Security Now! #1039 - 08-19-25

The Sad Case of ScriptCase

This week on Security Now!

• What AI website summaries mean for Internet economics. • Time to urgently update Plex Servers (again). • Allianz Life stolen data gets leaked. • Chrome test Incognito-mode fingerprint script blocking. • Chrome 140 additions coming in 2 weeks. • Data brokers hide opt-out pages from search engines. • Secure messaging changes in Russia. • NIST rolls-out lightweight IoT crypto. • SyncThing moves to v2.0 and beyond. • Alien:Earth -- first take. • What can we learn from another critical vulnerability?

Although it can prove awkward, escorting a terminated IT worker, as they collect their belongings and leave the building, is strongly advised.

Security News

Cloudflare vs Perplexity

What I learned in following up a bit further on the whole Cloudflare vs Perplexity question, is that the Internet is facing a profound change that's being driven by the presence of AI web-summary generators. When I went poking around to better educate myself about this issue, I discovered that a lot of the portion of the Internet that thinks about such things had blown up over this. And by "this" I don't mean the mechanics of bots and user agents, but over the fundamental change that users are driving in the way information is obtained from the Internet.

I found a terrific posting on a site called *"Contrary Research."* Last Friday they posted a piece titled *"Debating The Open Internet: Cloudflare vs. Perplexity."* <*The article*> examined and explained both viewpoints of the debate, and toward the end, said:

> *Regardless of what people may think the internet **should** do, it seems clear what it **will** do, which is to march to the beat of consumer preferences — just ask Betamax, LaserDiscs, and the Concorde.* [If this podcast's younger listeners are unaware of Betamax and LaserDiscs, that's the point.] *What the consumer wants, the consumer tends to get, consequences be damned. And today, the consumer is compelled by agentic Internet consumption. Many people believe the future of the Internet is zero-click. Those seeking to bring **that** future to life see Cloudflare's concern as the worries of a bygone era.*

I think **that's** the key more than anything else, and it signals a profound change in the economics of the Internet. While I've been working out for myself, in plain sight on this podcast, what AI means, I've observed that my own use of chatbot AI has evolved into using it as a sort of super Internet search engine. Whereas I would have once spent 15 minutes poking around the Internet looking for an answer, starting from a page of Internet search engine result links, today I often start and finish my search simply by asking ChatGPT. It's often all I need. I'll get a satisfying answer almost immediately and that will often be the end of my quest.

The reason this represents a massive change in the economics of the Internet is that the Internet is still by and large advertisement driven and in the old days – meaning before last year – during those 15 minutes of poking around, which no longer happens for me, I would have been exposed to many advertisements which would have served to finance the sites I was visiting. That's the traditional economic model that AI summarizing has flipped on its head – and killed.

TechCrunch's August 6th headline was: *"Google denies AI search features are killing website traffic"*. Whether or not and to whatever degree that might be true, the fact that it's a headline is the message.

In mid-April, Forbes wrote: *"Roughly 60% of searches now yield no clicks at all, as AI-generated answers satisfy them directly on the search results page. In addition, Google's AI Overviews have displaced top-ranked links by as much as 1,500 pixels – which is about two full screen scrolls on a desktop and three full screen scrolls on a mobile device – significantly lowering click-through rates even for highly ranked pages. Recent research has shown that AI Overviews can cause a whopping 15-64% decline in organic [website] traffic, based on industry and search type. This radical change is causing marketers to reconsider their whole strategy regarding digital visibility."*

Four months ago, over in the SEO (search engine optimization) Reddit, a poster wrote: *"In the recent months, one of our top performing websites' visits decreased by 66%. And after some investigation, we noticed everything is going well. We still have the same positions and the same*

click through rates. However, the only issue we see is that websites are not getting searches, it dropped by like more than 50%. When we search for it, we still see it on top like normal. Are people not using google search as often and relying more on AI? Are we missing something? Please advise and let us know if you are experiencing something similar."

And that posting began a thread that was followed-up on by many people saying variations of *"ChatGPT, Perplexity and Google's AI Overviews."* One person wrote:

> *I recently performed a study on SERPs (Search Engine Results Pages). It's quite obvious that three things are happening:*
>
> - *Zero click is a very real thing, every person in the study expanded the AI Overview, and nobody opened the citation links.*
> - *Some people scrolled to see up to the top 5 links, and fewer opened them.*
> - *Most people trust Google entirely and don't fact check the AIO's. Those who don't trust Google were showing signs that they eventually will. (One tester said out loud "hmm I don't feel I trust these entirely", but out of all the queries they performed they only briefly read the first result once).*
>
> *And yes, Google has lost some market share to ChatGPT, Perplexity, Claude, and others.*

Paraphrasing what the Contrary Research site said: Consumers usually wind up dictating what wins and what loses. It's quickly become clear that consumers simply want quick answers to their questions. They want them quickly and without a lot of added muss and fuss.

Given that so much of the web has been financed by search engines driving traffic to websites, which in turn generate revenue for themselves by presenting visitors with advertisements, Large Language Model ChatBots appear to be driving a generational change in the way the Internet finances itself. The political strategist James Carville is credited with coining the phrase *"It's the economy, stupid."* – meaning: *"Nothing else matters."* It's going to be interesting to see what shape the next generation Internet economy takes.

Update PLEX Media Server

Last Thursday, Plex notified some of its users to urgently update their media servers due to a recently patched security vulnerability. Though the unknown vulnerability doesn't yet have an assigned CVE-ID, Plex indicated that it impacts Plex Media Server versions 1.41.7.x to 1.42.0.x.

Plex said:

> *We recently received a report via our bug bounty program that there was a potential security issue affecting Plex Media Server versions 1.41.7.x to 1.42.0.x. Thanks to that user, we were able to address the issue, release an updated version of the server, and continue to improve our security and defenses.*
>
> *You're receiving this notice because our information indicates that a Plex Media Server owned by your Plex account is running an older version of the server. We strongly recommend that everyone update their Plex Media Server to the most recent version as soon as possible, if you have not already done so.*

Plex Media Server version 1.42.1.10060 has this vulnerability patched and can be downloaded from the server management page or the official downloads page. And props to Plex for being so proactive. That's very nice to see!

As our long-time listeners will all recall, Plex has experienced its share of critical and high-severity security flaws over the years. In March of 2023, CISA tagged a then three-year-old remote code execution (RCE) flaw (CVE-2020-5741) in the Plex Media Server as being actively exploited in attacks. And Plex had explained two years earlier at the time it released patches, successful exploitation can allow attackers to cause the Plex server to execute malicious code.

Our listeners will likely recall that it was a long-neglected Plex server running at a LastPass developer's home that was eventually found to be the cause of the devastating LastPass security breach that led many of us to decide it was finally time for us to change our password manager allegiances. The engineer had never updated their Plex server. This allowed the bad guys to surreptitiously install a keystroke logger onto the developer's PC, to obtain his LastPass authentication credentials, then compromise LastPass' network, corporate vault and backups.

