Security Now! #891 - 10-04-22

Poisoning Akamai

This week on Security Now!

This week we examine a puzzlingly insecure implementation by Microsoft in Teams' design and at their complete re-write of Microsoft Defender Smartscreen. Roskomnadzor strikes again, and Exchange Server is again under serious attack with a new 0-day. CloudFlare introduces Turnstile, their free CAPTCHA improvement and Google published a fabulously engaging 6-video YouTube series under the banner: “Hacking Google.” We'll then spend some time sharing and replying to listener feedback before we examine a breathtaking flaw that was discovered in Akamai's global CDN caching, and what became of it.

What could possibly go wrong?...

"fooquestionmark posted:

I have a 5000 gallon above ground pool in my basement. It feels nice down there but the water is freezing. I have a tiny ass pump on it right now that kind of flows water but I'm wanting to heat the hole pool to a reasonable temperature.

For reference.

I want something cheap that won't melt the pool as it is rubber. I know I'm not the only person out there with a pool in their basement.

Security News

Microsoft Teams — Unnecessarily Insecure

Three weeks ago, the security firm Vectra, published a report which closely examined the way Microsoft Teams manages its users' application authentication. Their report is long and we don't need the nitty gritty to clearly understand what's going on. So I'm just going to share two small pieces: https://www.vectra.ai/blogpost/undermining-microsoft-teams-security-by-mining-tokens

In their overview they explain:

In August 2022, the Vectra Protect team identified a post-exploitation opportunity allowing malicious actors with sufficient local or remote file system access to steal valid user credentials from Microsoft Teams due to their plaintext storage on disk. This plaintext credential management was determined to impact all commercial and GCC Desktop Teams clients for Windows, Mac, and Linux.

While credential harvesting from memory is a common post-exploitation step, we believe that lowering the bar necessary to harvest creds down to just simple read access to the file system expands opportunities for an adversary, simplifies their task, and is particularly interesting when stolen credentials offer an opportunity to retain user access unencumbered by otherwise pesky Multi-Factor Authentication (MFA) speedbumps.

With these tokens, attackers can assume the token holder's identity for any actions possible through the Microsoft Teams client, including using that token for accessing Microsoft Graph API functions from an attacker's system. Additionally, these tokens are equally valid with MFA-enabled accounts, creating an allowance to bypass MFA checks during ongoing use.

Microsoft is aware of this issue but indicated it did not meet their bar for immediate servicing.

Microsoft stores these credentials to create a seamless single sign-on experience within the desktop application. However, the implementation of these security choices lowers the bar.

*Anyone who installs and uses the Microsoft Teams client in this state is storing the credentials needed to perform **any** action possible through the Teams UI, even when Teams is shut down. When these tokens are stolen, it enables attackers to modify SharePoint files, Outlook mail and calendars, and Teams chat files. Attackers can tamper with legitimate communications within an organization by selectively destroying, exfiltrating, or engaging in targeted phishing attacks.*

The thing that truly frightens us is the proliferation of post-MFA user tokens across an environment – it enables subsequent attacks that do not require additional special permissions or advanced malware to get away with major internal damage. With enough compromised machines, attackers can orchestrate communications within an organization. Assuming full control of critical seats—like a company's Head of Engineering, CEO, or CFO—attackers can convince users to perform tasks damaging to the organization. How do you practice phish testing for this?

Okay. So this is one of those "head buried deeply in the sand" issues which we've increasingly been encountering from Microsoft. The kindest way to interpret this in Microsoft's favor is to suggest that Microsoft has structured itself so that it's deliberately cut off from the outside. Someone who has no authority or power is running interference and responds to any offering made at the foot of the Ivory Tower by incanting the phrase: *"We are aware of this and it is not a security concern."* And that's as far as any inquiry gets. If history is to repeat itself, especially now that this problem is well known, there will eventually be some egregious abuse of what is obviously a totally unnecessary and easily avoidable security weakness in Teams. At that point, a wholly unnecessary emergency will ensue and Teams will have this behavior-by-design changed.

It's totally true that having persistent and static access to a previous successful authentication creates a standing vulnerability. Perhaps that's been done so that other components such as Skype and Outlook are able to share in this authentication — as, indeed, they are. But in an alternative design, they could share a common authentication service which then relies upon encrypted authentication tokens which would at least tie the tokens to the local machine's authentication service, perhaps its TPM, and would make them much more tricky to abuse. Instead, Microsoft has chosen for whatever reason to simply store them in a well known location in every local machine's file system where they are accessible not only to all Microsoft components, but also to anyone else who might wish to abuse this implied trust.

Okay. But Vectra also had another interesting piece of background to share. They gave this section the clever title *"Electron – a Security Negative"* It was clever because, of course, we've all agreed that electrons carry a negative charge. Vectra is suggesting that the Electron development platform carries "Negative Security." Here's what they explained about Microsoft Teams use of the Electron application platform:

> *Microsoft Teams is an Electron-based app. Electron works by creating a web application that runs through a customized browser. This is very convenient and makes development quick and easy. However, running a web browser within the context of an application requires traditional browser data like cookies, session strings, and logs.*
>
> *This is where the root of this issue lies as Electron does not support standard browser controls like encryption, and system-protected file locations are not supported by Electron out of the box but must be managed effectively to remain secure. Therefore, by default the way Electron works incentivizes creating overly transparent applications. Since Electron obfuscates the complexities of creating the application, it is safe to assume that some developers may be unaware of the ramifications of their design decisions and it is common to hear application security researchers bemoan the use of this framework due to critical security oversights.*

Phrased another way, we might say that Microsoft Teams' choice to use the "easy to use" Electron development model, which employs JavaScript, HTML, and CSS, encourages rapid and easy application development by less experienced developers. From what Vectra said, it also sounds as though electron's browser-centric development environment encounters significant resistance when trying to do things like storing encrypted data into the file system. If that's true then we have another case of Microsoft placing its short term needs before long term quality.

