Security Now! #982 - 07-09-24
The Polyfill.io Attack

This week on Security Now!
What was Entrust's response to Google's decision to refuse trust of any of their TLS certificates signed after October 2024? How have the other CA's responded to this new opportunity? What's a Passkey Redaction Attack? And how worried should you be? And speaking of Passkeys, why not just have each website hold as many as we need? Wouldn't adding port knocking in front of the serious OpenSSH flaw we discussed last week prevent this problem? And if so, what's the larger lesson to be learned? And what about blocking an IP after some number of failed attempts? And finally, once again the Internet dodged a potentially devastating bullet.
What happened and what significant lesson should we take away?

Windows Patch Tuesday

Security News

Entrust Responds (both with a letter from the President and an interesting FAQ)

Thoughts on the Google Chrome Announcement… and
Our Commitment to the Public TLS Certificate Business
By Todd Wilkinson, President & CEO Entrust

Last week Google announced that they would no longer include Entrust root CA certificates in the Chrome Root Program. This means that the TLS certificates we issue after October 31, 2024, will no longer be trusted within the Chrome Root Store Program. We are disappointed by this decision and want to share how we intend to move forward.

We understand what led us here. We are committed to improvement. And Entrust continues to have operational capabilities to serve our customers' public and private digital certificate needs. These capabilities extend beyond the issuing roots in question.

Our recent mis-issuance incidents arose out of a misinterpretation we made of CA/Browser Forum compliance requirements. In our attempt to resolve this issue, our changes created additional non-security related mis-issuances.

In our attempt to provide additional flexibility to our customers, we provided extensions and delays in revocations that were not supported by the CA/Browser Forum Requirements, which mandate five-day revocation for all certificate mis-issuances.

This created an environment in which the community scrutinized past Entrust incidents. This identified past Entrust commitments, which if fully implemented, could have helped to prevent these incidents. We agree that there are opportunities for us to improve, and we have completed a thorough assessment of our CA operation in the last few months.

As a result of this assessment, we made changes in our organization, processes, and policies. For example, we have moved the CA product compliance team into our global compliance and operations teams to fully leverage the more robust capabilities of this larger organization. We have instituted a cross-functional change control board and a technical change review board to catch similar issues in the future. We are accelerating R&D for TLS certificate compliance and automation-related work while also improving the tracking of our public commitments and revising our public incident response practices to ensure such issues do not occur again.

I want to assure you that we are committed to continuing to serve as a public CA and that we will complete open issues and promised improvements in a timely manner. We are working with Chrome and the other browser root programs to address the raised concerns while also providing continuity for customers while we execute these changes. We have the expertise to do this, as demonstrated by our ability to deliver our many products and solutions designed to meet demanding global compliance requirements.

Entrust has been a publicly trusted CA for over two decades and has contributed to stronger Web PKI capabilities globally. We continue to have operational capabilities to serve customers' certificate needs today and will do so in the future. We respectfully ask for your patience as we work to ensure that you have no disruptions to the service you have come to expect from Entrust.

I would call that a pretty good and fair letter. He naturally failed to explain that the real concern was their history of repeated broken promises to repair, and that their excuses attempted to justify their actions by placing their reputation in the eye of their customers ahead of their duties as a CA. But I found their 4-question FAQ regarding this incident interesting as well. Here's what they asked themselves and answered:

Q: *What does it mean when Google says you've been "distrusted"?*

This means that public TLS certificates issued from nine Entrust roots after October 31, 2024, will no longer be trusted by Google Chrome by default. TLS certificates issued by those Entrust roots through October 31, 2024, will still be trusted by default through their expiration date.

Q: *Other CAs have said that after November 1 your TLS certificates won't be valid in the Google Chrome browser. Is that true?*

No, that's not true. All Entrust TLS certificates issued through October 31, 2024, will be trusted by default by Google through their expiration date. After October 31, we will have the operational capabilities to serve customers' certificate needs, with alternative or even partner roots if necessary. Our goal is to ensure that customers have no disruptions and continue to receive global support, certificate lifecycle management, and expert services to meet their digital certificate needs.

Q: *Does Google's decision mean Entrust is out of the digital certificate business?*

No. It means the public TLS certificates issued from nine Entrust roots after October 31, 2024, will no longer be trusted in Chrome browsers by default.

Google's decision has no impact on our private certificate offerings – including our PKI, PKI as a Service, and managed PKI – nor our code signing, digital signing, and email (S/MIME and VMC) offerings.

We have the operational capabilities to serve customers' certificate needs now, and in the future, with alternative or partner roots if necessary. Our goal is to ensure that customers have no disruptions and continue to receive global support, certificate lifecycle management, and expert services to meet their digital certificate needs.

Q: *Does this impact other Entrust solutions?*

This announcement has no impact on the broad range of Entrust identity-centric solutions.