So, Plex is being much more proactive today, which is great to see... and anyone who may still be using a Plex server would be "well served" to make sure you're running the latest release.

Allianz Life data gets leaked

Three weeks ago we noted that Allianz Life's network and servers had been breached and that they had lost control of their customer's data. So last week we learned that hackers had released that stolen data, exposing 2.8 million records worth of sensitive information on business partners and customers in ongoing Salesforce data theft attacks.

What we learned last month was that Allianz Life had suffered a data breach when the personal information for the "majority" of its 1.4 million customers was stolen from a third-party, cloud-based CRM system on July 16th. Although Allianz Life did not name the CRM partner, it was reportedly part of a wave of Salesforce-targeted thefts carried out by the ShinyHunters extortion group.

BleepingComputer reported:

> *Over the weekend, ShinyHunters and other threat actors claiming overlap with "Scattered Spider" and "Lapsus$" created a Telegram channel called "ScatteredLapsuSp1d3rHunters" to taunt cybersecurity researchers, law enforcement, and journalists while taking credit for a string of high-profile breaches. Many of these attacks had not previously been attributed to any threat actor, including the attacks on Internet Archive, Pearson, and Coinbase.*
>
> *One of the attacks claimed by the threat actors is Allianz Life, for which they proceeded to leak the complete databases that were stolen from the company's Salesforce instances. These files consist of the Salesforce "Accounts" and "Contacts" database tables, containing approximately 2.8 million data records for individual customers and business partners, such as wealth management companies, brokers, and financial advisors. The leaked Salesforce data includes sensitive personal information, such as names, addresses, phone numbers, dates of birth, and Tax Identification Numbers, as well as professional details like licenses, firm affiliations, product approvals, and marketing classifications.*
>
> *BleepingComputer has been able to confirm with multiple people that their data in the leaked files is accurate, including their phone numbers, email addresses, tax IDs, and other*

information contained in the database. BleepingComputer contacted Allianz Life about the leaked database but was told that they could not comment as the investigation is ongoing.

The Salesforce data theft attacks are believed to have started at the beginning of the year, with the threat actors conducting social engineering attacks to trick employees into linking a malicious OAuth app with their company's Salesforce instances. Once linked, the threat actors used the connection to download and steal the databases, which were then used to extort the company through email.

That's really pretty diabolical when you think about it. Bad guys convince an employee with sufficient access privileges that they're the company's IT department, or maybe an outside agency that's been tasked by the company with increasing the company's security profile. So the unwitting employee is instructed to download an OAuth application they'll use to authenticate. It would never occur to the employee that the OAuth app itself might be malicious, and that its use will be creating a backdoor for the bad guys.

Once again, we see that despite all of our fancy technology, it all depends upon people doing the right thing. And, unfortunately, it's the nature of security that every single person must never even once do the wrong thing, since all that's needed is a single slip-up. It's really not fair that the good guys must always be perfect every time, while the bag guys only need to find – or create – a single mistake once.

BleepingComputer even had a conversation with the perps in this instance. They wrote:

Extortion demands were sent to the companies via email and were signed as coming from ShinyHunters. This notorious extortion group has been linked to many high-profile attacks over the years, including those against AT&T, PowerSchool, and the SnowFlake attacks. While ShinyHunters is known to target cloud SaaS applications and website databases, they're not known for these types of social engineering attacks, causing many researchers and the media to attribute some of the Salesforce attacks to Scattered Spider.

However, ShinyHunters told BleepingComputer that the "ShinyHunters" group and "Scattered Spider" are now one and the same. <quote> "Like we have said repeatedly already, Shiny-Hunters and Scattered Spider are one and the same. They provide us with initial access and we conduct the dump and exfiltration of the Salesforce CRM instances. Just like we did with Snowflake."

It is also believed that many of the group's members share their roots in another hacking group known as Lapsus$, which was responsible for numerous attacks in 2022-2023, before some of their members were arrested. Lapsus$ was behind breaches at Rockstar Games, Uber, 2K, Okta, T-Mobile, Microsoft, Ubisoft, and NVIDIA. Like Scattered Spider, Lapsus$ was also adept at social engineering attacks and SIM swap attacks, allowing them to run over billion and trillion-dollar companies' IT defenses.

Over the past couple of years, there have been many arrests linked to all three collectives, so it's not clear if the current threat actors are old threat actors, new ones who have picked up the mantle, or are simply utilizing these names to plant false flags.

This story is interesting because it is another in what has become a long string of examples of the way modern attacks are now occurring: It is now the people, not the technologies, that present the greatest source of vulnerabilities. Therefore, it's people who are now being attacked.

The only recourse I can imagine for any large company with many employees, each and every one of which presents a potentially vulnerable point of entry for bad guys, is to **assume** that inadvertent misconduct **will** occur on the part of any employee. So work to design a network that inherently mistrusts its own users. That's the way our operating systems are now designed. There's a well-understood concept of a system's administrator versus its user.

Of course, it's easy for me to sit back here and armchair quarterback the network architecture that enterprises should design. I don't have the task of actually doing so, and I cannot imagine the difficulty of actually doing so. But for what it's worth, what I'm sure of, what all the evidence teaches us, is that the designers of any contemporary enterprise's information systems **must** design their systems under the assumption that malicious users will be authenticated on that enterprise's network.

It should be clear that having an impenetrable perimeter defense is important. But it's now equally clear that the battle has moved inside and is now being waged against individual employees. The malefactor's goal is to penetrate an employee's human defenses, and from there to move laterally into the enterprise's network. This means that today's and tomorrow's rational security design needs to be resilient against attacks from the inside.

Chrome tests Fingerprint Blocking in Incognito Mode

I just fired up Chrome to see what version was shipping. It's been quite a while since I last launched it, so I got the big "What's New" announcement page. Help/About showed that we're at major version 139 and the goodie I want to talk about isn't due to land until major version 140. Chrome 140 entered into beta two weeks ago on August 6th and is scheduled to begin rolling out to its general audience two weeks from today on Tuesday, September 2, 2025.

So, two weeks from today Chrome will have (get this!) ***Script Blocking in Incognito Mode***. https://chromestatus.com/feature/5188989497376768. The "Overview" says:

> *Mitigating API Misuse for Browser Re-Identification, otherwise known as Script Blocking, is a feature that will block scripts engaging in known, prevalent techniques for browser re-identification in third-party contexts. These techniques typically involve the misuse of existing browser APIs to extract additional information about the user's browser or device characteristics.*
>
> *This feature uses a list-based approach, where only domains marked as "Impacted by Script Blocking" on the Masked Domain List (MDL) in a third-party context will be impacted.*
>
> *When the feature is enabled, Chrome will check network requests against the blocklist. We will reuse Chromium's subresource_filter component, which is responsible for tagging and filtering subresource requests based on page-level activation signals, and a ruleset is used to match URLs for filtering.*
>
> *The enterprise policy name is PrivacySandboxFingerprintingProtectionEnabled.*

The section headlined "Motivation" says:

> *Browser re-identification techniques have been extensively studied by the academic community, highlighting their associated privacy risks. We want to improve user privacy in Incognito mode by blocking such scripts from loading.*

This is not at all what the Brave browser is doing. Brave is deliberately fuzzing the results of various fingerprintable modern browser technologies to prevent ***any and all known AND unknown*** 1st- and 3rd-party fingerprint tracking against a user's wishes.