Microsoft Defender Smartscreen — Re-written from scratch
In a posting last Thursday the 29th titled *"More reliable web defense"* Microsoft explained that they had scrapped and entirely re-written their Edge browser's built-in Smartscreen library. https://blogs.windows.com/msedgedev/2022/09/29/more-reliable-web-defense/

> *Starting in Microsoft Edge 103, users can navigate the internet with more reliable web defense thanks to the updated Microsoft Defender Smartscreen library that ships with Microsoft Edge on Windows. The updated SmartScreen library was completely re-written to improve reliability, performance, and cross-platform portability. These benefits are the foundation leading up to the security improvements that will increase our ability to protect users from emerging threats.*

And, at the end, they noted that:

> *For enterprise customers who experience compatibility issues and need to revert to the legacy Microsoft Defender SmartScreen, we added a temporary policy called "NewSmartScreenLibraryEnabled". This policy will become obsolete in Microsoft Edge 108.*

It's unclear why anyone would have compatibility issues unless something they were doing was tripping Smartscreen false-positive responses. But since we're currently at release 106 and the option to revert will only remain available in the next release, any enterprise having trouble with the re-write should address those problems quickly.

I'm a huge proponent of wholesale re-writes. As we know, we haven't yet figured out how to evolve software gracefully. Part of the problem is that the various challenges software might face once released into the field are often not fully appreciated by those who design and write it. So the process of watching an initial release interact with the world teaches its designers a lot. The first reaction is to patch over any shortfalls in the original design. But once those patches' patches have acquired patches, it's often the case that the best solution is to quit patching the patches and take everything that is now understood about the problem and start over.

So, bravo to Microsoft for deciding to start over. We'll never know what precipitated that decision, but Edge's users will likely be the winners.

Roskomnadzor blocks Soundcloud:
I never pass up the opportunity to mention Roskomnadzor! An opportunity presented itself when Roskomnadzor added the popular music streaming platform, SoundCloud to Russia's nationwide Internet blocklist. I have the URL to Roskomnadzor's blocklist page. And thank goodness for the Web's Western heritage, since at least the URL uses the Latin alphabet. I'm unable to make heads nor tails of anything on that blocklist page after its URL: https://blocklist.rkn.gov.ru/

The presumption is that the blockage was the result of SoundCloud's hosting of podcasts which were discussing Russia's invasion of Ukraine. You can't have any of that in a repressive totalitarian regime, so... no more SoundCloud.

Exchange Server under attack again
A Vietnamese group named "GTSC" discovered an active in-the-wild Exchange Server 0-day.

They wrote:

> *At the beginning of August 2022, while doing security monitoring & incident response services, GTSC SOC (Security Operations Center) team discovered that a critical infrastructure was being attacked, specifically to their Microsoft Exchange application. During the investigation, GTSC Blue Team experts, determining that the attack utilized an unpublished Exchange security vulnerability, i.e., a 0-day vulnerability, immediately came up with a temporary containment plan. At the same time, Red Team experts started researching and debugging Exchange de-compiled code to find the vulnerability and exploit code. Thanks to experience finding the previous 1-day Exchange exploit, the RedTeam has a great understanding of Exchange's code flows and processing mechanisms, therefore research time was reduced, and the vulnerability was uncovered quickly. The vulnerability turns out to be so critical that it allows the attacker to do RCE on the compromised system. GTSC submitted the vulnerability to the Zero Day Initiative (ZDI) right away to work with Microsoft so that a patch could be prepared as soon as possible. ZDI verified and acknowledged 2 bugs, whose CVSS scores are 8.8 and 6.3.*

Since this was reported, GTSC said that they've encountered other customers also experiencing a similar trouble and after careful testing they confirmed that those systems were being attacked using the same new 0-day vulnerability. To help the Exchange Server community temporarily stop the attack until an official patch is available from Microsoft, they published their coverage of their findings while responsibly excluding the information needed to recreate the attack.

The exploits cause Exchange Server to download a malicious DLL whose code is then injected into the always running and always busy svchost.exe process. Once that's done, the DLL is started and it phones home to the machine at the IP address of 137.184.67.33. A simple and effective RC4 cipher is used with a key chosen at runtime to encrypt their communications.

