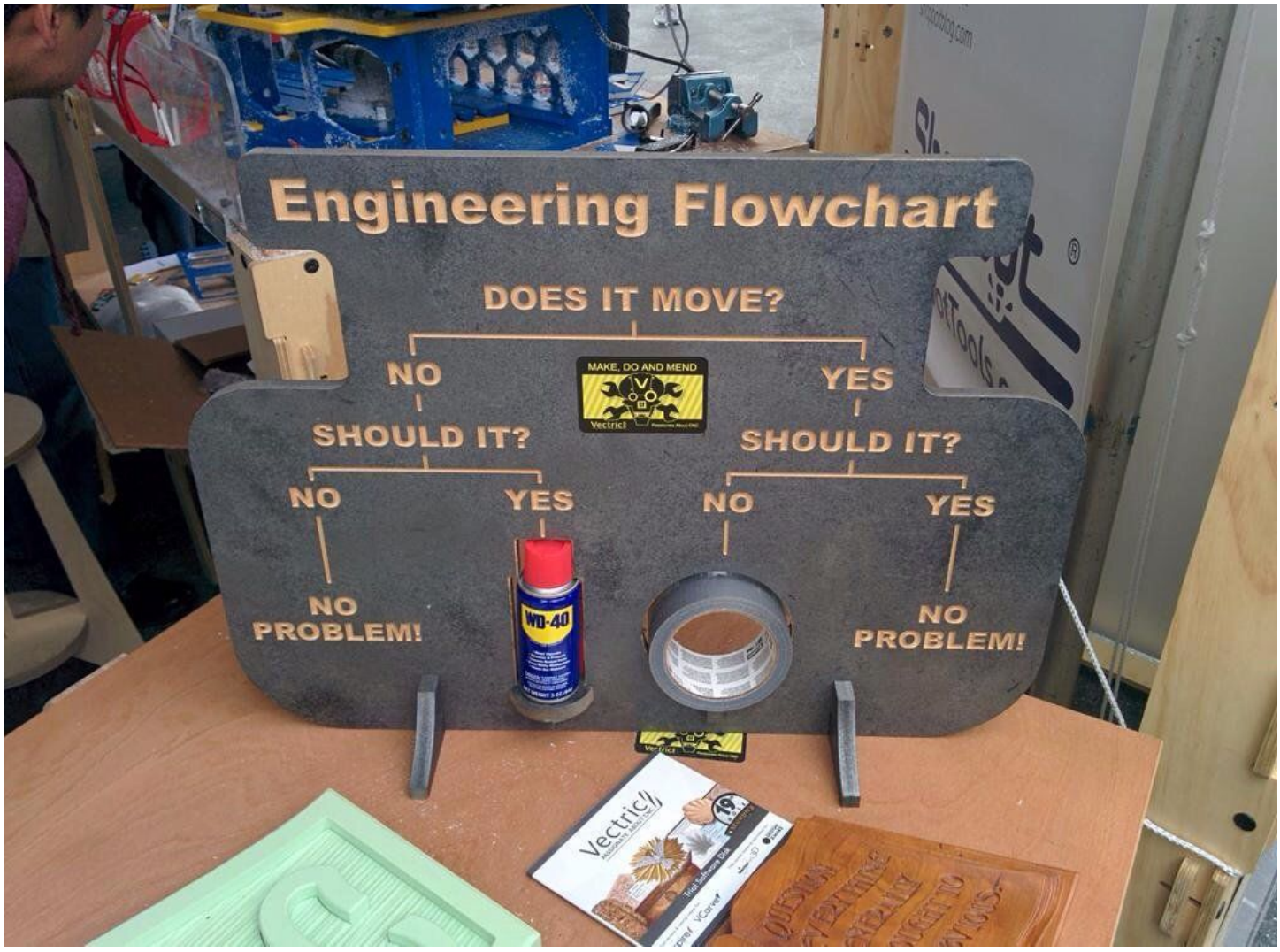
Security Now! #713 - 05-07-19

Post-Coinhive Cryptojacking

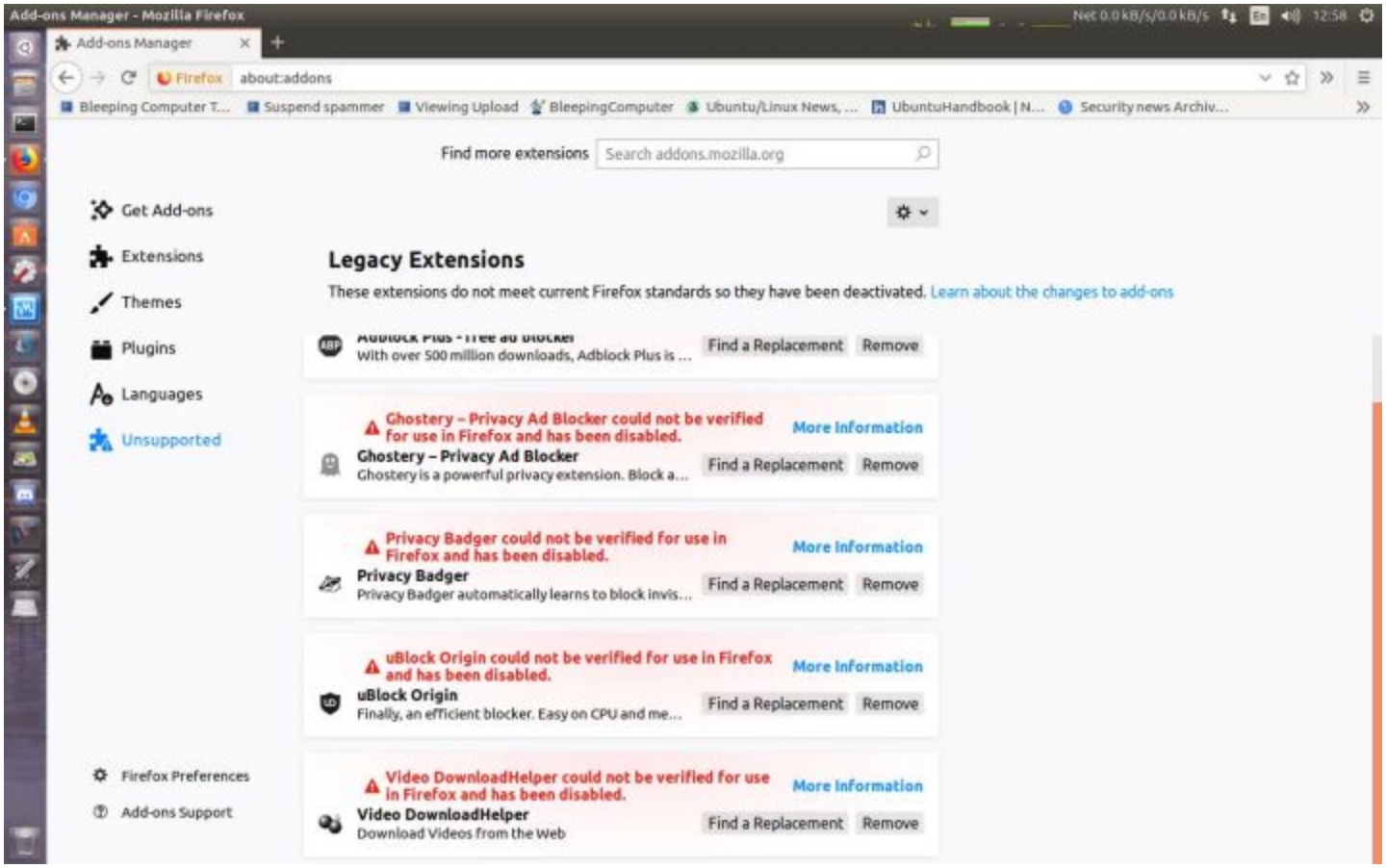
This week on Security Now!

This week we look at the mess arising from Mozilla's intermediate certificate expiration (the most tweeted event in my feed in a LONG time!), Google's announcement of self-expiring data retention, another wrinkle in the exploit marketplace, Mozilla's announcement about deliberate code obfuscation, a hacker who hacked at least 29 other botnet hackers, a warning about a very popular D-Link netcam, who's paying and who's receiving bug bounties by country, another User-Agent gotcha with Google Docs, a problem with Google Earth on the new Chromium-Edge browser, and a bit more about Edge's future just dropped at the start of Microsoft's Build 2019 conference. Then we take a look at the continuing and changing world of cryptojacking after Coinhive closed their doors last month.