What Chrome is doing is better than nothing, but a far cry from Brave. In the first place, Chrome is only doing anything for users who are in Incognito mode. And when in Incognito mode, based upon Google's description, Chrome will cross-reference the domain names of any 3rd-party resource fetches against their MDL – Masked Domain List – and if found, will proactively block the execution of any scripting by any resource returned from a fetch from any of those domains.

So on the one hand it's better than Brave in that all potentially troublesome scripting is blocked rather than allowed to run and be fuzzed. But on the other hand it only applies while the user is viewing websites in Incognito mode and it only blocks previously known troublesome domains.

Still it's better than nothing. And it also makes sense that Chrome would do this, since Chrome's MDL is already being used to deliberately obscure the user's IP... which is an extremely cool and useful feature, which I don't think Google and the Chromium developers have received enough credit for.

I've previously noted that despite any other measures we might take, our IPs are likely still providing the strongest possible tracking signal, since they so very rarely change. Given that, it's reasonable to ask what's the point of jumping through all of those other hoops with anti-fingerprinting and cookie erasing and all, if all of our fetches to third party trackers will be made from the same IP?

The Chromium developers clearly understood this. The MDL term, meaning Masked Domain List, is a list of domains from which someone using Incognito mode's Internet IP address will be masked. That's right. Google actually takes it upon themselves to proxy any requests a user in Incognito mode might make to any 3rd-party domain on the MDL.

The MDL is a public, GitHub-hosted list of domains that Chrome treats as higher-risk for cross-site tracking. So when one of those domains loads in a third-party context in Incognito, Chrome provides extra identity protection by routing the request through privacy proxies so that the untrusted 3rd-party site sees requests arriving from a masked IP rather than the user's actual Internet address. That's extremely cool.

As for the MDL, Google defines inclusion criteria, and Disconnect.me evaluates and maintains the list following those criteria. The list is published publicly and maintained on GitHub.

The "naughty list" contains domains that commonly run as a third party across multiple sites and either participate in ads and marketing data flows – so serving/targeting/measuring ads or collecting user data – or appear to collect device and user data which might be usable for cross-context re-identification. Additionally, Chrome also detects widely used JavaScript fingerprinting patterns which can get a domain listed.

The IP proxying has been in place for most of the year. But someone must have also noticed that it would still be possible to run powerful fingerprinting scripts through a proxy which would only be obscuring the user's IP address. So a browser could still be tracked evening without knowing its IP. Consequently, starting with Chrome 140 in two weeks, Incognito mode will be adding 3rd-party script blocking to its existing IP proxying. Props to Google and the Chromium team!

Other Chrome 140 additions
While we're on the subject of Chrome 140, it will be adding a few other notable cool features.

Anyone who has been annoyed by the need to encode text or binary data into URL-safe Base64 ASCII text (and also back the other way) doing this "by hand" in JavaScript, Google writes:

> *base64 is a common way to represent arbitrary binary data as ASCII. JavaScript has Uint8Arrays to work with binary data, but no built-in mechanism to encode that data as base64, nor to take base64'd data and produce a corresponding Uint8Arrays. This is a proposal to fix that. It also adds methods for converting between hex strings and Uint8Arrays.*

And here's a biggie regarding something we've just been talking about recently: a web browser directly accessing the network of its hosting machine. Google writes:

> *Chrome 140 restricts the ability to make requests to the user's local network, requiring a permission prompt. A local network request is any request from a public website to a local IP address or loopback, or from a local website (such as an intranet) to loopback.*
>
> *Gating the ability for websites to perform these requests behind a permission mitigates the risk of cross-site request forgery attacks against local network devices, such as routers. It also reduces the ability of sites to use these requests to fingerprint the user's local network. This permission is restricted to secure contexts. If granted, the permission also relaxes mixed content blocking for local network requests, since many local devices cannot obtain publicly trusted TLS certificates for various reasons.*

This means that IPs within the same network as the browser's host machine will require an affirmative granting of permission before Chrome 140 and later will fetch anything from that IP. I currently access my cable modem at 192.168.100.1, my pfSense firewall is at 192.168.0.1 and our ASUS router is at 192.168.1.1. So in two weeks any attempt to access those devices should produce some sort of "Are you sure?" permission request. That seems minimally intrusive and definitely worthwhile.

One of the things the testers of the DNS Benchmark have noticed, since the Benchmark has always tested remote DNS resolvers to see whether they would block or resolve private IPs (none should), is that the once-common prevention of DNS rebinding attacks has apparently disappeared from the public Internet.

A rebinding attack is something we saw a few months ago, when a public domain name was returning the IP 127.0.0.1. That can be used as a type of black hole to kill traffic, but doing that is not safe, and that was a malicious domain. Returning 0.0.0.0 is much better for null-routing a domain name. If a public domain were to return 192.168.1.1, then asking a browser to connect to a public-appearing domain would cause it to connect to a network's local ASUS router... which is almost certainly not what you would expect – or want – to have happen.

The abuse of this is known as a DNS rebinding attack and there's no clear reason for resolvers of public DNS domains to return non-routable IPs reserved for use within private networks.

Data Brokers keep their data deletion pages from being indexed

The Markup's headline was *"We caught companies making it harder to delete your personal data online"*. I suppose we shouldn't be surprised, but I thought it was interesting. The article's tease said: *"Dozens of companies are hiding how you can delete your personal data, The Markup and CalMatters found. After our reporters reached out for comment, multiple companies have stopped the practice."* The Markup explained what they found:

> *Data brokers are required by California law to provide ways for consumers to request their data be deleted. But good luck finding them. More than 30 of the companies, which collect and sell consumers' personal information, hid their deletion instructions from Google, according to a review by The Markup and CalMatters of hundreds of broker websites. This creates one more obstacle for consumers who want to delete their data.*
>
> *Many of the pages containing the instructions, listed in an official state registry, use code to tell search engines to remove the page entirely from search results. Popular tools like Google and Bing respect the code by excluding pages when responding to users.*

Upon reading that I was tempted to suggest that users ask Perplexity, which probably ignores the please don't index this page signal, but that would have been a cheap shot. <g> They wrote:

> *Data brokers nationwide must register in California under the state's Consumer Privacy Act, which allows Californians to request that their information be removed, that it not be sold or that they get access to it. After reviewing the websites of all 499 data brokers registered with the state, we found 35 had code to stop certain pages from showing up in searches.*

Okay. Hold on. There are 499 data brokers registered with the State of California. Yikes. Who said the data brokerage business is not booming? But in any event, 35 of those 499 had website pages containing search engine non-indexing flags. The Markup continues:

> *According to Matthew Schwartz, a policy analyst at Consumer Reports who studies the California law governing data brokers and other privacy issues, while those companies might be fulfilling the letter of the law by providing a page consumers can use to delete their data, it means little if those consumers can't find the page. Matthew said: "This sounds to me like a clever work around to make it as hard as possible for consumers to find it."*
>
> *After The Markup and CalMatters contacted the data brokers, eight said they would review the code on their websites or remove it entirely, and another two said they had independently deleted the code before being contacted. The Markup and CalMatters later confirmed that nine of the ten companies had removed the code.*
>
> *Two companies said they added the code intentionally to avoid spam at the recommendation of experts and would not change it. The other 24 companies didn't respond to a request for comment; however, three removed the code after The Markup and CalMatters contacted them.*
>
> *After publication, one company that had not previously responded, USPeopleSearch.com, said it had removed the code.*
>
> *Most of the companies that did respond said they were unaware the code was on their pages.*

What?! How'd that get there?? Uh huh. How, indeed. For example:

> *May Haddad, a spokesperson for data company FourthWall, said in an emailed response: "The presence of the code on our opt-out page was indeed an oversight and was not intentional. Our team promptly rectified the issue upon being informed. As a standard practice, all critical pages — including opt-out and privacy pages — are intended to be indexed by default to ensure maximum visibility and accessibility."*

Uh huh. The Markup and CalMatters later confirmed that the code had been removed as of July 31. I still cannot get over that number. One fewer than 500 registered data brokers in California.

> *Some companies that hid their privacy instructions from search engines included a small link at the bottom of their homepage. Accessing it often required scrolling multiple screens, dismissing pop-ups for cookie permissions and newsletter sign-ups and then finding a link that was a fraction the size of the other text on the page.*

This should not surprise anyone, though, right? These companies are scraping and purchasing personal information from everywhere they can about everyone they can. So the last thing they're going to do is invite anyone to delete the data they've purchased, California law notwithstanding. Unless the law explicitly and clearly states that their opt-out pages must be **as** accessible and searchable as any other page, with opt-out links prominently displayed and as visible as any others, and with stiff fines imposed if these requirements are ignored, companies are going to do whatever they can to make it difficult.

Describing two more instances, The Markup wrote:

> *Consumers still faced a serious hurdle when trying to get their information deleted.*
>
> *Take the simple opt-out form for "ipapi", a service offered by Kloudend, Inc. that finds the physical locations of internet visitors based on their IP addresses. People can go to the company's website to request that the company "Do Not Sell" their personal data or to invoke their "Right to Delete" it — but they would have had trouble finding the form, since it contained code excluding it from search results. A spokesperson for Kloudend described the code as an "oversight" and said the page had been changed to be visible to search engines; The Markup and CalMatters confirmed that the code had been removed as of July 31.*
>
> *Telesign, a company that advertises fraud-prevention services for businesses, offers a simple form for "Data Deletion" and "Opt Out / Do Not Sell". But that form is hidden from search engines and other automated systems, and is not linked on its homepage. Instead, consumers must search about 7,000 words into a privacy policy filled with legalese to find a link to the page. A spokesperson for Telesign didn't respond to a request for comment.*

Messaging in Russia

Assuming that we may have some listeners in Russia, this may be of direct interest to them. And for everyone else it's at least interesting as another example of the changing Russian cyber landscape.

Everyone's favorite Russian Internet watchdog – Roskomnadzor – has started restricting voice

and video calls over Meta's WhatsApp messenger and Telegram. Roskomnadzor said the two messengers were used to commit fraud and organize terrorist activities. We know from our previous reporting that there has been some correlation between Telegram use and arrests in Russia. The assumption has been that while the content of any messaging remained secret, the metadata, that is to say the fact of there having been messaging, may remain accessible to those in the position to monitor Telegram's digital traffic.

Forbes Russia reported last Monday that Russia's four largest telcos petitioned the government for the ban. They argued that a ban would return traffic to the phone networks and increase their revenue. So rather self-serving there. The ban also comes as the Russian government is pushing users over to its own soon-to-be-released national instant messenger, an app called Max.

And to that end: The Kremlin has ordered government officials to move their Telegram channels to the country's domestic messaging app Max. Officials will still be allowed to have accounts on other platforms, but the Max channels are now mandatory. The official Max accounts are expected to go live in the coming weeks, when the Max app is expected to come out of beta and become broadly available to the public. So that's rather clever. No one is saying they cannot also use something else. But if every government official is required to have an account on Max, it's foreseeable that over time people would likely gravitate to it, since all other officials will be there.

NIST's Crypto for IoT

The United States' NIST, the National Institute of Standards and Technology, is the organization the entire world has come to rely upon to corral and organize technical domain experts and manage the complex development of current and next generation technologies. Even though the results emerging from these efforts are open and free for everyone to use, there's still a desperate need for there to be universally agreed upon standards for things like communications protocols, device interfaces, and encryption algorithms. Nothing works for anyone unless we have as a bare minimum interoperability among interacting systems. NIST provides the required organization to see that we have at least that.

In fulfilling that mission, last Wednesday, NIST posted some welcome news under their headline: *"NIST Finalizes 'Lightweight Cryptography' Standard to Protect Small Devices"* with the teaser *"Four related algorithms are now ready for use to protect data created and transmitted by the Internet of Things and other electronics."* And NIST's announcement led with three bulletpoints:

- *Many networked devices do not possess the electronic resources that larger computers do, but they still need protection from cyberattacks. NIST's lightweight cryptography standard will help.*
- *The four algorithms in the standard require less computing power and time than more conventional cryptographic methods do, making them useful for securing data from resource-constrained devices such as those making up the Internet of Things.*
- *NIST has finalized the standard after a multiyear public review process followed by extensive interaction with the design community.*

This is crucially important technology for our future, so I want to share NIST's overview and their brief summary comments about these four new finalized and now standardized algorithms:

NIST's newly finalized lightweight cryptography standard provides a defense from cyberattacks for even the smallest of networked electronic devices.

Released as Ascon-Based Lightweight Cryptography Standards for Constrained Devices (NIST Special Publication 800-232), the standard contains tools designed to protect information created and transmitted by the billions of devices that form the Internet of Things (IoT) as well as other small electronics, such as RFID tags and medical implants. Miniature technologies like these often possess far fewer computational resources than computers or smartphones do, but they still need protection from cyberattacks. The answer is lightweight cryptography, which is designed to defend these sorts of resource-constrained devices.

I'll break in here to comment that all around us we see everything becoming smaller and lighter weight and running on smaller and smaller batteries. Leo's upper body is becoming encrusted with various AI monitoring, recording and summarizing technologies. In many cases they need to communicate through the air and privacy may be important.

Or take the case of wireless keyboards. We've seen how past keyboards used incredibly lame fixed byte XOR encoding, which statically flipped some of the bits of each transmitted byte. Determining the byte for any given keyboard and decrypting all of the keystrokes would make a great junior high school computer science fair project, because it's about at a 7th or 8th grade level of difficulty. Keyboards were forced to go to Bluetooth or proprietary systems to obtain greater security. But unless we have encryption that is both secure and lightweight – meaning requiring very little or very economical computation – either battery life or security will need to be compromised. Having a NIST-approved standard that's both secure and lightweight at the same time will translate directly into superior IoT consumer products.