The GTSC folks also explained:

> *While providing SOC service to a customer, GTSC Blueteam detected exploit requests in IIS logs with the same format as ProxyShell vulnerability:*
>
> *autodiscover/autodiscover.json?@evil.com/<Exchange-backend-endpoint>&Email=autodiscover/autodiscover.json%3f@evil.com.*
>
> *Checking other logs, we saw that the attacker can execute commands on the attacked system. The version number of these Exchange servers showed that they were already running with the latest update, so an exploitation using the actual Proxyshell vulnerability was impossible. Thus the Blueteam analysts confirmed that this was a new 0-day RCE vulnerability under active exploitation.*

There are indications of compromise (IoC's) published & available, and mitigation steps that can be taken. And last Thursday Microsoft publicly acknowledged the trouble with their own posting titled: *"Customer Guidance for Reported Zero-day Vulnerabilities in Microsoft Exchange Server"*

https://msrc-blog.microsoft.com/2022/09/29/customer-guidance-for-reported-zero-day-vulnerabilities-in-microsoft-exchange-server/

The trouble is, the mitigation which was first proposed by the GTSC people appears to be what Microsoft has copied and is echoing. It recommends adding a pattern-matching "Rewrite" rule to the IIS web server which hosts Exchange. But in an update from GTSC just yesterday, October 3rd, they noted: *"After receiving information from Jang (@testanull), we noticed that the regex used in the Rewrite Rule could be bypassed."* And they then link to a YouTube demo. I read that to mean that Microsoft's official proposed mitigation can be bypassed.

https://gteltsc.vn/blog/warning-new-attack-campaign-utilized-a-new-0day-rce-vulnerability-on-microsoft-exchange-server-12715.html

So, once again, things are not good for Exchange Server. Let's hope that Microsoft gets a permanent fix for this problem published soon, and that all Exchange Server users jump on getting their systems updated.

I'm (still) Not a Robot!

Last Wednesday, our friends at CloudFlare posted the formal announcement of the availability of some of the work they've been focused upon this past year. Their posting was: *"Announcing Turnstile, a user-friendly, privacy-preserving alternative to CAPTCHA"*

Through the years of this podcast we first introduced the abbreviation CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) and then we've followed the use and evolution of that idea as it has come up against the real world. So it's in keeping with that history that we introduce CloudFlare's new alternative. CloudFlare wrote:

> *Today, we're announcing the open beta of Turnstile, an invisible alternative to CAPTCHA. Anyone, anywhere on the Internet, who wants to replace CAPTCHA on their site will be able to call a simple API, without having to be a Cloudflare customer or sending traffic through the Cloudflare global network. Sign up here for free:* https://www.cloudflare.com/lp/turnstile/
>
> *There is no point in rehashing the fact that CAPTCHA provides a terrible user experience. It's been discussed in detail before on this blog, and countless times elsewhere. The creator of the CAPTCHA has even publicly lamented that he "unwittingly created a system that was frittering away, in ten-second increments, millions of hours of a most precious resource: human brain cycles." We hate it, you hate it, everyone hates it. Today we're giving everyone a better option.*
>
> *Turnstile is our smart CAPTCHA alternative. It automatically chooses from a rotating suite of non-intrusive browser challenges based on telemetry and client behavior exhibited during a session. We talked in an earlier post about how we've used our Managed Challenge system to reduce our use of CAPTCHA by 91%. Now anyone can take advantage of this same technology to stop using CAPTCHA on their own site.*

They then go on to explain that it's not only CAPTCHA's miserable user experience that is a problem...

> *While having to solve a CAPTCHA is a frustrating user experience, there is also a potential hidden tradeoff a website must make when using CAPTCHA. If you are a small site using CAPTCHA today, you essentially have one option: an 800 pound gorilla with 98% of the CAPTCHA market share. This tool is free to use, but in fact it has a privacy cost: you have to give your data to an ad sales company.*
>
> *According to security researchers, one of the signals that Google uses to decide if you are malicious is whether you have a Google cookie in your browser, and if you have this cookie, Google will give you a higher score. Google says they don't use this information for ad targeting, but at the end of the day, Google is an ad sales company. Meanwhile, at Cloudflare, we make money when customers choose us to protect their websites and make their services run better. It's a simple, direct relationship that perfectly aligns our incentives.*
>
> *In June, we announced an effort with Apple to use Private Access Tokens. Visitors using operating systems that support these tokens, including the upcoming versions of macOS or iOS, can now prove they're human without completing a CAPTCHA or giving up personal data.*
>
> *By collaborating with third parties like device manufacturers, who already have the data that would help us validate a device, we are able to abstract portions of the validation process, and confirm data without actually collecting, touching, or storing that data ourselves. Rather than interrogating a device directly, we ask the device vendor to do it for us.*
>
> *Private Access Tokens are built directly into Turnstile. While Turnstile has to look at some session data (like headers, user agent, and browser characteristics) to validate users without challenging them, Private Access Tokens allow us to minimize data collection by asking Apple to validate the device for us. In addition, Turnstile never looks for cookies (like a login cookie), or uses cookies to collect or store information of any kind. Cloudflare has a long track record of investing in user privacy, which we will continue with Turnstile.*

They then explain a bit more about what's under the hood...

> *To improve the Internet for everyone, we decided to open up the technology that powers our Managed Challenge to everyone in beta as a standalone product called Turnstile.*
>
> *Rather than try to unilaterally deprecate and replace CAPTCHA with a single alternative, we built a platform to test many alternatives and rotate new challenges in and out as they become more or less effective. With Turnstile, we adapt the actual challenge outcome to the individual visitor/browser. First we run a series of small non-interactive JavaScript challenges gathering more signals about the visitor/browser environment. Those challenges include proof-of-work, proof-of-space, probing for web APIs, and various other challenges for detecting browser-quirks and human behavior. As a result, we can fine-tune the difficulty of the challenge to the specific request.*

Turnstile also includes machine learning models that detect common features of end visitors who were able to pass a challenge before. The computational hardness of those initial challenges may vary by visitor, but is targeted to run fast.