Security News

Mozilla allowed an intermediate certificate to expire...
... and all browser add-ons signed by that then-broken chain refused to load!
https://bugzilla.mozilla.org/show_bug.cgi?id=1548973



(image from: https://www.bleepingcomputer.com/news/software/firefox-addons-being-disabled-due-to-an-expired-certificate/

This was different from the problem I ran into two weeks ago when I was forced to replace the code-signing certificate I'd been using in good standing for three years. My trouble was that the replacement certificate was unknown to Windows and 3rd-party AV.

As we discussed many years ago when first explaining the details of certificate-based security, certificates form a chain a trust, with a so-called "trust anchor" which is typically a self-signed certificate that's implicitly trusted due to its residence in the system's root certificate store.

It's possible for the "working certificate" at the end of the chain to be directly signed by the root certificate, but it's not typical or practical. We don't want our root certificates to be expiring often, since, in some settings, they can be difficult and tricky to securely replace on-the-fly. And any signing by the root requires the use of the root certificate's key, which is the most highly prized possession of any Certificate Authority. So prudence dictates that it not be taken out of safe offline storage very often. Instead, an intermediate certificate is created to be the signer of the end-point certificates, and it has an intermediate expiration date as well. That date must be later than any certificates it signs, but usually sooner than the root certificate that signs it... since it's only valid if its signer is still valid.

For example, GRC's current TLS EV certificate has a "not valid after" date of March 27th, 2020. It was signed by DigiCert's Intermediate "SHA2 Extended Validation Server CA" certificate whose "not valid after" date is October 22nd, 2028. And THAT intermediate certificate was signed by DigiCert's High Assurance EV Root CA with a "not valid after" date of November 10th, 2031.

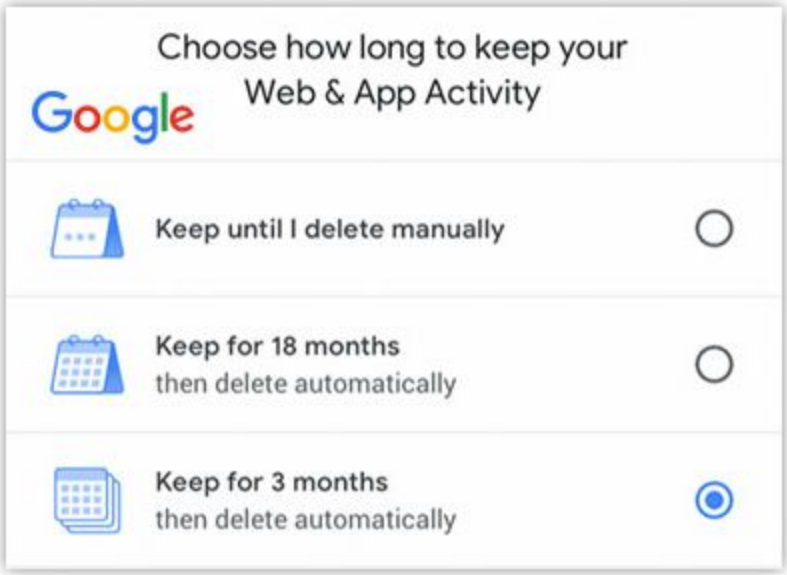
What happened over this past weekend was that in the months leading up to Saturday, no one at Mozilla had noticed that the intermediate certificate which anchored a bunch of popular Firefox browser add-ons that were still in current use (including uBlock Origin)... would be expiring last Saturday.

Unless it's running in Canary or Developer mode where add-on signing is overridden, Firefox protects is users from malicious add-on impersonation by validating the signing of all add-ons every time they are loaded by the browser. And that validation means that every certificate chaining back up to the root certificate must be valid... which, of course, stopped being true for a bunch of add-ons.

A bunch of "what to do" advice flew back and forth over the weekend with the assumption that the Mozilla folks would get this sorted out quickly. One thing any user could do if they were desperate would be to "back date" their system's clock by one day. I didn't have the chance to experiment, but it would make sense for add-on chains of trust to only be checked at load time, and not after. So that the date hack could be performed briefly when launching the browser, then put back to the proper date so that the dates of other stuff isn't messed up. (And you can tuck that hack away in the back of your mind if or when something similar happens in the future. :)

Anyway, with v66.0.4 this was fixed and all is well again.