NIST computer scientist Kerry McKay, who co-led the project with her NIST colleague Meltem Sönmez Turan, said: "We encourage the use of this new lightweight cryptography standard wherever resource constraints have hindered the adoption of cryptography. It will benefit industries that build devices ranging from smart home appliances to car-mounted toll registers to medical implants. One thing these electronics have in common is the need to fine-tune the amount of energy, time and space it takes to do cryptography. This standard fits their needs."

The standard is built around a group of cryptographic algorithms in the Ascon family, which NIST selected in 2023 as the planned basis for its lightweight cryptography standard after a multiround public review process. Ascon was developed in 2014 by a team of cryptographers from Graz University of Technology, Infineon Technologies and Radboud University. In 2019 it emerged as the primary choice for lightweight encryption in the CAESAR competition. This all showed that Ascon had withstood years of examination by cryptographers.

In the standard are four variants from the Ascon family that give designers different options for different use cases. The variants focus on two of the main tasks of lightweight cryptography: authenticated encryption with associated data (AEAD) and hashing.

AEAD algorithms are where the world has ended up with authenticated encryption because it's extremely useful. For example, I used it to securely store SQRL's user identity. For SQRL, there needed to be some parameters of the user's identity that were accessible without the user's secret key – in other words, stored as plaintext – and other parameters that needed to be protected by the user's secret, so stored encrypted. But all of the information, whether stored without encryption – which is known as "Associated Data" or with encryption, it all needed to be protected against tampering. If any bit of the stored data, whether the encrypted data or the

visibly readable plaintext, was altered, the authentication of the entire package would be broken. So these AEAD algorithms are very cool with many applications.

Here's what NIST says about the four ASCON algorithms:

> ***ASCON-128 AEAD*** *is useful when a device needs to encrypt its data, verify the authenticity of the data, or — crucially — both. A common weakness of small devices is their vulnerability to "side-channel attacks," in which an attacker can extract sensitive information by observing physical characteristics like power consumption or timing. While no cryptographic algorithm is inherently immune to such attacks, ASCON is designed to support side-channel-resistant implementations more easily than many traditional algorithms. Devices that can benefit from its approach include RFID tags, implanted medical devices, and toll-registration transponders attached to car windshields.*
>
> ***ASCON-Hash 256*** *takes all the data it encrypts and uses it to create a short "hash" a few characters long, which functions like a fingerprint of the data. Even a small change to the original data results in an instantly recognizable change in the hash, making the algorithm useful for maintaining the data's integrity — such as during a software update, to ensure that no malware has crept in. Other uses are for protecting passwords and the digital signatures we use in online bank transfers. It is a lightweight alternative to NIST's SHA-3 family of hash algorithms, which are widely used for many of the same purposes.*
>
> ***ASCON-XOF 128*** *and* ***ASCON-CXOF 128*** *are hash functions with a twist: Both algorithms allow the user to change the size of the hash. This option can benefit small devices because using shorter hashes allows the device to spend less time and energy on the encryption process.*
>
> *The CXOF variant also adds the ability to attach a customized "label" a few characters long to the hash. If many small devices perform the same encryption operation, there is a small but significant chance that two of them could output the same hash, which would offer attackers a clue about how to defeat the encryption. Adding customized labels allows users to sidestep this potential problem.*
>
> *McKay said the NIST team intends the standard not only to be of immediate use, but also to be expandable to meet future needs. She said: "We've taken the community's feedback and tried to provide a standard that can be easily followed and implemented, but we are also trying to be forward-looking in terms of being able to build on it. There are additional functionalities people have requested that we might add down the road, such as a dedicated message authentication code. We plan to start considering these possibilities very soon."*

So the world now has a new set of NIST-approved, well-vetted, easy-to-implement lightweight and secure cryptography standards. I have a link to NIST's announcement and to the 52-page specification for anyone who wishes to dig deeper:

https://www.nist.gov/news-events/news/2025/08/nist-finalizes-lightweight-cryptography-standard-protect-small-devices

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-232.pdf

Miscellany

SyncThing's major v2.0 upgrade

SyncThing's version announcement page started off with: "*This is the first release of the new 2.0 series. Expect some rough edges and keep a sense of adventure!* 🙏" There are places where a sense of adventure makes sense. But SyncThing takes an honored place in the middle of my workflow and "adventure" is not something I'm hoping to be treated to by my multi-system backup solution. What's more, the initial UI pop-up warned: *"This is a major version upgrade. A new major version may not be compatible with previous versions. Please consult the release notes before performing a major upgrade."* This is particularly salient for me because one of the systems I'm still sync'ing with SyncThing is a Windows 7 machine and I had to turn off its automatic updating quite a while ago when SyncThing's newer releases stopped supporting it. By the year's end I plan to be consolidating my two locations into one, and that will spell the end of the Windows 7 machine. But that hasn't happened yet.

SyncThing is this podcast's favorite file synchronization tool. Leo and I have both chosen it and we depend upon it as our primary cross-machine synchronizer. So I wanted to quickly note the changes to SyncThing with its move to v2.0:

- Database backend switched from LevelDB to SQLite. There is a migration on first launch which can be lengthy for larger setups. The new database is easier to understand and maintain and, hopefully, less buggy.

- The logging format has changed to use structured log entries (a message plus several key-value pairs). Additionally, we can now control the log level per package, and a new log level WARNING has been inserted between INFO and ERROR (which was previously known as WARNING...). The INFO level has become more verbose, indicating the sync actions taken by Syncthing. A new command line flag `--log-level` sets the default log level for all packages, and the `STTRACE` environment variable and GUI has been updated to set log levels per package. The `--verbose` and `--logflags` command line options have been removed and will be ignored if given.

- Deleted items are no longer kept forever in the database, instead they are forgotten after fifteen months. If your use case requires deletes to take effect after more than a fifteen month delay, set the `--db-delete-retention-interval` command line option or corresponding environment variable to zero, or a longer time interval of your choosing.

- Modernised command line options parsing. Old single-dash long options are no longer supported, e.g. `-home` must be given as `--home`. Some options have been renamed, others have become subcommands. All serve options are now also accepted as environment variables. See `syncthing --help` and `syncthing serve --help` for details.

- Rolling hash detection of shifted data is no longer supported as this effectively never helped. Instead, scanning and syncing is faster and more efficient without it.

- A "default folder" is no longer created on first startup.

- Multiple connections are now used by default between v2 devices. The new default value is to use three connections: one for index metadata and two for data exchange.

- The following platforms unfortunately no longer get prebuilt binaries for download at syncthing.net and on GitHub, due to complexities related to cross compilation with SQLite:
  - dragonfly/amd64
  - illumos/amd64 and solaris/amd64
  - linux/ppc64
  - netbsd/*
  - openbsd/386 and openbsd/arm
  - windows/arm
- The handling of conflict resolution involving deleted files has changed. A delete can now be the winning outcome of conflict resolution, resulting in the deleted file being moved to a conflict copy.