You can take advantage of Turnstile and stop bothering your visitors with a CAPTCHA even without being on the Cloudflare network. While we make it as easy as possible to use our network, we don't want this to be a barrier to improving privacy and user experience. To switch from a CAPTCHA service, all you need to do is:

1. *Create a Cloudflare account, navigate to the `Turnstile` tab on the navigation bar, and get a sitekey and secret key.*
2. *Copy our JavaScript from the dashboard and paste over your old CAPTCHA JavaScript.*
3. *Update the server-side integration by replacing the old siteverify URL with ours.*

There is more detail on the process below, including options you can configure, but that's really it. We're excited about the simplicity of making a change.

And I'm excited to have a recently re-engineered and thoughtful replacement being offered from a major top-tier privacy-first organization such as CloudFlare. I can't think of a better source for this. Their page provides much more detail, and I've made it this week's GRC shortcut of the week: https://blog.cloudflare.com/turnstile-private-captcha-alternative/ https://grc.sc/891

Google TAG history

Google has put together a marvelously produced and engaging series of six, 15- to 19-minute videos under the banner "Hacking Google." It's a world class production, such as you might expect from a company with Google's resources. And, yes, of course, these are ultimately promotional, but that doesn't dissuade me from recommending them without reservation because they are gorgeous and they will be of tremendous interest to this podcast's listeners.

I first stumbled onto the second one in the series, not realizing that it was number 2 since it was numbered 001. Of course, they started numbering from zero (as I've always wished we had the foresight to do here.) In any event, I've only had the time so far before today's podcast to watch that second one all the way through. But based upon the one I watched, which was about the genesis and mission of Google's TAG team — their Threat Analysis Group — which is doing so much good, both for Google and for the Internet's users at large.

There's a first, short, trailer, which then leads us into the series. The six videos are:

EP000: Operation Aurora
EP001: Threat Analysis Group
EP002: Detection and Response
EP003: Red Team
EP004: Bug Hunters
EP005: Project Zero

The videos are collected into a YouTube playlist for easy access, and I have a link to it in today's show notes. If you use it, it will take you to the playlist's page where you can see the six-video collection. For some annoying reason, I was unable to use a GRC shortcut redirection to make access easier. But if you use Google to search on the expression "Hacking Google Playlist" a few links down from the top you'll find the same YouTube playlist page:

https://www.youtube.com/playlist?list=PL590L5WQmH8dsxxz7ooJAgmijwOz0lh2H

I was able to point to the first video of the series and they are chained, so if you want to jump right in, you can use: https://grc.sc/hackinggoogle ... which will start you off with episode EP000.

Closing The Loop

james

Hi Steve, As a 2019 Honda Accord owner, I've followed your detailed explanation of the key fob hacking saga with a vested interest. I don't leave anything of value in the car specifically because you made it clear anyone can get in, and even if they can't steal the car without the fob, they can take anything at any time.

Unfortunately, last week I was their next victim. Fortunately for me they didn't get what they were after, as I don't keep my lug nut lock keys in the car anymore since the rims are quite worthy of theft. They did rifle through EVERYTHING, and got a bag from the trunk that I liked, but I didn't even notice the $20 in the glove box under the empty lug nut lock key bag!

Although the care and fob were confused (my fob would unlock but not lock the doors), I was able to get the car to the dealer, and by the time I got there the fob had somehow synced so was functioning normally. I explained the experience and my full knowledge that this is a Honda-specific technical flaw that I know the local dealer can't fix, but could they at least wipe and reset my system so that the currently stolen code wouldn't work anymore for this set of thieves? I was told by the service department manager, Bill, that the reset costs $220 and I would have to pay for it. I declined.

John / @J0hnnt

Hi Steve, I've noticed there has been a lot of discussion around phishing protection on some of the most recent episodes of SN and how SQRL and FIDO etc will address to resolve it. One thing that you haven't mentioned recently though, is that just using a password manger such as Lastpass/Bitwarden etc will also provide a degree of protection as the autofill will fail as a fake URL will not match the one linked to the stored credentials. Long time SN listener and thanks form Brisbane. John

Eric Seidel / @TRUHaRDHouSeiNC

Hi Steve. Have been listening to episode 889 and wanted to mention I've also been using Security Now since I got my CISSP for CPEs to maintain the CERT. I've never had any issue either when submitting them with ISC. Thanks again for a great resource for that.

Manuel / @manuelzc

I have a question that I'm hoping you might answer: I recently installed backup software (EaseUS Todo Backup) and later realized that it's chinese software. Now I'm wondering if I just compromised my computer and my whole home network? Do you have any advice on this way of thinking? Should I reinstall my Windows, look for a way to reinstall the BIOS/UEFI, buy a new phone, a new TV, a new Router, a new everything? Where should I draw the line?

Cristian Sanchez / @csharpsanchez

Hey Steve, wondering if you have any thoughts on this WaPo article's assertion that public WiFi is safe: https://www.washingtonpost.com/technology/2022/09/26/public-wifi-privacy/

I admit I clicked on it expecting to laugh at thoughtless misinformation, but the discussion in the comments turned me around. Is the near ubiquity of HTTPS enough to declare public WiFi safe? What about a MitM hijacking DNS?