Google: "Introducing auto-delete controls for your Location History and activity data"
https://www.blog.google/technology/safety-security/automatically-delete-data/



<quote> "Whether you're looking for the latest news or the quickest driving route, we aim to make our products helpful for everyone. And when you turn on settings like Location History or Web & App Activity, the data can make Google products more useful for you—like recommending a restaurant that you might enjoy, or helping you pick up where you left off on a previous search. We work to keep your data private and secure, and we've heard your feedback that we need to provide simpler ways for you to manage or delete it.

You can already use your Google Account to access simple on/off controls for Location History and Web & App Activity, and if you choose—to delete all or part of that data manually. In addition to these options, we're announcing auto-delete controls that make it even easier to manage your data. Here's how they'll work...

Choose a time limit for how long you want your activity data to be saved—3 or 18 months—and any data older than that will be automatically deleted from your account on an ongoing basis. These controls are coming first to Location History and Web & App Activity and will roll out in the coming weeks.

You should always be able to manage your data in a way that works best for you--and we're committed to giving you the best controls to make that happen."

I think that setting a 3-month auto-storage-expiration policy is probably a nice compromise for users of Google services who still desire the promise of location-based enhancement of their online experience without the creep-factor of everywhere they have ever been for the past ten+ years being silently logged, retained and searchable in a "SensorVault" database somewhere.

For users who are are sure they want NO retention right now, it can be "Paused" (as Google insists upon phrasing it) and then scrubbed through a manual process:

While signed into Google, you click on your profile picture, then on the Google account button.

In the left hand column select "Data & personalization" and in there you'll find "Web & App activity" and "Location History." Select each in turn, flip the toggle to "Pause" and confirm that's really what you intend.

The confirmation dialog for "Web & App activity" explains: "Pausing this setting doesn't delete any of your past data. You can see or delete your data and more at myactivity.google.com."

And the confirmation dialog for "Location History" similarly explains: "Pausing this setting doesn't delete any of your past data. You can see or delete your data and more at maps.google.com/timeline."

You know… I'm really not much of an uber privacy nut. And many of our listeners were unhappy recently to hear my capitulate on that front. But I have to say that going over to "myactivity.google.com" and scrolling back through days of stuff I've done -- in private, with no one obviously looking over my shoulder -- and then knowing that all that's being archived somewhere for god knows what purpose... it IS a bit creepy.

An apparently Ukrainian, talented and prolific hacker has been selling 0-day exploits to various Advanced Persistent Threat groups...

Kaspersky Lab has been watching the network comings and goings of a prolific hacker whose name is believed to be "Volodimir" - a Ukrainian name - but who uses "Volodya" as a nickname which often appears in the hacker's code.

For the past three years this hacker has been selling Windows 0-days to at least three different cyber-espionage groups, as well as cyber-crime gangs. The hacker's activities reinforce beliefs that some government-backed cyber-espionage groups are regularly purchasing 0-day exploits from third-parties in addition to developing their own, in-house.

APT groups believed to be operating out of Russia and the Middle East have often been spotted using zero-days developed by real-world companies that act as sellers of surveillance software and exploit brokers for government agencies.

However, Kaspersky's recent reporting show that APT groups do not shy away from dipping into the underground hacking scene to acquire exploits initially developed by lone hackers for cyber-crime groups.

This Volodya character, who Kaspersky Lab characterizes as one of the most prolific vendors of zero-days earlier first came to light back in 2016 when selling an exploit under the moniker "BuggiCorp". At that time the hacker's actions were in the news after putting a Windows 0-day up for sale on the infamous Exploit.in cyber-crime forum.

At the time, the ad was a surprise because you'd rarely see a hacker advertise Windows 0-days in public since most such transactions happened in private. So the insider world watched as "BuggiCorp" dropped his initial asking price several times, from $95,000 to $85,000... at which point he eventually sold his 0-day to a cyber-crime group... an thus began to burnish his reputation.

BuggiCorp then established a dedicated clientele and continued to sell other 0-days privately, some with prices reaching up to $200,000.

The researchers at "GReAT" -- Kaspersky's Global Research and Analysis Team (GReAT) -- who are Kaspersky's elite APT tracking unit said that Volodya is fluent in Russian, although likely of Ukrainian origin because Volodimir is not a Russian name, but Ukrainian.