So their most interesting response was #3, noting that even after Google begins to conditionally mistrust certificates signed by any of their root certs, nothing prevents them from riding on someone else's coattails. Nothing, that is, other than Google releasing another update to Chrome which then also blocks that. Remember that Entrust originally bootstrapped themselves into the commercial SSL/TLS browser certificate business by having Thawte sign their intermediate certificate, thus rising the coattails of the trust that Thawte had earned. This leaves two questions: Would another CA think it was worth their while to sign an intermediate certificate for

Entrust, knowing that then their own reputation would become entangled with Entrust's? And if this were done, what would Google do? Would they move to also block Entrust's intermediate certificates? And actually, they could probably do that preemptively in Chrome if they wished to. We'll see what Halloween brings this year.

Other major Certificate Authorities respond
No one will be surprised to learn that competing certificate authorities were not slow to report the news of Entrust's fall from grace and offer to provide their services in replacement. Sadly, not all CA's did this in a conscientious fashion. For example, the CA Sectigo misstated the future.

Remember that Sectigo is not one of this podcast's favorite CA's. In fact, they would likely be the last CA I would ever choose. Long time listeners may recall from whose loins "Sectigo" sprung... that would be none other than Comodo. Essentially, Comodo so badly damaged their own reputation due to a series of famous CA screw-ups, that they decided to rebrand themselves in the apparent hope that no one would notice.

In any event, Comodo/Sectigo gave their rapacious posting the title: *"Google to distrust Entrust SSL/TLS certificates: What this means for the industry"* And then they tried to tell us:

In a significant move to enhance digital certificate security, Google has announced its decision to distrust all public SSL certificates issued by Entrust, effective after October 31, 2024.

[Okay, they got it correct, there.]

This announcement has sent not just ripples, but waves through the industry, particularly among Entrust customers who now face the urgent task of transitioning to new Certificate Authorities (CAs).

[Okay, except there's nothing whatsoever urgent about it. As we are all well aware, all currently issued certificates remain valid throughout their current lifetime and everyone has July, August, September and October to chose another CA when it comes time to replace any Entrust SSL/TLS browser certificate.]

The catalyst for distrust

Google's decision is rooted in a series of compliance failures by Entrust. [And, again... look who's talking!] *Over the past several months, Entrust has experienced significant issues, including extremely delayed revocations and multiple lapses in meeting established security standards. Google's Security Blog noted, "Over the past six years, we have observed a pattern of compliance failures, unmet improvement commitments, and the absence of tangible, measurable progress in response to publicly disclosed incident reports." This lack of progress and ongoing issues justified the revocation of trust in Entrust's public roots.*

To be trusted by a browser, a CA must comply with specific requirements defined by the CA/Browser Forum. [I suppose they are an authority there.] Transparency is crucial, as CAs are expected to work in good faith with browsers to fix and prevent issues. Recent root program audits indicated a lack of confidence in Entrust's TLS certificate issuance practices, so this news wasn't completely unexpected to the industry, and prompted Google's decision to *distrust Entrust certificates in the Chrome browser.*

Implications for businesses

For businesses using Entrust certificates, this development necessitates immediate action. [No, it deliberately doesn't.] *Any website using an Entrust certificate issued starting November 1 will be treated as an unsecured site on Google Chrome, and likely other major browsers will follow suit. Companies must source a new certificate authority* ***before the deadline*** *to avoid their websites being flagged as untrusted.*

[Whoops. No. Companies must source a new certificate authority before their present Entrust certificate dies a natural death by having reached its expiration date.]

Choosing a reputable Certificate Authority

Considering Entrust's failings, businesses must reassess their relationships with CAs. A reputable CA should demonstrate robust compliance with industry standards, transparent operations, and a proven track record of security and reliability. Companies like Sectigo, which offers comprehensive certificate lifecycle management solutions, present viable alternatives.

These guys would NOT be my first choice. But we know who my first choice would be. That would be none other than DigiCert. So I went over to see what DigiCert had to say about this. Again, this end of Entrust trust does represent a significant marketing opportunity for every other CA in good standing.

Rather than going point-by-point through DigiCert's similar marketing piece, I'll just say that they did the honorable thing, as I was hoping they would. Under their topic:

What Does That Mean for Entrust Customers?

- *Public TLS certificates issued off of Entrust roots whose earliest SCT is after October 31, 2024 will no longer be valid on Google Chrome.*
- *Those Entrust certificates will be treated as an unsecured site.*
- *Any TLS certificate with a Signed Certificate Timestamp dated before October 31, 2024 will be valid for its term.*

With the slight exception that they meant before November 1st, DigiCert got it exactly right – "valid for its term."

And then they finished with *"We're Here to Help"* – writing: *"We understand this incident poses significant risk of business disruption to a large number of organizations. As the world leader in globally trusted PKI and TLS/SSL solutions, we are committed to making our services and solutions available to help you maintain critical operations and ensure uninterrupted business continuity during the transition from Entrust—and beyond."*

As it happens, GRC's DigiCert certificate expires at the end of this month and thanks to their excellent service, I already have its replacement standing by. I'll be switching all of GRC's servers over to using it shortly.