Long time listener and big fan of the show. Keep up the good work. Sincerely, Cristian

Washington Post:

"You probably don't need to worry about public WiFi anymore. Here's what a creep in a coffee shop could actually learn about you"

From uncovered webcams to reused passwords, it's tough to keep track of how much risk our everyday digital activities actually pose. For example, take WiFi networks in airports and coffee shops. They're part of life for anyone who travels or works remotely. They also have a reputation as cybersecurity risks. Do they still deserve it?

To see what potential hackers could see on a shared network, we invited professionals from cybersecurity company Avast to "compromise" my home network (all with my consent). We logged onto the same network at the same time, just like we would at a coffee shop, to see how much data a bad actor with a few free tools could learn about an unassuming WiFi user. What we found (or didn't find) might be a relief for the coffee shop crowd.

After a few minutes clicking around my finance, work, streaming and social media accounts, Avast's team could see the sites I'd visited (though not what I'd done there), the time of day and the specific device I used (in this case, a MacBook Pro). It's not nothing, but it wouldn't do hackers much good if they were looking to rip me off.

Chester Wisniewski, a principal research scientist at security company Sophos said that it's also relatively reckless for hackers to sit around messing with public networks: "That type of data isn't only low yield, it's high risk. If I can phish your password from my chair in Moldova and have zero risk of going to jail, why would I get on an airplane and go to your local Starbucks?"

In the internet's earlier days, the vast majority of web traffic was unencrypted — meaning anyone savvy enough to eavesdrop on a network could see everything you type into a website. [Our longtime listeners will all remember "Firesheep."]

By 2017, the balance had shifted with more than half of all web traffic using the encrypted "HTTPS" protocol, according to data pulled from Mozilla. Today, few legitimate sites remain unencrypted, with more than 90 percent of webpages loaded in the United States obscured from prying eyes, according to Mozilla's data. This means even if someone used a public network to spy on you, what they'd discover probably wouldn't be very valuable.

[And the article finishes...]

Focus your energies on cybersecurity chores within your control — such as setting strong passwords, saying "yes" to software updates and learning the signs of a scam — and don't sweat the public WiFi too hard. If a site, link or app seems sketchy, steer clear. [Which is always great advice.]

Sean Nelson / @seannelson

Steve, I have been looking for a product like this for years, and I finally found it. I think it might interest you, too. It's an SD card with encryption built in. This allows you to take pictures without risking that the contents will be viewed or confiscated by anyone else.

From what I can tell, it works by storing a master key set by the user. Then, each time the card powers on, it creates and stores a new symmetric session key, which gets processed through the master key for safekeeping. Any new files get encrypted using that new symmetric master key. When the device is powered off, the key is removed from volatile memory and the only persistent copy of the key is encrypted with the master key.

IF you take a picture and power off the camera, next time it powers on, no pictures are visible. You can take new pictures, but they, too, are unreadable after you turn off the camera.

However, when you attach the SD card to a computer and supply it with the master key, it is able to download the pictures along with each associated session key and unlock everything on the camera.

I have been looking for this for my dashcam for a long time. Given the state of the world, many others might be looking for something similar. --Sean

I think this is very cool and clever. And it's another example of the principle I often observe, which is that the generic and well-proven tools we already have can be combined in an endless number of ways. From what Sean described, here's how I would design this device, and it's likely what its maker, SwissBit, has done:

During setup on a PC a public key pair would be created and the SD card would be provided with the public key — and that's all.

Then, whenever the SD card is powered up, an internal high-entropy generator would synthesize a transient 256-bit symmetric key in RAM. That key would be immediately encrypted under the card's configured public key and only the encrypted key would be written to non-volatile store and retained. Then, during the card's use, all data flowing in and out of the SD card being read and written would pass through that AES-256 cipher, which would be transparent to the device it's plugged in to.

When the camera and/or SD card is powered down, the RAM-resident 256-bit symmetric key is forgotten and lost. Since it was encrypted under the user's public key, the only way for it to be decrypted would be with the use of the matching private key, which is deliberately unknown to the camera and the SD card.

So, it's a very slick solution for using an SD card to capture photos and videos while never allowing the contents of the card to be exposed.

https://www.swissbit.com/en/products/security-products/ishield-camera/

Jeff / @1Jeff1

Hey Steve... I noticed on Threema's blog that they just joined "Proton, Brave, the Tor Project, and a couple of other Internet services to launch the Privacy Pledge initiative." See: https://privacy-pledge.com/

The Privacy Pledge: 5 Principles for Building an Internet Where Privacy is the Default

We, the undersigned, commit ourselves to rebuilding the internet so that it returns to the ideals set out by its founders: a democratic platform designed to facilitate the free exchange of information, open communication, and privacy for the individual. In doing so, we believe it can serve the needs of people, not just corporations. This internet should be private by default and give each user a choice over who has access to — as well as control over — their personal data. An internet like this would be open and accessible to everyone, support democratic values, protect the fundamental right to privacy, and ensure free access to information.

This internet would support the growth of ethical business models, but it would first require that companies hold themselves to a higher standard of conduct that puts users first. By giving people control over their personal information, we can stop companies and governments from the spying, commodification, and attempted manipulation of users that have come to typify the internet today.