The biggest functionality change is the decision not to retain deleted files forever. I assume that the long-term endless collection of every past file, depending upon the application, might have finally caused the development team to reassess their previous "forever" policy. Still, a 15-month default seems ample.

Multiple connections between peers seems like a nice addition. But I suppose that the lack of prebuilt binaries might be a bit of an inconvenience, especially for Windows on ARM. Building binaries from source is a common occurrence for the various Linuxes and Unixes.

The final thing I'll mention is that since the v2.0.0 release I've watched the sub-version number advance to v2.0.1 and then to v2.0.2. That's to be expected following feedback from a greater number of users after release. And we don't know what sorts of "Adventures" they may have had with the very first cut releases. I scanned the changelog and it appeared that the changes may have related mostly to the need for some users to build their own binaries. There are tweaks to minimum compile-time library versions and such.

Since my current SyncThing v1.30.0 for 64-bit Windows, which was built on June 20th les than two months ago, is working perfectly for me, I see no need to go seeking "Adventure" from my cross-device file synchronizing system. So I'll be remaining where I am until I decommission that old Win7 machine a few months from now.
https://github.com/syncthing/syncthing/releases/tag/v2.0.0

Sci-Fi

Alien:Earth

IMDB's ranking has dropped to 7.8 following last Tuesday's wider release of the first two episodes. And that makes sense given that the earlier release during Comic-Con would likely be a strongly skewed demographic. My wife was somewhat bored by it and is glad that she'll only need to sit through another six episode hours of the first season. But I'm curious enough that I want to see what the writers do with the various pieces they've set in motion.

It was interesting to see all of the "Alien" mythology present, and to appreciate how much of it we've internalized from the previous movies. You see a ragged-edged hole in the floor, and you immediately think "ah, yes, molecular acid for blood." You see some egg-shaped pods split at their top sitting beneath a blue-tinged mist and you think "don't get your face too close to those!" So there was a great deal of familiar comfort in what we saw during last week's two introductory episodes. There were also some promising new critters that we don't yet know much about. Perhaps they will be further developed. And I could see what Lorrie meant. The aliens themselves have become rather boring because they are now so well known. We now know what they look like. We're aware of their entire life-cycle. So we have a creature that is pure animal, has no language, cannot be negotiated with, is physically huge, ruthlessly brutal and effectively unstoppable. While that's terrifying if it's in your neighborhood, it's also somewhat limited as a plot device. The value of the Alien franchise has always been the human interest side of the crew's reactions to this creature and the events surrounding it. Without that, we only have what Bill Paxton's character said with some disgust: "it's a bug hunt."

So the most interesting new feature, which I'm sure is the intended focus of the series, are the recent earth-bound experiments with a new "hybrid" which is created by transferring a human consciousness into a fully synthetic super-humanly strong and highly-intelligent body. The female leader of that group has already manifested an unexpected ability and I'm curious to know what she and her fellow hybrids will do next. So this evening I'll be watching episode #3 – not with super-high expectations of being blown away, but at least with some curiosity.

The Sad Case of ScriptCase

For this week's podcast topic I wanted to focus upon an interesting vulnerability that won't make any headlines. As I dug into the story I started to get a sinking feeling, not about it specifically, but about something we often touch on here, which is the state of today's cybersecurity environment. I titled today's podcast "The Sad Case of ScriptCase," not because what I discovered about ScriptCase was special. What was sad about ScriptCase was that it was not special. So let's first back up to take a look at this flaw, which somewhere around 2800 of ScriptCase's users have remained vulnerable to months after it was discovered and patched by its publisher, then we'll look at what this all means.

This story begins with a vulnerability disclosure posting by the well known cybersecurity company Synacktiv. Their posting on the 4th of July was titled *"ScriptCase - Pre-Authenticated Remote Command Execution."* Now, everyone who follows this podcast will be well aware of the severity of the inherent severity of any pre-authenticated remote command execution. "Pre-authenticated" means that anyone anywhere can remotely execute commands on the targeted system without any need to be authenticated because this remote command execution is able to somehow be induced before any authentication is required. That being the case, we would not be surprised to see that the severity of this was set to CRITICAL.

Synacktiv wrote:

> *ScriptCase is a low-code platform that generates PHP web applications. Developers use a graphical user interface to design and generate their website. "Production Environment" is the name of an extension of ScriptCase that will be called "production console" in the advisory for clarity. It is an administrative field to manage database connections and directories. While ScriptCase itself is not necessarily deployed with the website, the production console mostly always is.*
>
> *Pre-authenticated remote command execution is achieved by chaining two vulnerabilities: the first is the ability to change the administrator password of the production console under certain conditions, and the second is a simple authenticated remote command execution in the connection features where user input is directly concatenated to an ssh system command.*

So this seems like a rather straightforward mess for anyone who has this system deployed in the field. Synacktiv discovered a means for remotely changing the administrative password, and they also discovered that remote user-supplied data is being directly concatenated onto an SSH command.

The ability for an anonymous unknown remote user to remotely change an administrative password is clearly a very big bad mistake. We all know that I draw a very clear distinction between mistakes – that happen – and policies or designs that are necessarily the product of some deliberation. When you hear that a system allows anyone to remotely change an administrator's password you think that it must be a horrible bug. But digging into the details of this reveals that the author of this system failed to provide the safeguards we all live with daily. Specifically, any new password can be set without the user providing the current password. Ouch! So this, too, was a design-level decision.

So here are the interesting details of this first mistake: There's a flaw in the system's PHP codes logical flow. The first thing the "change_password" function does is check to see whether the user's session has an "is_authenticated" variable defined for it.

Since the "is_authenticated" variable is only created inside the "initialize_session" function, the intent was that only someone who was already authenticated would be able to change their password. This perhaps excuses the lack of need to provide the current password, though if that had been required this house of cards would not have collapsed.

The clear intention was that at the time the user was being authenticated the "initialize_session" function would be called to initialize the session and that would define the "is_authenticated" variable for the future. What the author of this code failed to take into account is that a failed login attempt, trivially caused simply by directly calling the "login.php" function with an HTTP GET query, causes the "initialize_session" function to be called, and thus the "is_authenticated" variable would be created.

At that point, the system falsely believes that the user's session has been authenticated and the "change_password" function, which checks for the presence of that variable, will then allow itself to be remotely called. And since that function requires no provision of the user's current password, any unauthenticated remote user is able to set whatever password they choose for the system's administrator simply by deliberately failing a first login attempt, setting the administrative password to anything they wish, then logging on for real.

So we have an unintended code path, coupled with the bad design pattern of not always requiring a user's current password when they're requesting its change, and we have the first half of a critical remote command execution vulnerability.