Passkey Redaction Attacks
A number of our listeners picked up on a weird story that was featured on the Dark Reading site. The story's headline was *"Passkey Redaction Attacks Subvert GitHub, Microsoft Authentication"*. Naturally that raised my eyebrows. The tagline beneath the headline was: *"Adversary-in-the-middle attacks can strip out the passkey option from login pages that users see, leaving targets with only authentication choices that force them to give up credentials."*

Okay. So first, it's what they're calling an Adversary in the Middle attack. So it appears that the politically correct "PC" police have decided that the use of the traditional and time-honored term "Man in the Middle" is no longer proper – as if "Man in the Middle" was actually a reference to some man somewhere and that as a consequence all men should be taking offense at this slur to our gender. Well, I **am** a man, and I'm not the least bit offended because I (and everyone else by the way) have always understood that the use of the term "man" in man-in-the-middle was always meant as an abstraction. So if any of our listeners were also rolling their eyes at the use of the new replacement term "Adversary in the Middle" attack, know that you were not alone.

Okay. That's out of my system for the time being and I've completely distracted from the article. Dark Reading refers to a piece from a site I've never encountered before, called eSentire. Everything about this is a bit odd, since the article is written by a guy named Joe Stewart and in the introduction, which he wrote, he refers to himself in the 3rd person. I'm only going to share the first bit in his introduction, where Joe writes:

> *In the past year, the uptake of passkeys has surged, with industry giants such as Apple, Microsoft and Google championing their adoption. Joe Stewart, Principal Security Researcher with eSentire's Threat Response Unit (TRU), has been reviewing many of the leading software providers' implementation of passkey technology and their current "authentication process."*
>
> *Regrettably, Stewart found that cybercriminals can still break into accounts protected by passkey technology on many online platforms, such as banking, e-commerce, social media, website domain name administration, software development platforms and cloud accounts, by using Adversary-in-the-Middle (AitM) phishing attacks.*
>
> *Stewart explains in this [I guess he should know] blog how passkeys are designed to work, how threat actors can currently circumvent passkey security, and the steps that online software providers and websites, that use or intend to use passkey technology, must take to ensure that their passkey implementation is secure from threat actors.*

I read through Joe's article looking for this new problem he had discovered. And... there isn't one. It all boils down to *"if someone goes to a phishing site that's pretending to be the authentic site, the miracle that is Passkeys won't protect you because the fake site won't offer Passkey authentication. Instead, it will offer any of the traditional authentication fullbacks that are still completely prone to interception by phishing."* It's unclear who would have ever believed otherwise, but Joe wants to be sure everyone is on the same page.

And, for what it's worth, we have a new term: the Passkey Redaction Attack which is a subset of phishing. Since the FIDO2 WebAuthn Passkeys technology is immune from phishing, the phishing site simply removes the Passkeys login option and gets ya the old fashioned way.

Closing The Loop

Ahmad Khayyat from Riyadh, Saudi Arabia

> *Hello Steve.. Instead of synchronizing passkeys, isn't it more secure to have a passkey per device, locked into that device's TPM (or equivalent facility)? Instead of backing up passkeys, have backup passkeys (on additional devices). Moreover, it's probably more feasible to convince sites to support multiple passkeys per user than to convince Google, Apple and Microsoft to support passkey portability.*

The big problem is that there's no way to know which sites support multiple passkeys and which don't. The Passkeys spec states that Passkey supporting sites should provide many-to- one Passkeys to account mappings. But as we know today, not all do. And running into a site that doesn't, means that the user cannot add another device to that site, which is a breakdown of the Passkeys promise. Hopefully this will eventually change. But it's also true that having Apple, Google and Microsoft performing their own cross-device synchronization of Passkeys, just as they've always done for passwords, takes the pressure off of sites to improve their Passkeys implementations. So, for the time being, the only practical solution is to either remain within one of the closed ecosystems provided by the big three, or use a 3rd-party password manager such as 1Password or Bitwarden (both sponsors of the TWiT network), which will provide the kind of cross-platform compatibility that Passkeys was intended to provide, but doesn't yet universally.

Max Feinleib / @MaxFeinleib

> *I'm guessing that nuevoMailer is pronounced "nwey-voh," as in the Spanish for "new."*

Now that he says it and I've heard it, I'm sure that's the case. NuevoMailer. Thanks, Max!