To build an internet where privacy is the default, we believe all organizations operating online should adhere to the following five principles:

1. The internet, above all, should be built to serve people. This means it honors fundamental human rights, is accessible to everyone, and enables the free flow of information. Businesses should operate in such a way that the needs of users are always the priority.

2. Organizations should only collect the data necessary for them to sustain their service and prevent abuse. They should receive people's consent to collect such data. People should likewise be able to easily find a clear explanation of what data will be collected, what will be done with it, where it will be stored, how long it will be stored for, and what they can do to have it deleted.

3. People's data should be securely encrypted in transit and at rest wherever possible to prevent mass surveillance and reduce the damage of hacks and data leaks.

4. Online organizations should be transparent about their identity and software. They should clearly state who makes up their leadership team, where they are headquartered, and what legal jurisdiction they fall under. Their software should be open source wherever practical and open to audits by the security community.

5. Web services should be interoperable insofar as interoperability does not require unnecessary data collection or undermine secure encryption. This prevents the creation of walled gardens and creates an open, competitive space that fosters innovation.

This is the internet that we deserve. This is the internet we are fighting for. It is within our reach, we simply need to be bold enough to seize it.

Brave / David Carroll / Mailfence / Mojeek / Neeva / Open-Xchange / OpenMedia / Proton / Threema / The Tor Project / Tutanota / You.com

Donn Edwards / @donnedwards

Hi Steve, surely the EU nonsense would go away if EU sites changed the analytics.google.com URL to analytics.google.eu? Then Google could use their EU servers to do the analytics in keeping with EU rules and laws.
Keep up the good work. /Donn

I didn't know whether "analytics.google.eu" worked, so I gave it a try. It returned a DNS lookup failure. So that doesn't appear to be an immediate option for those in the EU. What I assume Donn meant was that **if** Google wanted to respond to this concern this might be a clean means for them to do so. With nation after nation ruling that the use of Google Analytics, as it stands today, is unlawful, something likely needs to change.

Poisoning Akamai

Last Thursday, we got a glimpse into a world-shaking flaw that very few people knew about until now. A 23-year old Italian security researching enthusiast by the name of "Jacopo Tediosi" and his friend "Francesco Mariani" stumbled upon a flaw in Akamai's CDN that could have ruined many days for half of the world's websites that rely upon Akamai.

A content distribution network is, at its heart, a massive global distributed cache. It keeps track of a website's content, pretty much everything, and holds the most recent current version of a website's content in a cache which is local to that remote site's visitor. In that way, since the cache is close, the site's performance remains snappy even though its web server might be on the other side of the planet and on a not-particularly-fast connection.

First of all, imagining that this can be done, at scale, safely, is nuts. But that hasn't stopped the world's very successful CDN's from doing it anyway. What's absolutely obvious is that this only works if the massively distributed web asset cache maintains its coherence, so that it always accurately reflects the correct contents of a remote website. If it were possible, in some way, to deliberately alter a CDN's distributed cache to change any of those locally cached remote web assets, like a site's JavaScript, nothing less than havoc would rein, since such an attack would be tantamount to having the ability to directly alter a site's served content.

Since this podcast is titled "Poisoning Akamai" you already know that this was somehow done. But the reason this podcast is titled "Poisoning Akamai" is because the story told by 23-year old Jacopo of their discovery — and the discovery's aftermath — is absolutely worth sharing.

https://medium.com/@jacopotediosi/worldwide-server-side-cache-poisoning-on-all-akamai-edge-nodes-50k-bounty-earned-f97d80f3922b

Posting to medium.com, Jacopo opens his story by explaining:

> *In March 2022, my friend Francesco Mariani and I were teaming up on a private Bug Bounty program organized by Whitejar to search for bugs on a website that was using Akamai CDN. The Akamai WAF (Web Application Firewall) rules were bothering us while experimenting with the most common attack types, so we quickly got bored and started trying more esoteric payloads and mixing them. Finally, we ended up finding a vulnerability that really made us exclaim: "WOW, we 'broke' half the web!". But let's start from the beginning:*
>
> *At one point, we were intrigued by an unusual "DNS Failure" response, received by sending twice an HTTP/1.1 GET request to the host being tested with the "Connection: Content-Length" header and containing another GET request to www.example.com as body.*

