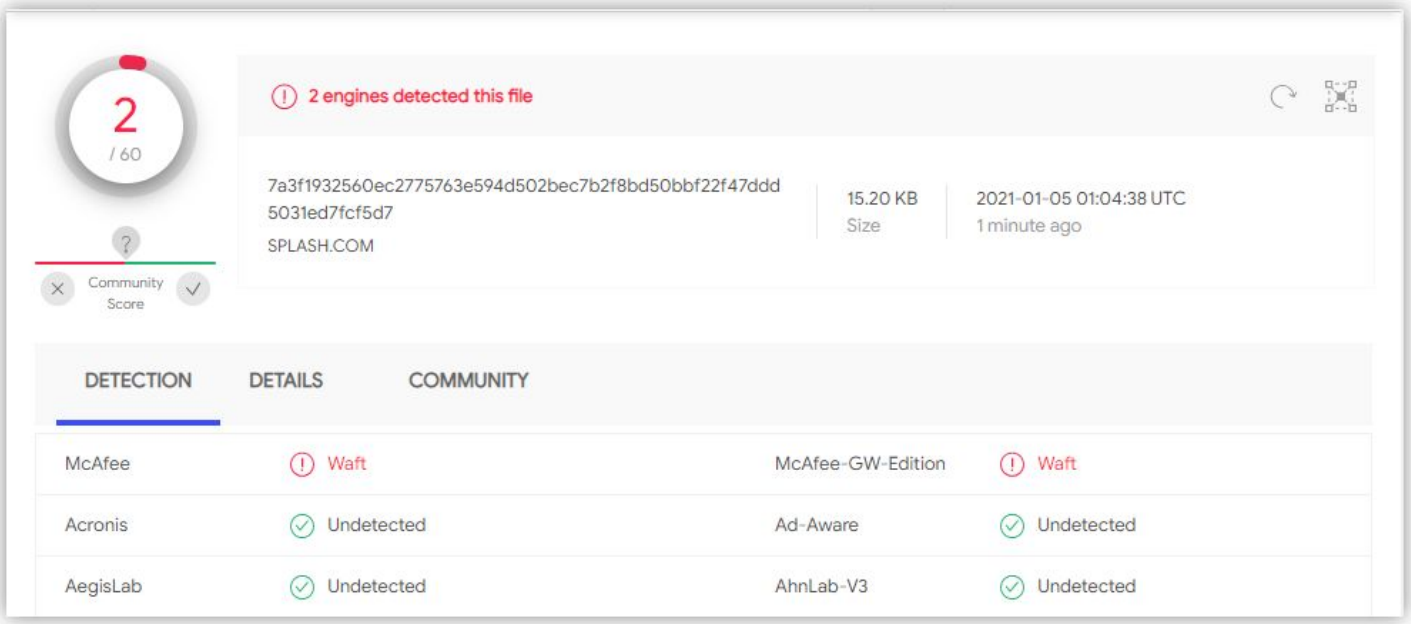
Browser News

Chrome struggles with A/V pre-scan file locking

We appear to be having more and more problems with the collision of evil forces with the forces of good which are working to thwart the evil efforts. We've seen how it's no longer sufficient for code to be signed with a certificate. Today, newly minted code signing certificates are not initially trusted, no matter the reputation of the certificate's signer or signee. After all, since fraudulent certificates are possible, the only safe presumption must be that anything new, that hasn't had time to earn itself a reputation, must be mistrusted by default, even if it means generating worrisome false-positive alarms for users. And even after going through all of the trouble of obtaining a certificate and waiting while it earns a reputation for itself, the SolarWinds Orion experience just demonstrated that even code signed with a long-trusted certificate can still be evil. So, yeah, everything is a mess.

Just yesterday I received a report in the GRC forums that McAfee A/V was declaring one of ReadSpeed's DOS files "SPLASH.COM" to be a Trojan. Splash.com is the splash screen program I wrote for ReadSpeed. It is literally a splash screen. When it's run, it puts the VGA adapter into 16 color 640x480 graphics mode, it sets the color pallet to the optimal 16 colors for the image, then decompresses the image, whose data is the rest of the program, into the VGA's memory. Since most of the screen is black, with only little bips of non-black, I wrote a simple run-length compressor for that purpose, and SPLASH.COM contains a matching run-length decompressor. So it must be that the random binary noise that's contained within the body of the program — by the purest of coincidences — matches some heuristic signature, causing McAfee to believe that this is a Trojan:

Out of curiosity, while I was assembling this podcast last night, I decided to introduce SPLASH.COM to VirusTotal to see how widespread this trouble was…



Yep. Out of 60 virus engines queried, only McAfee and McAfee-GW-Edition thought that this was an instance of the Waft Trojan. And really, if this actually was a Trojan, you'd have to feel kinda

sorry for it. Since it would be running in 16-bit DOS, with hardly any memory around and no Windows OS API to attempt to subvert, no TCP/IP stack, the Internet would be just a far off dream, and it would be totally stranded and cut off from the world. So it's really probably better that this was purely a false-positive and that the little 15.2 Kbyte .COM file program was just a splash screen.

Which brings us to the news that inspired that preamble:

In a brand new instance of the cure often causing more trouble than the thing it's attempting to prevent, we have the news that A/V programs have adopted the practice of preemptively locking newly appearing files out of an abundance — or perhaps an overabundance — of caution. They are doing this until the new file can be scanned and given the AOK for use by the system.

But it turns out that under Windows 10 this new preemptive practice has been creating some trouble for Chrome. And the cure for this invokes one of my favorite terms: kludge. Wikipedia defines kludge as: "A workaround or quick-and-dirty solution that is clumsy, inelegant, inefficient, difficult to extend, and hard to maintain. This term is used in diverse fields such as computer science, aerospace engineering, Internet slang, evolutionary neuroscience, and government."

What was Google's solution to A/V scanners preemptively locking the files it was trying to work with? Retry the operation until it works. This is so sad. It makes a mockery out of the term "Computer Science." There's no science here. This kludge was recently placed into the Chromium commit-queue, and thus into Chromium, to address bug: 1099284. That posting reads: https://chromium-review.googlesource.com/c/chromium/src/+/2559156

> Retry ReplaceFile to protect against anti-virus
>
> Anti-virus programs and other scanners may briefly lock new files which can lead to frequent problems with saving bookmarks and other files that use the ImportantFileWriter. This attempts to deal with this by retrying the racy ReplaceFile step a few times.
>
> This change also adds instrumentation to record how many retries are needed, for future tuning. It also moves the SetLastError call as close as possible to the function that will consume the last error code.
>
> This is only done on Windows because that is hoped to be the only place where it happens.

If you're catching the failure and retrying until the operation succeeds, it's not clear to me how maintaining a record of how many retries were needed, helps. I'm sure they realize that this is a horrific solution. So perhaps adding some instrumentation makes it seem more highfalutin and covers up the fact that it's a horrible kludge.

And, of course, Google is the victim here, not the culprit. From the standpoint of any program that's using the OS, the culprit is clearly the 3rd-party afterthought add-on anti-virus system

which has effectively broken and significantly destabilized the underlying operating system. What?!? Now all OS client programs, like Chrome, are supposed to add retry code to their logic in order to deal with a badly-written A/V scanner that's causing the OS to transiently return errors?

No. If the authors of any of these A/V scanners are listening, please consider this: Since any A/V scanner that is doing this would need to hook the operating system's file system API, the proper way to do this is to suspend any OS client thread that's requesting access to any file that the A/V system has not yet cleared for access. Then, once the file has been cleared, the thread that was attempting to access the blocked file would be allowed to continue, the request would succeed, and everyone would be happy. Most importantly, none of the operating system clients would need to add special-case handling code to deal with a file that's suddenly been locked out from under it by some mysterious 3rd-party. The only thing a client might notice would be that the operating system call for the file took a bit longer than normal to complete. But only real time operating systems provide guarantees of maximum call response time. And no one who has ever used Windows would confuse it with a real time operating system.

Security News

zyfwp / PrOw!aN_fXp

Get a load of this revelation: More than 100,000 Zyxel firewalls, VPN gateways, and access point controllers contain a hardcoded admin-level backdoor account that can grant attackers root access to devices via either the SSH interface or the web administration panel.

As ZDNet put it, and I agree: "The backdoor account, discovered by a team of Dutch security researchers from Eye Control, is considered as bad as it gets in terms of vulnerabilities."

Owners of these devices are advised to update systems as soon as time permits and in any event to immediately, if not sooner, disable external access to the device's SSH and WebAdmin access. Until that's done, anyone from DDoS botnet operators to state-sponsored hacking groups and ransomware gangs could abuse this backdoor account to access vulnerable devices and pivot to internal networks for additional attacks. And no one can any longer be naïve enough to imagine that won't happen.