John / @PsyVeteran

> *Hey Steve, listening to you two discussing the OpenSSH bug in SN981. I note that a port knocking setup would completely mitigate this front door camping. Love listening to you guys and excited for 999 and beyond. On a side note, while I am all signed up for email, I can't see what address to send SN feedback on?*

Of course John's correct that port knocking would prevent exploitation of the OpenSSH flaw. And I suppose if one were really a belt and suspenders type, then adding that additional layer of pre-authentication authentication would make sense. But the problem is, OpenSSH is supposed to be super secure and the only protection anyone needs all by itself. Leo and I both authenticate with passwords and certificates. A certificate is like a 2048-bit secret key which cannot possibly be brute forced. Then a password is added to that, just to keep anyone who might obtain local access to the certificate from using it. So my point is, OpenSSH already provides so much security that anyone could be forgiven for **not** deploying an additional outer layer of protection.

But having said that, there is a larger more general lesson to be learned here. And that's the inherent problem with the security of any monoculture. Relying entirely upon security from any single source is never going to be as safe as using multiple different sources of security. Or stated another way, using a **"layered security solution"** is able to provide the strongest security by far. And if you're going to go to the trouble of layering, it's even better when the various layers have nothing in common with one another.

So John's observation that adding a "layer" of port knocking security is exactly that. And the advantage of port knocking is that it does not require a fixed IP for the "knocker." The whole point of knocking is that the secret knock identifies the IP of the knocker, whose traffic is then allowed to progress into the next layer of security.

I've often noted that I have all of the links between my various sites protected, not only with their own native security, whatever that may be and about which each of their vendors will boast at great length, but then also with point-to-point IP address filtering so that none of those links are even visible to anyone else out on the Internet. Their own protection may be strong, but why take an unnecessary risk? Of course, filtering by IP address is only feasible when the IP addresses of the endpoints are relatively static. But there really is no stronger security than adding a layer of simple IP address filtering to Internet traffic whenever and wherever possible. If anyone is listening to this has publicly accessible endpoints which have fixed IPs where IP-based filtering is NOT in place but could be... allow me to urge you to consider preventing anyone else from even being able to see those ports. It's worth it.

John also mentioned that he signed up for our email system but didn't see the inbound email address listed anywhere. That's mostly by design because I would prefer to only give Security Now! Listeners direct access to this mailbox. The address is shown on GRC's old /feedback.htm page, and since many others have mentioned this, I've changed the "reply to" address for the mailings to the Security Now! list to the correct address, which is "securitynow@grc.com". So everyone who has already received today's podcast email with this show's summary, picture of the week and full show notes, will see that it was sent from "securitynow@grc.com" so simply hitting reply will properly address a reply to me.

And speaking of heterogeneous layers of security...

Dr Brian of London 🇮🇱 / @brianoflondon

fail2ban : I can't believe any serious sysadmin isn't running this.
https://github.com/fail2ban/fail2ban
That completely negates this ssh vulnerability as far as I can tell.

After being installed into any Linux system, "fail2ban" creates a daemon running in the background. The project describes itself by writing:

Fail2Ban scans log files like /var/log/auth.log and bans IP addresses conducting too many failed login attempts. It does this by updating system firewall rules to reject new connections from those IP addresses, for a configurable amount of time. Fail2Ban comes out-of-the-box ready to read many standard log files, such as those for sshd and Apache, and is easily configured to read any log file of your choosing, for any error you wish.

This provides another layer of security based upon dynamic IP-address filtering, and it's another layer that any mature security solution will have. Why would any service allow any single remote IP address to be pounding away with repeated authentication failures? That's just nuts. But, sadly, it's still not built in. This sort of protection won't block a widely distributed attack, but the exact timing required to exploit this recent SSH flaw means that it would be very difficult to launch this attack through a widely distributed network of attackers.

The Polyfill.io Attack

We recently examined the consequences of the removal of a cloud-hosted service whose DNS CNAME record continued to point to the missing service. This then allowed ne'er-do-wells to establish their own replacement cloud service under the abandoned domain name as a means of hosting their malicious services within someone else's parent domain. That's not a good thing. But, bad as that is, it's not the worst thing that can happen. Arguably, the worst thing that could happen would be for a domain which is hosting a highly popular and often downloaded code library to fall into nefarious hands. And I would not be talking about this if it hadn't happened.

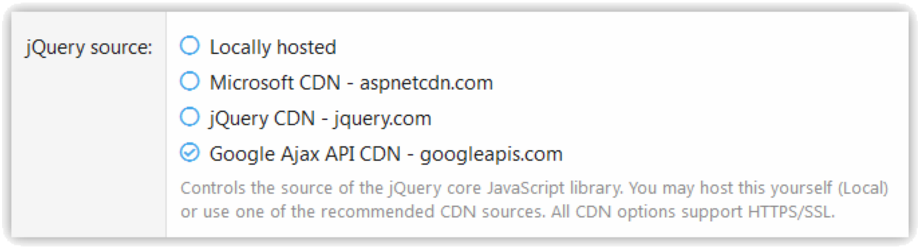
So first I'll back up a bit and explain what's going on here...