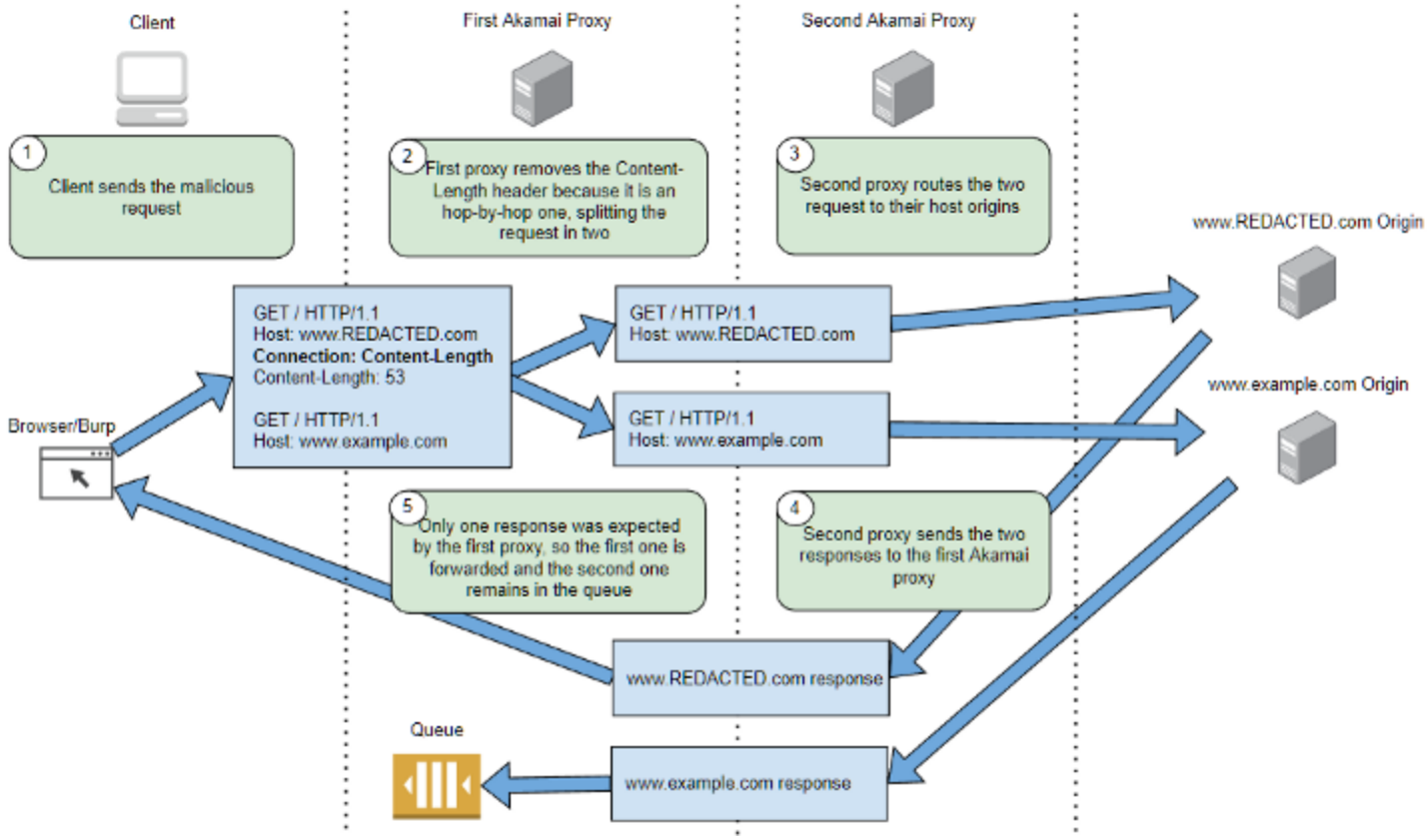
At this point I'll interject that I remember that we did an extensive podcast on exactly this subject, but I cannot recall what the specific topic was. It involved chaining HTTP requests or problems with HTTP chaining. What these guys are talking about is having an HTTP-style GET query (not a POST query) where the GET query contains another HTTP GET query following all of

its query headers. That's not the normal format for GET queries, where the specification of the object being queried is in the GET URL path. But, again, I know that we covered this years ago.

In this case, their example shows a GET query containing the query headers "Connection: Content-Length" and "Content-Length: 53". Upon seeing this odd "DNS Failure" response, Jacopo wrote: *"Weird behaviors like this can often be overlooked while testing so many things, but luckily this time we decided to dig deeper."* He said:

> *I have to admit, it took me a while to figure out what was going on, and I also had to reread Nathan Davison's excellent article on "hop-by-hop" headers that I had studied in the past.*

It is, indeed an excellent article which I would recommend to anyone who's interested in digging more deeply into this topic, though it's not necessary for understanding what's going on here: https://nathandavison.com/blog/abusing-http-hop-by-hop-request-headers



> *As explained in the RFC 2068 — Section 13.5.1, there are some special headers named "hop-by-hop", which are removed from proxies before forwarding requests to the next proxy or the destination.*
>
> *The addition of the "Connection" header allows including more "hop-by-hop" headers in addition to the default ones.*

Specifying the "Content-Length" header as "hop-by-hop", it happened that Akamai's first proxy removed it, turning the request body into a second request. Akamai's second proxy then resolved the two requests separately. Since the first proxy received two responses but only one was expected, a desynchronization occurred, and the second response was queued and subsequently sent in response to requests from other clients/users, causing an HTTP Smuggling Vulnerability.

Understanding this in detail requires a certain degree of knowledge about network architectures, web protocols, and other fancy stuff, so I try to explain it more easily with a chart.

I've included Jacopo's chart (above) since for those who are interested it very clearly shows what happens. It all boils down to an Akamai caching server query parsing error which results in a single query being split in two and forwarded to two separate destination client web servers. Then the two separate queries are answered, each by their own server and returned to the cache for caching. But the cache was only waiting for the reply to a single original query. So it places the unexpected reply in a queue, and that reply will be returned to someone else.

Jacopo says:

However, I could not immediately understand why the DNS error was showing up and why www.example.com was not being resolved. The answer was actually quite simple, but my co-worker's intuition was crucial: Akamai's proxy that routes requests appeared to resolve DNS only internally within Akamai's network.