The device models affected include many of Zyxel's top products from its line of business-grade devices which are typically deployed throughout private enterprise and government networks.

- Advanced Threat Protection (ATP) series - used primarily as a firewall
- Unified Security Gateway (USG) series - used as a hybrid firewall and VPN gateway
- USG FLEX series - used as a hybrid firewall and VPN gateway
- VPN series - used as a VPN gateway
- NXC series - used as a WLAN access point controller

Since the roles of these devices place them at the Internet-facing edge of a company's network, once they are compromised attackers are able to pivot and launch attacks against internal hosts. And, again, after what we've been through in 2020, no one should imagine that wouldn't happen to them.

At this time, patches are available for 4 of the 5 devices, the ATP, USG, USG Flex, and VPN series devices. Patches for the NXC series are expected this Friday, January 8th.

In an interview with ZDNet last week, IoT security researcher Ankit Anubhav said that Zyxel should have learned its lesson from a previous incident that took place in 2016. Our listeners may recall it since we discussed it at the time — it was tracked as CVE-2016-10401.

Back then, Zyxel devices contained a secret backdoor mechanism that allowed anyone to elevate any account on a Zyxel device to root using the super-user password: "zyad5001".

Ankit told ZDNet: "It was surprising to see yet another hardcoded credential [in Zyxel products] specially since Zyxel is well aware that the last time this happened it was abused by several botnets." Ankit noted that the "zyad5001" password is still present in the arsenal of most password attack based IoT botnets." They'll give it a go on the off chance that it works and give them root access.

But this time the situation is much worse. While the 2016 backdoor was a powerful elevation of privilege, it still required attackers to first gain access to a low-privileged account on a Zyxel device — so they can elevate it to root. But today's backdoor grants attackers direct access to the device without any preconditions. And Ankit observed that "... unlike the previous exploit, which was used for Telnet access only, today's vulnerability requires even less expertise since it's possible to directly use the credentials on the port 443 web panel." And, finally, the targets will be far more interesting to higher-end attackers. The 2016 flaw impacted home routers whereas today's mess affects enterprise-grade devices.

The Dutch researcher who found the backdoor in his own Zyxel USG40 performed a Internet-wide scan and located more than 100,000 vulnerable Zyxel devices which were answering on port 443 — the port that shared for the SSL VPN and web admin access.

"Zyxel security advisory for hardcoded credential vulnerability"
https://www.zyxel.com/support/CVE-2020-29583.shtml

The story of **why** this happened is as interesting as the fact that it did. Get a load of what Zyxel says:

> Summary
>
> Zyxel has released a patch for the hardcoded credential vulnerability of firewalls and AP controllers recently reported by researchers from EYE Netherlands. Users are advised to install the applicable firmware updates for optimal protection.
>
> What is the vulnerability?
>
> A hardcoded credential vulnerability was identified in the "zyfwp" user account in some Zyxel firewalls and AP controllers. The account was designed to deliver automatic firmware updates to connected access points through FTP.
>
> What versions are vulnerable—and what should you do?

After a thorough investigation, we've identified the vulnerable products and are releasing firmware patches to address the issue, as shown in the table below. For optimal protection, we urge users to install the applicable updates. For those not listed, they are not affected. Contact your local Zyxel support team if you require further assistance.

What's wrong with this picture? This doesn't pass the smell test. They claim to have deliberately hardcoded remote admin access credentials throughout their enterprise-grade product line for the purpose of delivering automatic firmware updates to connected access points through FTP. Perhaps they are just extremely inept, or perhaps they are not telling the truth. For one thing, the FTP port is not open. The HTTPS port is open. So, more than anything else, this looks very much like on demand backdoor access to more than 100,000 enterprises. And should Western users worry that Zyxel is a Chinese network equipment manufacturer located in Taiwan? I don't know.

If you want to know how to do automatic updates just look around. Update clients don't hold listening ports open waiting for updates to be sent in. No. Update clients periodically phone home to check-in with the mothership to see whether anything has happened recently. That way, no listening ports are ever externally exposed. And the only thing a client needs to do upon receiving an update is to use a built-in public key to decrypt a hash of the file to verify its authenticity before copying it into its own firmware. Only the authorized publisher of the product's firmware would have the matching private key that's used to encrypt the updated firmware's hash and include it for verification.