This hacker appears to be the author of the Win32k Elevation of Privilege Vulnerability fixed in March. That 0-day (since patched of course) had been in use by a cybercrime group focused on financially-focused thefts.

But this 0-day is only the latest of Volodya's accomplishments. An earlier one from 2016, which was also a Win32k Elevation of Privilege Vulnerability was a potent Windows 0-day that was found leveraged in the wild... and linked to the activities of the the infamous Russian Fancy Bear APT (also known as APT28, Pawn Storm, Sednit, Sofacy, or Strontium), and they are infamous for being one of the two Russian hacking groups that perpetrated the 2016 DNC hack.

Kaspersky has been watching Volodya and has seen him selling other 0-day exploits through the years to other high-end APT groups. And the hacker has also worked with low-end cybercrime groups, which have also been buying and using some of these zero-days as well.

Kaspersky noted that in addition to zero days, Volodya is also developing exploits for patched vulnerabilities -- which we would call 1-days -- and also exploits for older vulnerabilities, that are considered stable and reliable and could still work for unpatched machines.

As Kaspersky sums it up, Volodya appears to be making a profitable career out of selling 0-day and other exploits and is building quite a portfolio behind his name. We also don't know that "Volodya" is not the front for a larger group or team a exploit developers, or even an exploit brokering company that fronts for other independent hackers.

I wanted to share this news from Kaspersky because this fleshes out another wrinkle of the contemporary underground ecosystem for exploits. I keep flashing on that very first scene from the first "The Matrix" movie where Neo is selling a hack for something to his friends. It was released on March 31st of 1999, so just over 20 years ago. Back then it seemed like such a stretch -- real science fiction fantasy. But it was amazingly prescient.

Firefox to follow Chrome in banning browser extensions containing obfuscated code.
https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/AMO/Policy/Reviews-2019-05

We talked about this decision by the Google Chrome folks last October and the Chrome ban which took effect at the beginning of the year. Google's analysis had determined that 70% of malicious browser extensions employed deliberate code obfuscation. And it made their task of manually inspecting the code for malicious behavior so much more and unnecessarily difficult. So they decided that the simplest solution was to just say "no more deliberately unreadable code."

So last Thursday the Mozilla folks decided to follow Chrome's example with the updated policies to go into effect this June 10th... one month after the notice.

The guidlines have a lot to say about other aspects of extentions, "no surprised for the user", "value offered", etc. But on the issue of the format of the submission, Mozilla writes:

Add-ons may contain transpiled, minified or otherwise machine-generated code, but Mozilla needs to review a copy of the human-readable source code. The author must provide this information to Mozilla during submission along with instructions on how to reproduce the build.

The provided source code will be reviewed by an administrator and will not be redistributed in any way. The code will only be used for the purpose of reviewing the add-on. Failure to provide this information will result in rejection.

Add-ons are not allowed to contain obfuscated code, nor code that hides the purpose of the functionality involved. If external resources are used in combination with add-on code, the functionality of the code must not be obscured. To the contrary, minification of code with the intent to reduce file size is permitted.

So, as an industry we're creaking forward step-by-step, figuring out how to do all of this. How to be as fair as possible to every player, offer value, not place undue constraints upon others while offering as much protection as possible from those with dishonorable intentions.

A hacker hacked into and took over the botnets of 29 other hackers.
Ankit Anubhav is an IoT botnet researcher whose work we've looked at previously. He's a Principal Researcher with NewSky Security. He arranged a discussion / interview but a hacker who calls himself "Subby". What transpired was interesting and entertaining...
https://www.ankitanubhav.info/post/c2bruting

As we know, typical IoT botnets, including Mirai and QBot, rely upon obtaining access to their target devices using the device's weak/default credentials. However, as it turns out, the hackers themselves are not very security conscious and are using very poor weak and often default passwords to protect their command and control servers. This would mean that, in theory, another black hat could come along and launch a brute force attack against these Command and Control servers to obtain access to their bot networks without bothering to build up their own. And as Ankit learned during his interview, this is exactly what happened. The hacker hacker who calls himself "Subby" brute forced at least 29 IoT command and control servers, finding that they were using extremely trivial credentials:

BOTNET C2 IP	PORT	BOTNET FAMILY	USERNAME	PASSWORD
185.xx.xx.205	1791	LOLIGANG	Emily	rawr
139.xx.xx.31	8372	FROSTY	root	root
104.xx.xx.111	1543	SEPTEMBER	root	school
77.xx.xx.251	81	SORA	root	cam1
46.xx.xx.130	45	HOHO	admin	420
165.xx.xx.84	1791	LOLIGANG	jef	jef123
178.xx.xx.5	1791	LOLIGANG	root	wed
46.xx.xx.238	45	DEMONS	root	hoe
23.xx.xx.117	38149	F34RL3SS_TACTIX	k3znor	k3znor
157.xx.xx.173	666	JOSHO	haks	haks0
68.xx.xx.111	1791	LOLIGANG	oof	oof
193.xx.xx.144	1024	AMAKANO	root	root
68.xx.xx.183	2700	OWARI	root	bullet
167.xx.xx.115	1791	LOLIGANG	admin	raw
149.xx.xx.74	1791	LOLIGANG	goofy	root
185.xx.xx.206	30666	KILLER	un5t48l3	sikerim
165.xx.xx.138	45	HOHO	admin	admin
77.xx.xx.205	81	SORA	root	user2019
178.xx.xx.28	1791	LOLIGANG	root	root
185.xx.xx.200	7854	LIGHT	root	skrtt
185.xx.xx.164	1791	JOSHO	root	Ch4
67.xx.xx.63	1791	LOLIGANG	root	boi
178.xx.xx.28	9375	LOLIGANG	root	root
173.xx.xx.223	81	SORA	root	wdj123
185.xx.xx.238	9375	Z3HIR	delay	gay
185.xx.xx.85	6667	SPC	website	api
185.xx.xx.199	1791	AKIRA	yakou	1337
185.xx.xx.249	9375	Z3HIR	psn	root
46.xx.xx.135	3301	KALON	ankit	ankit

Ankit's Interview with Subby...

I decided to ping Subby to know more answers besides the data, like why and how he is doing this, and what is the motive. Some of the excerpts from the interview are as follows:

Ankit - What technique you are using for brute forcing the servers?

> *Subby - I have a network of honeypots configured to capture binaries over Telnet/SSH. The captured C2 IPs are then port scanned via NMAP to find the C2 port. For brute forcing, I am using a dictionary style attack coupled with a password list which has common user: pass combo's. In addition to this, each C2 undergoes a random style password attack which continues up to 6 alphanumeric characters under the user 'root'. I change the user to something specific if I have prior knowledge of the C2. Each cracked password is added to the password list used when brute forcing the C2s in future.*

Ankit - As you have found out, many of the credentials are very weak. Why you think this is happening?

> *Subby - It's obvious as to why this is happening. A large percentage of botnet operators are simply following tutorials which have spread around in the community or are accessible on YouTube to set up their botnet. When following these tutorials, they do not change the default credentials. If they do change the credentials the password they supply is generally weak and therefore vulnerable to brute forcing.*

Ankit - How much total bot count you have achieved brute forcing these c2?

> *Subby - Within the 1st week of brute forcing, I surpassed 40,000 devices. This was quite an inflated number due to possible duplication. It is well documented that botnet operators like to artificially increase their bot count. I estimate the number to be closer to 25,000 unique devices. I was able to get a reliable network traffic graph produced of the traffic generated from all the botnets combined and it was just under 300gbit/s. This high number was achieved because of the vast amount of Digital Ocean servers on many of the botnets. It is well known that Digital Ocean are relatively slow in comparison to other hosts when dealing with abuse complaints. Since then, the number of C2's vulnerable to brute forcing has lowered considerably, (30-40%).This is likely due to how vocal I've been when brute forcing the servers, I have actively contacted botnet operators letting them know that I managed to obtain access to their C2.*

Ankit - Why are you doing this? Are you using this for DDoS?

> *Subby - The main reason I undertook this task initially was to see how well brute forcing would work on C2 servers and whether it would be an efficient way of getting access to devices, rather than having to use exploits or the usual loading onto devices with weak passwords via Telnet/SSH. Since Mirai was released, Telnet has slowly become saturated and it's hard to get a decent number of bots. I have only used the C2s to attack network traffic graphs which are setup to be attacked to analyze inbound traffic.*

Ankit writes in this "Conclusion"...