The Synacktiv guys wrote:

> *An attacker can arbitrarily reset the password of the administrator of the production console, and so take it over. With this access, the attacker could retrieve database credentials and get access to them. As the production console is also vulnerable, the attacker could also leverage it to gain access to the server.*
>
> *Recommendation: Access to the password reset feature should be given only to authenticated users (change the condition in the "change_password" function). Also, it should be based only on the session cookie. The "change_password" function should not take an email argument from the user, but extract it from the session.*
>
> *While waiting for an official fix from the vendor, one should restrict the access to the ScriptCase Production Environment extension completely Blocking /prod/lib/php/devel/iface /login.php and /prod/lib/php/nm_ini_manager2.php should be enough to prevent any unwanted connection as well as the exploitation of the password reset vulnerability.*

So at this point we're remotely logged on as the system's administrator. So what's next?

The "what's next" is the exploitation of CVE-2025-47228, a shell injection allowing remote command execution. The exact sequence of actions is somewhat too dense to explain verbally on the podcast, but the situation is similar to what we've already seen. The developer of this low-code PHP-driven website creation system developed a system to allow less sophisticated users to create a complex PHP-driven website using a graphical user interface. The idea was that it would not be necessary to understand PHP to create a website.

Unfortunately, Synacktiv's analysis of the design and implementation of this tool would lead an impartial observer to conclude that this tool's developer also failed to fully understand the operation of PHP enough to create a system that not only worked, but worked securely.

So what are the real world consequences of all this?

For that we turn to VulnCheck's recent posting last Thursday, which is what brought all of this to my attention. On August 14th, VulnCheck posted under their headline: *"ScriptCase - Hunt It, Exploit It, Defend It"* They began with three key takeaways:

- *Hundreds of ScriptCase instances remain exposed a month after disclosure, with attackers actively scanning for them.*
- *Exploitation is simple, requiring only a few curl commands once a target is found, allowing full remote code execution.*
- *Clear detection paths exist, including version strings, network signatures, and suspicious processes or PHP files in the webroot.*

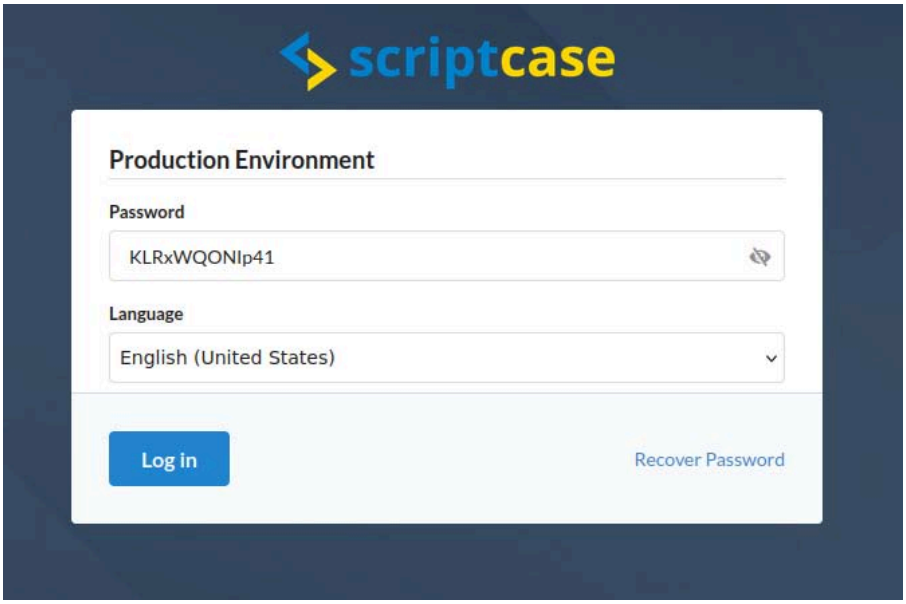
They write:

> *One month ago, Synacktiv published their disclosure and deep dive on a vulnerability chain affecting ScriptCase. The vulnerabilities, CVE-2025-47227 and '28, are an unauthenticated password reset and an authenticated command injection that, when combined, give an unauthenticated attacker full remote code execution. And yet, despite public disclosure, functional exploits, and available patches, hundreds of ScriptCase instances remain exposed on the Internet.*
>
> *That leaves the obvious question: does this matter enough to go hunting?*
>
> *At VulnCheck, one way we determine if a vulnerability matters is by looking for targets online. The logic is pretty easy: if there are zero targets online, well, who cares? If there are many targets online, then we care. If it's somewhere in between 0 and many... it depends.*
>
> *Naively, we started with a Shodan query of title:"ScriptCase". The results were annoying.*
>
> *Annoying because they aren't real ScriptCase servers at all. These are honeypots. Frankenpots that seemingly pollute every single query. We've written before about the problem in "There are Too Many Damn Honeypots", and this is another textbook case.*
>
> *But the fact that there are honeypots suggests that others care about ScriptCase too. So we grabbed a copy of the software, built a Shodan query to avoid the decoys, and, in a rare win, even got a simple Google search to work for finding actual instances. (The AI-slopification of Google has largely destroyed this, so this felt like a small miracle).*
>
> *The VulnCheck Initial Access Intelligence team routinely develops queries for Shodan, FOFA, ZoomEye, and Censys to track down vulnerable targets. While building out our fingerprints for ScriptCase on these services, we also found that our friends over at driftnet had turned up a solid hit count of roughly **2,800** ScriptCase servers exposed to the Internet via their Scan Content functionality.*
>
> *Finally, it's not just researchers looking for ScriptCase. GreyNoise is tracking a couple **dozen** known malicious IPs scanning specifically for /scriptcase/. That's proof attackers are on the hunt too. At the end of the day, you've got all the ingredients to answer "does this matter?" There are discoverable targets online. There's a public proof of concept. And attackers are actively looking for these systems. That matters.*

And now, having determined that this matters, they look at exploitation, writing:

If finding vulnerable ScriptCase servers is straightforward, exploiting them is even easier. Synacktiv's blog and proof-of-concept go into detail, but the reality is that it boils down to just a few curl commands. No custom tooling required.

Once the password reset has been achieved, we can navigate to the production environment login page and authenticate with the new credentials.



Once authenticated, we land in the production environment. With access to the production environment, we can move on to CVE-2025-47228, a command injection vulnerability in the connection creation and testing feature. The injection logic lives in a modified version of the third-party library ADOdb. First, the command is built in $str_command using attacker-provided variables.

With a webshell or reverse shell in place, the exploitation chain is complete, but that's only half the story. For defenders, the question becomes: how do you spot this activity before or after it happens?

Defenders should check whether their ScriptCase deployment is vulnerable. By default, the landing page exposes a version string in its HTML, which can be compared directly against patched releases. We built a passive version scanner to run across Shodan data, and 57% of observed instances still reported a vulnerable version.

That's as of the publication date of this, which was last Thursday. They conclude:

Whether you're hunting, exploiting, or defending, the playbook is straightforward: know how to find vulnerable targets, understand how the exploit chain works, and have clear detection and response strategies in place. The attackers looking for /scriptcase/ aren't waiting for you to patch, and the sooner you close these holes, the less likely you are to see your own server in someone else's shell prompt.