But here we are again. Zyxel says in their vulnerability disclosure: "For optimal protection, we urge users to install the applicable updates." That doesn't sound like some sort of advanced "don't worry we've got this" advanced FTP push-update delivery system. This sounds like it's up to every one of those more than 100,000 enterprises to take the initiative — over the New Years holiday — to be informed about this critical vulnerability and take the initiative to manually update the Zyxel devices on their borders. How many will? And how many more enterprises will be compromised?

I used to like Zyxel. Their hardware seemed solid and well constructed. But not, apparently, their firmware. Given that Zyxel failed to learn from their previous fiasco in 2016 and that they have done something even worse this time, it would be difficult to recommend the use of their products. Ever.

"Swatting" goes IoT

The practice of SWATting (SWAT as in "Special Weapons And Tactics") involves calling police to report a nonexistent emergency which, if it were real, would necessitate a forced entry weapons drawn heightened alert response from local law enforcement. Of course, local law enforcement has no way of knowing whether any given call for help is real or a dangerous and expensive prank. So they might be required to assume it's real as has certainly been the case in the past.

Now, the US Federal Bureau of Investigation is bringing awareness to an emerging trend where swatting victim's smart home security systems are being used to first launch and then observe swatting events.

Last week the FBI posted a Public Service Announcement titled: "Recent Swatting Attacks Targeting Residents With Camera and Voice-Capable Smart Devices"

https://www.ic3.gov/Media/Y2020/PSA201229

The FBI wrote:

The Federal Bureau of Investigation (FBI) is issuing this announcement to warn users of smart home devices with cameras and voice capabilities to use complex, unique passwords and enable two-factor authentication to help protect against "swatting" attacks. Smart home device manufacturers recently notified law enforcement that offenders have been using stolen e-mail passwords to access smart devices with cameras and voice capabilities and carry out swatting attacks.

What is Swatting?

Swatting is a term used to describe a hoax call made to emergency services, typically reporting an immediate threat to human life, to draw a response from law enforcement and the S.W.A.T. team to a specific location. Confusion on the part of homeowners or responding officers has resulted in health-related or violent consequences and pulls limited resources away from valid emergencies.

Swatting may be motivated by revenge, used as a form of harassment, or used as a prank, but it is a serious crime that may have potentially deadly consequences.

Offenders often use spoofing technology to anonymize their own phone numbers to make it appear to first responders as if the emergency call is coming from the victim's phone number. This enhances their credibility when communicating with dispatchers.

How is this version of Swatting carried out?

Recently, offenders have been using victims' smart devices, including video and audio capable home surveillance devices, to carry out swatting attacks. To gain access to the smart devices, offenders are likely taking advantage of customers who re-use their email passwords for their smart device. The offenders use stolen email passwords to log into the smart device and hijack features, including the live-stream camera and device speakers.

They then call emergency services to report a crime at the victims' residence. As law enforcement responds to the residence, the offender watches the live stream footage and engages with the responding police through the camera and speakers. In some cases, the offender also live streams the incident on shared online community platforms.

The FBI is working with private sector partners who manufacture smart devices to advise customers about the scheme and how to avoid being victimized. The FBI is also working to alert law enforcement first responders to this threat so they may respond accordingly.

Protection and Defense

Users of smart home devices with cameras and/or voice capabilities are advised of the following guidance to maximize security.

- Because offenders are using stolen email passwords to access smart devices, users should practice good cyber hygiene by ensuring they have strong, complex passwords or passphrases for their online accounts, and should not duplicate the use of passwords between different online accounts. Users should update their passwords on a regular basis.

- Users should enable two-factor authentication for their online accounts and on all devices accessible through an internet connection in order to reduce the chance a criminal could access their devices.

- It is highly recommended that the user's second factor for two-factor or multi-factor authentication be a mobile device number and not a secondary e-mail account.

So, in other words, be diligent about all of the good security practices we all know we should be employing. What we have here is another new way in which a failure to secure our perimeter might be used against us. And for what it's worth, this style of live streaming of intrusion isn't new. Around Christmas time a year ago, the publication Vice reported on a podcast called "NulledCast" which live streamed to content sharing platform Discord an incident where criminal actors hijacked a Nest and Ring smart home video and audio to harass the residents in creepy ways.