An unhealthy "design pattern" that's developed, is websites hosting pages which dynamically pull significant chunks of code from 3rd party sites. Back in the good old days, a website provided all of the pieces of the web page that it would present. Naturally, advertising ended that with ads being filled-in by 3rd party advertising servers. And though we've never had the occasion to talk about it directly before, the same thing has become commonplace with code libraries. A perfect example came to mind with GRC's own XenForo web forums.

XenForo uses a mix of technologies, with PHP being its primary implementation language on the server side. But the forums' browser-side polish, things like fading button highlights and pop-up floating dialogs, are provided courtesy of JavaScript running in the user's browser. And rather than reinventing the wheel, XenForo, like most websites, takes advantage of the wildly popular jQuery library. Wikipedia explains:

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animations, and Ajax. [Ajax being the browser's web page code making queries to other resources.] It is free, open-source software using the permissive MIT License. As of August 2022, jQuery is used by 77% of the 10 million most popular websites. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin, having at least three to four times more usage than any other JavaScript library.

jQuery has become so widely and universally used and standardized that you could almost think of it as a browser extension since so much code takes advantage of it. If you're a developer of web pages that depend upon jQuery, you'd like to obtain the benefits of any bugs that are fixed, speed improvements or other optimizations as soon as they become available. So rather than hosting the jQuery library yourself, you instead ask the browser itself to download the latest version of jQuery from the Internet. The question is, where should it be obtained? I've included a screenshot of one of the configuration options for the XenForo system:



jQuery source:
- Locally hosted
- Microsoft CDN - aspnetcdn.com
- jQuery CDN - jquery.com
- [✓ Google Ajax API CDN - googleapis.com]

Controls the source of the jQuery core JavaScript library. You may host this yourself (Local) or use one of the recommended CDN sources. All CDN options support HTTPS/SSL.

The option labeled “jQuery source” has four radio buttons, so we get to pick one from among the four. The choices are: Locally hosted, Microsoft CDN at aspnetcdn.com, jQuery CDN at jquery.com, or Google Ajax API CDN at googleapis.com. And the default from the XenForo folks was that final choice, to obtain the library on-the-fly every time it’s needed directly from Google.

We might wonder why Microsoft and Google are interested in being benefactors by providing access to code libraries on their content delivery networks. But in a world where 3rd party cookies are able to track their users, every time any web browser anywhere fetches a copy of the jQuery code library from Microsoft or Google, that browser’s cookie for that CDN is sent along with the site the user’s browser is visiting and the user’s current IP address. So Microsoft or Google will obtain some additional information to add to their pile about every user’s whereabouts. In return, our browsers probably don’t need to wait very long to receive that library since both of those CDN’s are high performance.

Okay, so the main point here is that a model has gradually evolved over time where websites we visit are no longer providing all of the content that runs their pages, and that in addition to ads, and gravatars, and web fonts, and a bunch of other stuff, today’s web pages also obtain significant code libraries from 3rd-party servers.

At the beginning of our discussion of this I noted that *“Arguably, the worst thing that could happen would be for a domain which is hosting a highly popular and often downloaded code library to fall into nefarious hands. And I would not be talking about this if it hadn’t happened.”*

Armed with this bit of background, we can easily understand the consequences of this and of how serious it could be. So what happened?

First we need to answer the question: **What is a polyfill?**

There are times when a device or a library may not be the latest and greatest, like when someone is still using an older version of Internet Explorer which may not support the latest HTML standards. What can be done in such cases is that a so-called “polyfill” library can be used.

A website that wishes to be able to run on the widest range of browsers, while still taking advantage of the latest features of the most recent browsers, can choose to have its web pages download and invoke a polyfill library. Then, when the page’s code runs, the polyfill library will check to see whether the underlying web browser supports the various features that the site wishes to use... and if the answer is “no”, the polyfill library itself will make up the difference. It will “fill-in” for the down-version browser by emulating the functionality that’s natively missing from the browser. So, in short, polyfilling is another popular practice and library. It’s mostly used to fill in for missing JavaScript features, but both PHP and Python have available polyfills.

So just as with the wildly popular jQuery library, polyfill libraries, while not as wildly popular as jQuery, are still in wide use. That makes it, lets just say very bad, when a Chinese company purchases the polyfill.io domain and its associated Github account and then proceeds to do bad things with them. Essentially, this company purchased a position of extreme power and responsibility and then chose to misbehave.

Exactly two weeks ago, on June 25th, the forensics Team at the security site “Sansec” broke the news with their headline *“Polyfill supply chain attack hits more than 100,000 sites”*. And before I go on, I’ll just note that, as we’ll see, their initial estimation of more than 100,000 sites fell short by around 280,000 short. The real number has turned out to be more than 380,000 sites.