One final piece of this that I didn't yet share was Synacktiv's disclosure timeline. Although they patiently waited until July 4th before their public release, the developer of ScriptCase surely tried their patience:

Date	Description
2025.02.18	First message sent to the editor.
2025.03.12	Contact support via live tchat.
2025.03.20	Advisory report sent to the editor.
2025.03.28	First response from the editor.
2025.04.04	Editor asks to re-test the vulnerability on latest version.
2025.04.29	Synacktiv confirms the vulnerability still works on latest versions (see affected versions).
2025.05.15	Synacktiv contacts the editor for a status update on the progress of the vulnerability analysis.
2025.05.30	Synacktiv contacts the editor for a status update on the progress of the vulnerability analysis.
2025.06.05	Synacktiv sends the exploitation script to the editor.
2025.07.04	Public release

So, from their initial contact which occurred on the 18th of February to their eventual release of the exploitation details on the 4th of July, nearly five months elapsed with Synacktiv typically responding within days and ScriptCase's side often responding either never or only after Synacktiv had waited patiently for several weeks. As I have repeatedly observed, the bizarre system of vulnerability discovery and reporting and updates and patching – and never patching – is so badly broken.

ScriptCase <https://www.scriptcase.net/> has one of those stunning, lovely state-of-the-art websites with beautiful graphics, tasteful imagery and design, that would inspire confidence in anyone who visited. The company is based in Orlando Florida and along the bottom of the first page the names of several of their more prominent 45,000+ customers scroll. If you wait a minute you'll see the names of Bosch, HP, Hyundai and Yamaha slide past.

One reason ScriptCase might have taken so long to respond to Synacktiv's many attempts to communicate is that their developers are far too busy just trying to keep up, fixing the many other things that appear to be broken with the product. I thought Microsoft was bad. Okay, Microsoft is bad, but these guys are even worse. They are the exact opposite of my style of product development, where code is carefully developed and well tested. When it's published it's finished and it's working so well that it can and often does go decades without ever being touched again. I would say that the ScriptCase code goes about three days before being touched again, except that then it's touched again and again and again, continuously.

A look through their ChangeLog, which goes back 11 years – and only gets them back to major version 8 < https://www.scriptcase.net/changelog/ > reveals that they've been updating their product every few days; sometimes 3 days, sometimes 4 days, sometimes it's 5 days; I've seen a week. And this appears to have been going on since the early 2000's. Every few days they release another update and their changelog reports 10 or so important-seeming things they've just fixed. Maybe they're ex-Microsoft engineers?

This development style drives me nuts. One of the reasons I stopped using GitLab for development tracking was that its developers would never leave it alone. They were constantly

spewing out new features that were often mixed in critical "must patch immediately" updates. The process to update had never received much attention. It was not clean or seamless. Each one was a mess and it was not possible to skip any – it was an endless series of incrementals and it was a mess. As our listeners know, I've also been annoyed by Notepad++'s author who, similarly, has a seemingly never-ending list of things to fix.

If today's model of vulnerability discovery, patching and updating is already badly broken because users get tired of stopping everything they're doing to update some software that's working fine for them, what do you imagine happens when new versions of a non mission critical website authoring system are being offered weekly? No one cares. And before long no one will bother to install updates.

I looked through ScriptCase's ChangeLog for the two CVE's that Synacktiv went so far out of their way to report and manage. There's no sign of them anywhere.

Last Wednesday on August 13th they fixed four security problems:

- CVE(26188) Missing Permissions Policy in the Scriptcase environment.
- CVE(26202) Missing 'Cache-Control' in the Scriptcase environment interface.
- CVE(26203) Missing Content Security Policy Instances in the Scriptcase environment.
- CVE(26206) Missing 'X-Frame-Options' Header Instances in the Scriptcase environment.

Tuesday August 5th they fixed another two:

- CVE(26024) SHELL INJECTION (REMOTE COMMAND EXECUTION) in production environment. Production environment needs to be updated.
- CVE(26182) Duplicate HTTP Headers Detected in Scriptcase applications.

The Shell Injection Remote Command Execution in production environment sure sounds like Synacktiv's, but it has a different CVE and there's no mention of, or thanks anywhere, to Synacktiv.

VulnCheck noted that more than half of the publicly accessible instances of ScriptCase were still vulnerable a month after the disclosure and that dozens of known malicious IPs had been seen actively scanning for vulnerable systems. Dozens!

And we all know where this story ends. Every one of those enterprises that made the terminal mistake of giving this far-from-secure ScriptCase system any presence on the public Internet, almost certainly without need, will find itself ransomed and extorted. I never want to see that happen. No one ever deserves that. But the saddest thing is that the correct lesson will never be learned from the experience.

While we've made an example out of ScriptCase, they are more the rule than the exception. There's now a massive industry composed of super-slick appearing fancy websites which front for not-very-professionally designed software that nevertheless does the job and supports its existence.

We've spent a great deal of time on this podcast examining the extreme difficulty of making any software securely publicly accessible. The only rational conclusion is that this should never be done unless "public accessibility" is the entire purpose of the software. Public accessibility is the entire purpose of a public web server, a public email server, or a DNS server. But it is assuredly not the purpose of the "low-code" ScriptCase website designer.

ScriptCase does not exist for the purpose of being on the public Internet. It has no purpose or reason for being widely visible on the Internet. And THAT is the lesson that should be learned from this. ***Not*** *"Oops! A bug was found in some random software system we use and before we could update with the patches high-power super-skilled anti- Western genius hackers in China or Russia got into our system and are now holding us for ransom."*

No!! – that is NOT the lesson. There will always be bugs, just like this, occurring in random networked software. Always. And the anti-Western genius hackers are also never going away. They're now part of the ecosystem, too. So we should not be waiting around for the day when all the bugs are gone and the hackers have been arrested. That day is never coming. Ever.

In the same way – and following the same philosophy – that today's IT designers need to design their networks so that malicious insiders cannot damage the company, the IT managers need to understand that the **only only only** server-style systems that can be publicly visible to everyone everywhere are the servers that are expressly designed to be publicly exposed – those whose sole purpose is to offer widely available public services.

THAT is the proper lesson to take away. It does not matter whether a server appears to require an identity authenticated login. It doesn't matter. We've seen this over and over and over. How many times are we going to point the finger at this mistake or that mistake or they didn't update their software, before we start to realize that the actual mistake is ever attaching anything to the public Internet that does not, by virtue of its purpose, need to be widely visible to everyone everywhere?

I titled today's podcast "The Sad Case of ScriptCase," not because what I discovered about ScriptCase was special, but because it was so sadly common. No company should have become a victim to ScriptCase's mistake because no company should have ever made their ScriptCase instance publicly visible to everyone, everywhere, on the Internet.

Any company that rigorously adopts and enforces the policy and philosophy of **never** having anything publicly visible to everyone everywhere unless that is the server's entire purpose will automatically be protecting itself from **all** of the ScriptCase's now and in the future.

Bugs are never going away, ever. And neither are bad guys. So it should be obvious that the only possible solution is to make certain that the bad guys can never get their hands on those bugs.