I don't mean to creep everyone out, but this is what was shown — and I was thinking about Leo's mentioning that Lisa didn't want cameras spread around the house: One incident captured a man talking to young children through the device in their bedroom, claiming to be Santa.

Vice reported last year: "In a video obtained by WMC5, Action News in Memphis Tennessee, courtesy of the family, you can see what the hacker would have seen: A viewpoint that looms over the entire room from where the camera is installed in a far corner, looking down on the children's beds and dressers while they play. The hacker is heard playing the song 'Tiptoe Through the Tulips' through the device's speakers, and when one of the daughters, who is eight years old, stops and asks who's there, the hacker says, 'It's Santa. It's your best friend.'"

Vice also reported finding posts on hacker forums offering simple Ring credential stuffing software for as little as $6. To their credit, Ring is responding. By Feb. 2020, Ring had rolled out an added layer of security beyond its already mandatory two-factor authentication, the familiar one-time six-digit code to log on. They also have alerts when someone logs onto the account and tools to control access by third-party service providers which could also be breached.

And Ring is also preparing to roll out end-to-end video encryption, which was originally slated for release by the end of 2020. Ring's announcement last September 24th, said: "With End-to-End Encryption, your videos will be encrypted on the Ring camera, and you will be the only one with the special key (stored only on your mobile device) that can decrypt and view your recordings."

The more our digital technology becomes intertwined with our analog lives, the more inherent exposure our analog lives are going to have to flaws in the digital domain.

A new serious problem in the PHP Zend Framework
Many of our podcasts through the second half of 2020 contained mentions of potentially serious WordPress vulnerabilities. WordPress is written in the PHP language and, somewhat incredibly, where a server's implementation language can be determined, 79.1% of those servers are hosting sites implemented in PHP.

Well known sites written in PHP include Facebook, 360.cn, Wikipedia, Zoom, Microsoft, Vk.com and of course Wordpress. And two sites of note which recently switched to PHP are The Washingtonpost and Tripadvisor. So PHP is clearly the current web site implementation language of choice. And PHP's usage is not declining. Its usage on the web has remained absolutely flat over the past 18 months. Active Server Pages, Ruby, Java and Scala are the also-rans in the group which together make up the remaining 20% of web site implementations.

It would be possible to simply write a site from scratch in PHP, as many people have. But over time, web authors noticed that they were writing and rewriting the same things over and over — just because there are so many things that different websites have in common. What this suggested was that PHP was still a little too "low level." So frameworks were born to function as a sort of meta-language function and class library implemented in, and running on top of, PHP. These so-called PHP Frameworks provide classes which abstract many of the common ways all sites work and the things all sites need to do. This allows site creators to focus more on the site and less on the mechanics.

So, against this background, we have newly discovered trouble in one of PHP's more venerable frameworks, Zend. Zend is currently in use by more than 570 million installations and it's the most used framework in enterprise.

We've talked about deserialization bugs in the context of JAVA, where a complex JAVA data structure needs to be stored or communicated. It's a bit like compression and decompression and the decompression or deserializing phase is inherently an interpreter. Unfortunately, those who write interpreters often implicitly assume that their serializer, which may always produce a sane and trustworthy serialization, is the only source of the data they are now being asked to deserialize. And that assumption often leads to security flaws. When attackers are able to provide the object to be deserializing, the real trouble begins.

So here we are again. Yesterday, an untrusted deserialization vulnerability was disclosed in the Zend Framework which currently has 570 million installations. This flaw can be exploited by attackers to achieve remote code execution on PHP sites — potentially 570 million of them. The vulnerability is tracked as CVE-2021-3007 and from postings on Github this also appears to impact Zend's successor project known as Laminas due to the fact that a significant amount of Zend's code was simply moved into the new project's codebase. The vulnerability exists in the most recent, current, Zend Framework 3.0.0. No exploits have been released so far, but stay tuned.

And in another PHP-related posting yesterday, another deserialization vulnerability was recently discovered and disclosed by the German RedTeam PenTesting group. This one was found in another PHP Framework, Yii:

https://blog.redteam-pentesting.de/2021/deserialization-gadget-chain/

In their posting, the RedTeam guys note that insecure deserialization was #8 on OWASP's top 10 list of web application security risks, and Check Point mentions presentations about PHP deserialization flaw exploitation dating back to 2010.

https://owasp.org/www-project-top-ten/

So the generic problem has been long recognized and is obviously still present today. Fixing these problems is easy. It's finding them that's quite difficult! And it's much better to have penetration testers finding them than highly motivated bad guys. Given PHP's crazy popularity, and the increasing intrusion pressure we seem to be witnessing, I won't be surprised if we're talking more about deserialization flaws in PHP in the future.