Polyfill.js is a popular open source library to support older browsers. 100K+ sites embed it using the cdn.polyfill.io domain. Notable users are JSTOR, Intuit and World Economic Forum. However, in February this year, a Chinese company bought the domain and the Github account. Since their purchase, this domain was caught injecting malware onto mobile devices via any site that embeds cdn.polyfill.io. Any complaints were quickly removed from the Github repository.

The polyfill code is dynamically provided based on the HTTP query headers, so multiple attack vectors are likely. Sansec decoded one particular malware which redirects mobile users to a sports betting site using a fake Google analytics domain (www.googie-anaiytics.com). The code has specific protection against reverse engineering, and only activates on specific mobile devices at specific hours. It also does not activate when it detects an admin user. It also delays execution when a web analytics service is found, presumably to not end up in the stats.

The original polyfill author recommends that Polyfill should no longer be used at all, as it is no longer needed by modern browsers. Meanwhile, both Fastly and Cloudflare have put up trustworthy alternatives, if you still need it. This incident is a typical example of a supply chain attack.

They followed up this report with a series of four updates one per day.

On June 25th:

Google has already started blocking Google Ads for eCommerce sites that use polyfill.io.

On June 26th:

Someone launched DDoS attacks against our infrastructure and BleepingComputer (who was the first to cover our research).

On June 27th:

Cloudflare has implemented real-time rewrites of cdn.polyfill.io to their own version. A little later, Namecheap has put the domain on hold altogether, which eliminates the risk for now. However, you are still recommended to remove any polyfill.io references from your code.

On June 28th:

We are flagging more domains that have been used by the same actor to spread malware since at least June 2023: bootcdn.net, bootcss.com, staticfile.net, staticfile.org, unionadjs.com, xhsbpza.com, union.macoms.la, newcrbpc.com.

And then Censys weighed in with some terrific reporting based upon their extensive visibility into the Internet. I should note that my server logs are full of port probes from Censys. They want to know what's going on and who's running what. They are another site like Shodan, but the nice thing, for me at least, is that their reverse DNS resolves to their domain name, so at least, if I'm curious, I can see who's knocking at the door. I still don't let them in, but it's marginally less creepy to know that it's not an attacker.

Anyway, last Friday, under the headline *"Polyfill.io Supply Chain Attack – Digging into the Web of Compromised Domains"*, Censys wrote:

> *On June 25, 2024, the Sansec forensics team published a report revealing a supply chain attack targeting the widely-used Polyfill.io JavaScript library. The attack originated in February 2024 when Funnull, a Chinese company, acquired the previously legitimate Polyfill.io domain and GitHub account. Shortly thereafter, the service began redirecting users to malicious sites and deploying sophisticated malware with advanced evasion techniques.*
>
> *On June 27, 2024, Namecheap suspended the malicious polyfill.io domain, mitigating the immediate threat for now. However, Censys still detects 384,773 hosts embedding a polyfill JS script linking to the malicious domain as of July 2,2024.*
>
> *These hosts include websites associated with major platforms like Hulu, Mercedes-Benz, and WarnerBros. Security experts strongly advise website owners to remove any references to polyfill.io and its associated domains from their codebase as a precautionary measure. Cloudflare and Fastly have offered alternative, secure endpoints for polyfill services as a workaround.*
>
> *Further investigation has uncovered a more extensive network of potentially compromised domains. Researchers identified four additional active domains linked to the same account that owned the polyfill.io domain. Censys detected 1,637,160 hosts referencing one or more of these endpoints. At least one of these domains has been observed engaging in malicious activities dating back to June 2023, but the nature of the other associated domains is currently unknown.*
>
> *This incident highlights the growing threat of supply chain attacks on open-source projects.*

1,637,160 website hosting servers are emitting web pages that are causing their visitor's web browsers to download resources from these malicious domains. Speaking of the hosts that they have found attempting to pull from polyfill.io, Censys added...

> *The presence of domains like "www.feedthefuture.gov" in these top results also highlights the use of polyfill.io across various sectors, including government websites. In total, Censys observed 182 affected hosts displaying a ".gov" domain.*
>
> *While estimates of the scale of affected websites vary widely between sources (Sansec reported 100,000, while Cloudflare suggested "tens of millions"), it's clear that this supply chain attack has had a widespread impact.*
>
> *Further investigation has uncovered an extensive network of potentially related domains. A twitter user discovered that the maintainers of the polyfill GitHub repo leaked their CloudFlare*

API secrets within the repo. It was thereafter discovered that the leaked Cloudflare API key is still active, and shows 4 additional active domains linked to the same account:

Bootcdn[.]net bootcss[.]com staticfile[.]net staticfile[.]org

Bootcss[.]com, has been observed engaging in malicious activities that are very similar to the polyfill[.]io attack, with evidence dating back to June 2023.