We were using a VPN to verify that the desynchronization was an "open" one, meaning that it affected the responses given to IP addresses other than the ones we were attacking from. Also, believing it possible that the bug concerned all Akamai customers around the world, we changed our target from www.[REDACTED].com to more popular sites.

To our amazement, we noticed that it worked on them all and that, sometimes, "smuggled" responses were being server-side cached from Akamai Edge Nodes for the entire geographic area close to the IP sending the malicious request. This allowed us to semi-permanently (depending on cache times) create new arbitrary contents within almost any domain served by Akamai, resulting in a HUGE impact!

As a Proof of Concept, we created, for the whole Italian area, the newly cached page demo.paypal.com/jacopotediosi_hackerone.js, containing the content of www.sky.com/robots.txt (another Akamai customer, because we didn't own a host on the Akamai network to use for publishing our arbitrary contents).

Okay. So just to make this clear. They arranged to place a brand new page into Akamai's web cache named "/jacopotediosi_hackerone.js" associated with the root of demo.paypal.com. Thus anyone else in the region who was also being served by Akamai's cache would be able to retrieve that cached page by asking for it from the demo.paypal.com web server. Upon receiving that query, the local Akamai cache would see that it had a copy of that page in its local cache and return it (quite quickly) to whomever asked.

Once we understood the seriousness of the situation, we decided to report it ethically and responsibly, first of all to Akamai. Unfortunately, we quickly realized that Akamai doesn't have a Bug Bounty program, Hall of Fame, swag giveaways, or anything similar.

Da [redacted]@akamai.com>

Oggetto **Re: [security] Hop-by-hop vulnerability on Akamai** 25/03/2022, 20:29

A Me <jacopo.tediosi[redacted]>

Cc Akamai Security <security@akamai.com>

Hi Jacopo - It's quite clear what's happening here (thanks again for the very detailed report!)

We had no issues confirming your report. We are working on mitigations and making changes to our HTTP parsing & processing logic.

Unfortunately Akamai does not have a bug bounty program at this time and we are not able to provide a direct monetary award, but regarding some other means of recognition of your contributions I'll get back to you soon.

Thanks

We are white hats, but we were still not willing to work for free, because this vulnerability was very critical, and our skills are rare, complex, and sought after, and we think they deserve to be valued. So, while Akamai was patching following our report, we chose to race against the time by asking for bounties from single Akamai customers.

While this may sound strange, from our point of view on technologies, those who use a framework/plugin/CDN/whatever assume both their benefits and risks. Thanks to our work Akamai and all their customers have been made aware of a security issue and have been able to fix it, so it's just fair that they pay for our service because, without us, the vulnerability would still be there.

We used bbscope to extract links for all the public programs on the most popular bug bounties platforms. Next, we wrote a short bash script to filter from the list only the domains whose DNS pointed to Akamai:

In other words, they very cleverly reported this bug as it related to **specific** major Akamai customers whose websites were, indeed, still in serious danger until Akamai pushed out a fix. And what were their results? Jacopo wrote:

***Whitejar** immediately gave us €5,000 for their private program.*

*On **Bugcrowd**, they were not competent enough to understand the vulnerability and closed both our reports for Tesla.com as "duplicated" (of a ticket clearly not related to ours) and for LastPass.com as "not applicable" because they were unable to reproduce.*

***Intigriti**, regarding the Brussels Airlines program which we showed was vulnerable, told us that "Brussels Airlines is already aware of any request smuggle vulnerabilities in their web assets" (yeah, "ANY", lol), and closed our ticket as "duplicated".*

*On **HackerOne**, some programs refused our tickets and closed as "not applicable":*

- *Starbucks replied the vulnerability, in their opinion, wasn't a major security issue.*
- *PlayStation Staff failed to reproduce (even after we created a new page for them under the www.playstation.com domain).*
- *Marriott informed us that cache issues were temporarily out-of-scope for awards.*

However, many other programs paid us. We received:

- *$25,200 from PayPal*
- *$14,875 from Airbnb*
- *$4000 from Hyatt Hotels*
- *$750 from Valve (Steam)*
- *$450 from Zomato, and*
- *$100 from Goldman Sachs.*

In particular, Airbnb handled the situation outstandingly, applying custom rules on Akamai's WAF in less than 24 hours to block requests containing "Connection: Content-Length" even before Akamai's official fix.

PayPal was also a curious case, because they confirmed our report and issued a bounty long after Akamai's fix. So we don't know if they ever saw the vulnerability working or if they just trusted our PoC video.

Akamai fixed it by applying some rules that prevent specifying the "Content-Length" keyword within the "Connection" header value, but we are not sure that there are no bypasses or some other unexpected similar ways to split the requests.

Unfortunately, Microsoft and Apple acknowledged our reports after Akamai had already deployed a fix, but they thanked us anyway via private e-mails.

I think, from all of this, I'm most disappointed in Akamai. Their business is doing something that's so inherently dangerous, where a mistake, as we've just seen, could have truly horrific consequences, and they have no formal or informal bug bounty established. The idea that a company of that size could just say "Gee, thanks guys" and not write them a sizeable check on the spot is unconscionable. The way these guys arranged to monetize their discovery was quite clever. But that also means that they knew full well that for an exploit with this much power and virtually universal application, they could have asked the likes of Zerodium for a million dollars and they would have received it.

Anyway... I thought that was a very interesting story about something that happened earlier this year and because the guys were White Hat hacker, few people ever knew about it.