Miscellany

And now, for a little extra pain...
Bitcoins are currently trading at: $32,883 <sigh>

"Kim Zetter" is female
And it was also brought to my attention that I incorrectly implied that Yahoo! New's "Kim Zetter" was male. In last week's show notes I used Kim's name, but on the podcast I referred to Kim as "he" when, had I known better I should have said "she". So, just for the record. :)

ReadSpeed & SpinRite

We are continuing to see very interesting results from our podcast listeners who are experimenting with ReadSpeed and their SSDs. Here's a GRC forum post from last Wednesday by dale (https://forums.grc.com/threads/readspeed-before-after-spinrite-ran.346/) who wrote:

Hi, This WD 500GB Blue SSD was purchased 18 months ago and installed in a 9 year old computer that served mainly, until recently as a PLEX server, which probably accounts for the low speed in region 0:

```
Driv Size  Drive Identity     Location:    0      25%     50%     75%     100
---- ----- -------------------------- ------- ------- ------- ------- -------
 81  500GB WDC  WDS500G2B0A-00SM50        17.9   349.3   544.7   544.7   541.4

                Benchmarked: Monday, 2020-12-28 at 16:19
```

After running SpinRite (at Level3):

```
Driv Size  Drive Identity     Location:    0      25%     50%     75%     100
---- ----- -------------------------- ------- ------- ------- ------- -------
 81  500GB WDC  WDS500G2B0A-00SM50       326.5   532.7   531.6   531.3   534.5

                Benchmarked: Wednesday, 2020-12-30 at 13:06
```

And Dale concluded by writing "Nice!! Way to go Steve!!"

I very much appreciate Dale's enthusiasm for the fact that the read performance of the first gigabyte of his 18 month old half a terabyte Western Digital SSD has jumped up from 17.9 megabytes per second to 326.5 megabytes per second. That's obviously a huge improvement in his device's read performance. And it's very likely that much more than the first gigabyte of the drive was similarly improved. Given how slow the front of his drive was, it had to be that the drive was struggling significantly to read back its data, so this huge recovery of performance must also signify an improvement in the drive's future reliability.

But there is a tradeoff that anyone doing this needs to be aware of and I wanted to talk about this so that everyone doesn't start running SpinRite Level 3 on their SSDs. This tradeoff is apparent in Dale's 50% and 75% benchmark location numbers. His SSD's read performance in those regions **appears** to have dropped from 544.7 megabytes/second to 531 megabytes/second. And I say "appears" because that's not actually what happened. Those regions of Dale's SSD were not slowed down by passing SpinRite over them. Instead, Dale's SSD now believes that those regions are **in service** and containing valid data rather than known to be blank. And THAT is an important change.

SSDs and recent SMR shingled format spinning drives maintain a logical-to-physical mapping layer. When they are powered up this mapping is loaded into their onboard RAM for high-speed access. "Logical sectors" are what the user sees at the SSD's interface and "physical regions" are where that data is actually stored. The creation of this separation — this abstraction layer — is the way SSD's perform wear leveling by spreading repetitive writes to the same logical regions across different physical regions.

Now, one might imagine that at the start, a fresh and empty SSD would have a 1-to-1 mapping layer with logical sectors uniformly mapped out across the physical medium. Everything's just ready to go. But that's not the case. A brand new unused SSD has an empty mapping table because if nothing has been written to the SSD, there's nothing to read back. So any read from an unused SSD doesn't even bother looking at the SSD's physical storage media. That mapping layer already knows it's empty. So in response to a read, the SSD instantly returns sectors of zeroes — and thus it's read performance appears to be amazing. Saturating its interface's capacity.

So that's what Dale saw at his SSD's 50% and 75% points. Those logical regions had never been written to so they were not actually read from. Zeros were just blasted back to the benchmark at full SATA III link speed, which is 544.7 megabytes per second.