*The two main malicious domain names involved were both registered at the end of May. The evil jQuery was introduced by **highlight.js** hosted on **cdn.bootcss.com**. When the request for **highlight.js** has a specific Referer and mobile User-Agent, the server will return **highlight.js** with malicious code, otherwise it will return normal code, which is highly disguised. Moreover, there are specific conditions for whether malicious jQuery is introduced when this code is executed."*

In other words, matching on a specific referrer enables the code to only target users visiting specific websites and matching on the mobile device's User-Agent causes it to only target users using specific device types.

So this is all really quite bad. The one person whose reactions we haven't yet heard from is Andrew Betts, the originator of polyfill. Not polyfill.io, just polyfill. Way back on February 25th, after the change of ownership of polyfill.io and before all this came to a head, Andrew Tweeted:

If your website uses http://polyfill.io, remove it IMMEDIATELY. [The caps were his.] I created the polyfill service project but I have never owned the domain name and I have had no influence over its sale.

No website today requires any of the polyfills in the http://polyfill.io library. Most features added to the web platform are quickly adopted by all major browsers, with some exceptions that generally can't be polyfilled anyway, like Web Serial and Web Bluetooth.

Domains that serve popular 3rd party scripts are a huge security concern. The team at http://google-analytics.com for example could read or modify almost any website in the world - like http://gov.uk, or http://wellsfargo.com.

If you own a website, loading a script implies an incredible relationship of trust with that third party. Do you actually trust them?

I should explain that Andrew never endorsed the distribution of his polyfill library via a 3rd party. He intended to have the hosting site that needed to use the library to host it themselves. He's clearly aware of the havoc that would ensue if a major 3rd party – like Google with their ubiquitous analytics code being injected into every website around – were to go rogue or be compromised by a sophisticated attack. But somewhere along the way, someone – not Andrew – had the bright idea of hosting polyfill at polyfill.io and over time everyone started to use it rather than host it themselves. And this was the crucial mistake that so many in the industry made.

Think about it:

You've purchased TLS certificates to protect your site. You have electronic and physical security, maybe even hardware security modules to protect your secrets. You've implemented Passkeys to offer the latest in state-of-the-art public key login authentication. You've gone through the popular checklists of things you need to do to be secure and everything is checked off. You've done everything you can think of. Then you have every one of your precious and secure site's web pages downloading and running active JavasScript code that runs with full permissions in a first-party context from an unknown entity in China that turns out to be malicious.

What's wrong with this picture? What's ***RIGHT*** with this picture?

So what happened in this case? Without some deeper investigation it's impossible to say exactly. I took a look back in time thanks to the Internet Archive's Wayback machine. The Wayback machine began taking snapshots of the polyfill.io site back in 2013, 11 years ago. And it certainly all looked above board and solid for years. It was a service being brought to the world by the Financial Times with bandwidth and CDN services provided by Fastly and the public domain polyfill code present on Github.

An early version of the site's homepage said:

Just the polyfills you need for your site, tailored to each browser. Copy the code to unleash the magic:

<script src="https://cdn.polyfill.io/v2/polyfill.min.js"></script>

Polyfill.io reads the User-Agent header of each request and returns polyfills that are suitable for the requesting browser. Tailor the response based on the features you're using in your app, and see our live examples to get started quickly.

The polyfill service is developed and maintained by a community of contributors led by a team at the Financial Times. It evolved from a previous service developed by Jonathan Neal, and our cdn.polyfill.io domain routes traffic through Fastly, which makes it available with global high availability and superb performance no matter where your users are.

But things changed and it's interesting to trace the site's evolution through the years. The Wayback machine makes that easy. What I discovered was that something happened toward the end of the year, last year, on November 1st of 2023. Last year on October 31st, the site was offering polyfills while boasting that in just the last 30 days it had fulfilled 61.2 BILLION requests while delivering 292.98 TRILLION bytes worth of polyfills... in just the past 30 days. And Fastly's name was still proudly highlighted in red against a black background.

But then, the very next day on November 1st, the site's pages changed. Gone was the activity tracking, the reference to Github or any mention of Fastly. So as I said, it would take interviewing the parties involved to learn more. One guy who would likely know the whole story is Jake Champion whose Twitter handle is @JakeChampion. His personal bio at JakeChampion.name mentions his involvement with the polyfill project and his copyright can be found at the bottom of the earlier pages. But he has his Twitter account locked for viewing only by approved followers.

What's clear is that maintaining the 100% free polyfill service was doubtless becoming expensive and it was the definition of thankless. Eleven years earlier it would have been much easier than today. And with nearly 300 terabytes of polyfills being served every 30 days, that's 10 terabytes per day, day in and day out.

So it's not difficult to imagine someone coming along and offering to purchase that massive burden. Or maybe just take it off their hands. But however it happened, in the blink of an eye, 384,773 websites were suddenly inviting an unknown stranger's JavaScript code into and onto their previously super-secure and secured pages.