But once a SpinRite Level 3 "refresh" pass was made over the entire drive, rather than being partially empty, now its mapping table is full, with every logical sector assigned to an actual physical region of the SSD's storage medium. So, after SpinRite's pass over the SSD, the ReadSpeed benchmark revealed the true read performance of the drive's media. And it was quite respectable at 531.6 megabytes per second. Not very much slower than a SATA III link's maximum speed. And we did observe that the frighteningly slow front region of the drive is now running far faster and probably far more reliably in the future than it was before.

But there's also a new problem which SpinRite in the future will be fully aware of, and will handle automatically, but which SpinRite of today does not: The easiest way to express what a current SpinRite Level 3 pass has done, is that SpinRite will have fully "untrimmed" the SSD by writing across its entire logical surface, thereby leading it to believe that the SSD is entirely in use.

Leaving an SSD in an "untrimmed state" doesn't damage it in any way, but it can impede the performance of future writes. In order to write to an SSD, the electrostatic storage capacitors of its cells must first all be discharged and drained of their electrons. And only then can they be charged up to the varying voltage levels required to represent the data bits stored in each cell.

The hitch is that the erasing can only occur in relatively large page blocks which contain multiple logical sectors. So if the SSD knows that one of these page blocks already has all of its cells discharged and emptied, it's able to bypass the discharge phase and much more quickly write into that empty region. But if the SSD believes that all of the logical sectors within a region may be containing valid data — because they had been previously written, if only by SpinRite performing a Level 3 pass — then in order to write one sector into that page, the entire page must first be read, then the entire page must be discharged and erased, then the entire page must be rewritten with only that one sector of its data changed.

And even when a lot of data is being written, such that all of the sectors filling the entire page are being rewritten, knowing that a page is already empty and discharged allows the SSD to skip the discharge phase and to simply charge up the page's cells with their new data.

The process of informing an SSD that its logical sectors do NOT contain any useful data is known as "Trimming" and all modern operating systems are now trim-aware. And we've talked about this in the past. In fact we talked about this not so long ago when it was found that Windows 10 was attempting to TRIM its hard drives. It turns out that in that case this was being done in error and was generating kernel exception logs. But it's not as nuts as it might seem, since the latest SMR — Shingled Magnetic Recording — drives share many of the same characteristics as SSDs and do support and can benefit from the TRIM command.

A trim-aware operating system is continually updating the SSD (or SMR drive) with information when specific sector ranges no longer contain active file system data. The most common example would be whenever a file is permanently deleted from the file system rather than being moved into a trash container. When the SSD receives this TRIM information, it will likely unlink those storage pages from its sector mapping table and may place them onto a "garbage collection" list which causes them to be scheduled for erasure in the background so that when more storage is needed, already emptied pages will be ready to receive writes.

As I've mentioned before, SpinRite's awareness of its drive's contents will be raised from the drive's surface where it is today, to include popular file systems. The various flavors of FAT and NTFS will be added first, followed by the various file systems supported by Linux and other important OSes. This awareness will allow SpinRite to run far more quickly on sparsely populated drives and to trim any empty regions that it might have the occasion to write to.

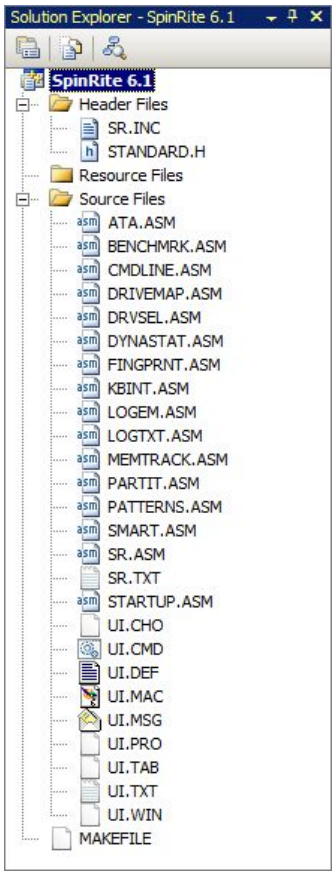
Today's operating systems often have the ability to TRIM on demand. In Windows 10 the powershell command is: "**`Optimize-Volume -DriveLetter C: -ReTrim -Verbose`**"

And the various Linuxes also have trim facilities, often invoked periodically through CRON.

So, until we have a SpinRite that's aware of the file systems on our drives — and I am doing nothing other than working toward that goal now — anyone who runs ReadSpeed and discovers slow performance at the front of their drive — which postings into the forum are revealing to be surprisingly common — and who then performs a Level3 SpinRite scan to fix those regions, could stop SpinRite before it gets out into the nether regions of the drive. Or let SpinRite refresh the entire drive, which might result in having the drive take defective regions it discovers out of service, then, after booting back into your operating system, ask your OS to manually perform a full TRIMing of the drive which will release and unmap all of the drive's unused space.

The upshot of all this is that you will not only have restored the drive's full read performance, but also preserved its full write performance. And eventually SpinRite will do all of this for us automatically.

And speaking of SpinRite…



Sunday evening I posted to my progress tracking blog over in the forums an entry titled: **"I'm at work on the SpinRite v6.1 code!"**

I wrote...

> 10 days after ReadSpeed's release, the shakedown test of the benchmark's new hardware drivers has been a tremendous success. We have identified only one problem with a system in Denmark using an old Asrock motherboard. I've located and purchased one of them in Germany. So when it arrives, I'll see what needs to be done to get the drivers running on it, too. But aside from that one problem, as far as we know, this new technology in ReadSpeed is ready for SpinRite's use.
>
> So, today, Sunday January 3rd, 2021, I successfully assembled SpinRite's 31 assembly language source code files and created a working DOS executable. Now I begin the work of incorporating the new drivers into SpinRite.

And the show notes show an image I snapped of Visual Studio's Solution Explorer panel showing SpinRite 6's file set and the newly project "SpinRite 6.1"

SolarBlizzard

A new flaw discovered in SolarWinds' Orion software
As we know, the way the bad guys were able to get their malicious "Sunburst" malware deployed into victim networks was by poisoning SolarWinds' source code repository so that all new builds of their executables would automatically and surreptitiously include the attacker's Trojan.

Now we're learning the details of an authentication bypass vulnerability which also exists in the Orion software which may have been leveraged by adversaries as a 0-day to deploy the other SuperNova malware into specific targeted environments.

According to an advisory published by CERT, the SolarWinds Orion API that's used to interface with all other Orion system monitoring and management products suffers from a security flaw (CVE-2020-10148) that could allow a remote attacker to execute unauthenticated API commands, thus resulting in a compromise of the SolarWinds instance.

The advisory states: "The authentication of the API can be bypassed by including specific parameters in the Request.PathInfo portion of a URI request to the API, which could allow an attacker to execute unauthenticated API commands."

"In particular, if an attacker appends a PathInfo parameter of 'WebResource.adx,' 'ScriptResource.adx,' 'i18n.ashx,' or 'Skipi18n' to a request to a SolarWinds Orion server, SolarWinds will set the SkipAuthorization flag, which allows the API request to be processed without requiring authentication."

Remember that SolarWinds' updated security advisory of December 24 made note of an unspecified vulnerability in the Orion Platform that could be exploited to deploy rogue software such as SuperNova. But details of the flaw had remained unknown until now.

https://msrc-blog.microsoft.com/2020/12/31/microsoft-internal-solorigate-investigation-update/

Then, Last Thursday's "Microsoft Internal Solorigate Investigation Update" provided some interesting news, including that unauthorized access had been gained to some of Microsoft's source code. But the posting also shares some interesting philosophy. They wrote:

> As we previously reported, we detected malicious SolarWinds applications in our environment, which we isolated and removed. Having investigated further, we can now report that we have not found evidence of the common TTPs (tools, techniques and procedures) related to the abuse of forged SAML tokens against our corporate domains.
>
> Our investigation has, however, revealed attempted activities beyond just the presence of malicious SolarWinds code in our environment. This activity has not put at risk the security of our services or any customer data, but we want to be transparent and share what we're learning as we combat what we believe is a very sophisticated nation-state actor.

We detected unusual activity with a small number of internal accounts and upon review, we discovered one account had been used to view source code in a number of source code repositories. The account did not have permissions to modify any code or engineering systems and our investigation further confirmed no changes were made. These accounts were investigated and remediated.

At Microsoft, we have an "inner source" approach – the use of open source software development best practices and an open source-like culture – to make source code viewable within Microsoft. This means we do not rely on the secrecy of source code for the security of products, and our threat models assume that attackers have knowledge of source code. So viewing source code is not tied to an elevation of risk.

I thought that was cool and interesting, so I wanted to share that. It would be interesting to know whether this is the way Microsoft has always operated, or whether this philosophy represents some new enlightenment.