While digging around for all of this interesting information and backstory I ran across one other chilling bit. This was posted over on Medium by Amy Blankenship who describes herself as a full stack developer at a financial technology company. She primarily writes about React, Javascript, Typescript, and testing. And under the headline *"Those Scary Stories About Polyfill.io? They're Just the Beginning"* she posted the following to Medium:

The Internet exploded this week about how polyfill.io was injecting malicious code into websites that were linking to it from the CDN at the URL where it has been served for years.

The thing is, ownership of the Github repo and the download domain were transferred in February. They didn't wait until this week to start making changes. Here's how I know.

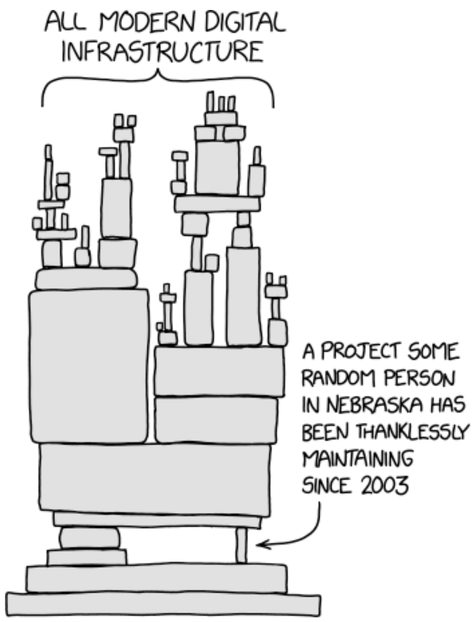
In February, we started to get mysterious errors when some of our users tried to log in. The stack trace the Sentry error boundary was giving us didn't make any sense. ***It was deep within the okta-react library code we were using.*** *But we had not recently upgraded our okta-react, okta-js, or sentry code — or the code that called it.*

Long story short, in the course of stepping through the code in the debugger, I found that some code in Okta was expecting to receive an object that had iterable properties. Sometimes when it received the object, those properties could not be iterated. Digging further, I found that the object was returned from code ***deep within the polyfill library.***

In other words, Okta, the major identity, cloud security and access management company had some deeply embedded dependency upon the polyfill.io library, which it was pulling from polyfill.io. Wikipedia reminds us:

Okta, Inc.is an American identity and access management company based in San Francisco. It provides cloud software that helps companies manage and secure user authentication into applications, and for developers to build identity controls into applications, website, web services, and devices. It was founded in 2009 and had its initial public offering in 2017, reaching a valuation of over $6 billion.

That's right. And its core Okta-react and Okta-JavaScript library code was pulling from a library that was now being supplied by a clearly hostile and malicious company named FunNull, based in China. We know that because shortly after FunNull acquired the polyfill.io domain and github account, the Okta library began mysteriously acting up.


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

And this is, of course, where we're reminded of Randal Munroe's wonderful KCD cartoon showing a large and ungainly collection of stacked blocks reaching up to the heavens. We would call it a "house of cards" if it were composed from playing cards rather than blocks.

That detailed and stacked assortment is then labeled "All modern digital infrastructure".

And then down near the very bottom off to one side is a crucial twig whose presence is holding up the entire assembly such that without it everything would come tumbling down. And that little twig is labeled: "A project some random person in Nebraska has been thanklessly maintaining since 2003."

https://www.xkcd.com/2347/ *(Randall Munroe titled his wonderful cartoon "Dependency")*

Unfortunately, in this case, as the result of a number of unfortunate events, it would be labeled: "Control acquired by a hostile and malicious Chinese company."

Leo and I live in California. This state has a major, seismically active, geological fault running through it known as the San Andreas Fault. The San Andreas Fault runs North-South about 750 miles through California, forming part of the tectonic boundary between the Pacific Plate and the North American Plate. This is the same fault that Lex Luthor was determined to trigger and use to turn California's Eastern neighboring State of Nevada into beach front real estate by sinking all of California into its Pacific ocean. Fortunately... well... we have Superman... so that hasn't happened yet.

The relevance of this is that the somewhat precarious state of our Internet's security infrastructure puts me in mind of the nature of seismic faults, and the earthquakes that accompany them. As tectonic plates slowly shift, pressure builds up, and that's not a good thing. So here in California what we're looking for – and hoping for – is a more or less continuous series of small earthquake tremors where no one is hurt and the china plates remain on their shelves. People text each other asking "Hey! Did you just feel that?" That's much better than a long period of quiet, which allows massive pressures to build up, only to be released in a single massive event. So what we want here in California is lots of small, barely noticeable events.

How many times on this podcast, especially in the past five years, where it seems to be happening more frequently, have I noted that we all just dodged a bullet? That the Internet just experienced another tremor? We discovered a serious problem that did not take everything down with it. So here again, the industry just received another wake up call with no known actual, let alone horrific, damage being done, and an important lesson was hopefully learned that was not expensive. Let's all hope that things remain like this, with a continuing series of small and harmless earthquakes, rather than, as it's referred to in California... "the big one" hitting.