In one previous case, we observed the SQL database of an IoT botnet having root:root credentials before, but as we see now, the problem is bigger and not a one-off case. Pure novices in the field of IoT are increasing. We are not talking about script kiddies, but such low skilled actors who are unable to set up a botnet from source, yet they want to launch a DDoS by doing nothing other than pressing a button. We also observed mistakes as novice as not replacing the botnet dummy C2 IP with their own IP.

Unstable (UN5T48L3), the Turkey based author of Z3hir IoT botnet has gone to the extent to release a video where he tells how to replace the dummy C2 (0.0.0.0) with the attacker's IP. When asked about the video, he told "Yes, these script kiddies are not changing IPs and they are blaming me when the botnet does not work". Recently @VessonSecurity also observed a similar case where his honeypots found attacks with dummy C2 "INSERT-IP-HERE-N*GGA", pointing to the fact that the threat actor forgot to change the dummy C2 with a real one, yet proceeded to attack the IoT devices (and subsequently the honeypots) with a non functioning C2.

Interestingly, despite not knowing what they are doing, the script kiddies often succeed, thanks to good support and tutorial videos by threat actors. In many cases like having a secure password and updating the IoT device can save one from these low hanging fruit attacks.

D-Link is not encrypting its DCS-2132L camera data.
https://www.welivesecurity.com/2019/05/02/d-link-camera-vulnerability-video-stream/

Title: "D-Link camera vulnerability allows attackers to tap into the video stream"
Title: "ESET researchers highlight a series of security holes in a device intended to make homes and offices more secure."

The camera in question is the D-Link DCS-2132L. It's a very popular camera with a very large user base being sold by many many online and large brick and mortar retailers. So there's a reasonable chance that some of our listeners will have these cameras.

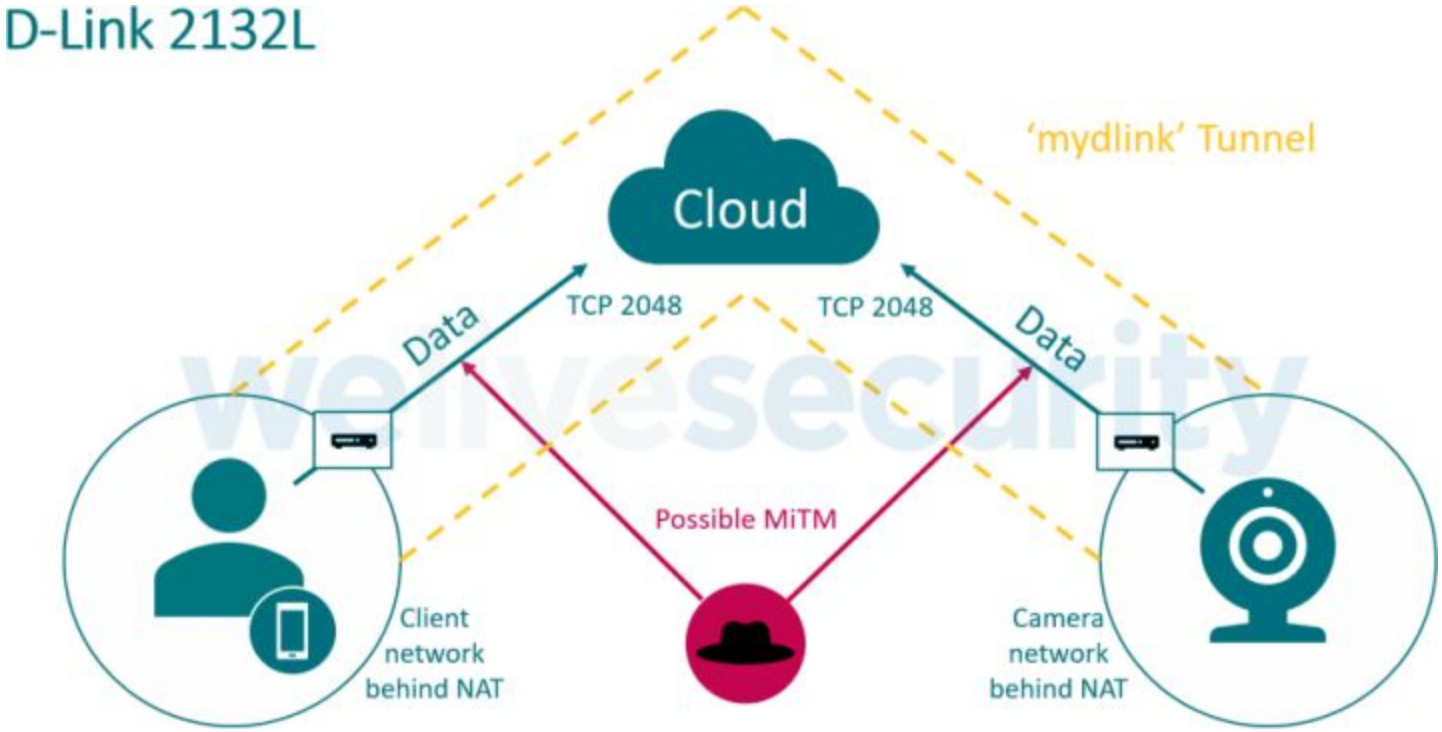
ESET writes:

Many people are looking to improve the security of their homes or offices by installing "smart" cameras. With a direct connection to the internet, their surveillance stream is just a few clicks away and available at any time. Yet, this kind of convenience can quickly turn sour if the camera suffers from a security vulnerability that opens the door to unauthorized actors. As shown by ESET smart home research, this is the case with the D-Link DCS-2132L cloud camera, which allows attackers to not only intercept and view the recorded video, but also to manipulate the device's firmware.

The most serious issue with the D-Link DCS-2132L cloud camera is the unencrypted transmission of the video stream. [Yes, you heard that correctly. It's an Internet-connected streaming video security camera… that doesn't encrypt its video stream!] ESET writes: It runs unencrypted over both connections – between the camera and the cloud and between the cloud and the client-side viewer app – providing fertile ground for man-in-the-middle (MitM) attacks

and allowing intruders to spy on victims' video streams.

In other words, they really didn't pay any attention to security. Even if encrypting the output of the IoT camera would have somehow been challenging, it would have been trivial to encrypt the feed returning from their cloud servers to the user's monitoring app. But they didn't bother with that, either.



The viewer app and the camera communicate via a proxy server on port 2048, using a TCP tunnel based on a custom D-Link tunneling protocol. Unfortunately, only part of the traffic running through these tunnels is encrypted, leaving some of the most sensitive contents – such as the requests for camera IP and MAC addresses, version information, video and audio streams, and extensive camera info – without encryption.

A MitM attacker intercepting network traffic *[actually, just any passive eavesdropper]* between the viewer app and the cloud or between the cloud and the camera, can use the data stream of the TCP connection on the server (cloud) port 2048 to see the HTTP requests for the video and audio packets. These can then be reconstructed and replayed by the attacker, at any time, to obtain the current audio or video stream from that camera. In our experiments, we obtained the streamed video content in two raw formats, specifically M-JPEG and H.264. To reconstruct the video stream, one needs to take a few steps (which can be easily automated via a simple program or a script):

1. Identify the traffic that represents video streams. This traffic consists of multiple blocks of data, each block having a specific header and defined length.
2. Separate the data parts from the headers.
3. Finally, the parts of the video are merged into one file.

And as if unencrypted video and audio were not sufficient, thanks to the fact that the device supports on-the-fly firmware updates but does not offer any firmware update authentication mechanism, a sufficiently motivated adversary could surreptitiously install whatever malicious firmware they might wish to into any camera at any time…

… not that they really need to, though, since the device's manufacturer-supplied firmware really is sufficiently malicious.

And, of course, the camera messes with UPnP to expose its insecure HTTP server to the internet.

As ESET wrote: [The] D-Link DCS-2132L also had a few other minor, yet still concerning, issues. It can set port forwarding to itself on a home router, by using Universal Plug and Play (UPnP). This exposes its HTTP interface on port 80 to the internet and can happen without the user's consent even with the "Enable UPnP presentation" or "Enable UPnP port forwarding" fields in the settings disabled. [This really is unbelievable.] Why the camera uses such a hazardous setting is unclear. Currently close to 1,600 D-Link DCS-2132L cameras with exposed port 80 can be found via Shodan, most of them in the United States, Russia and Australia.

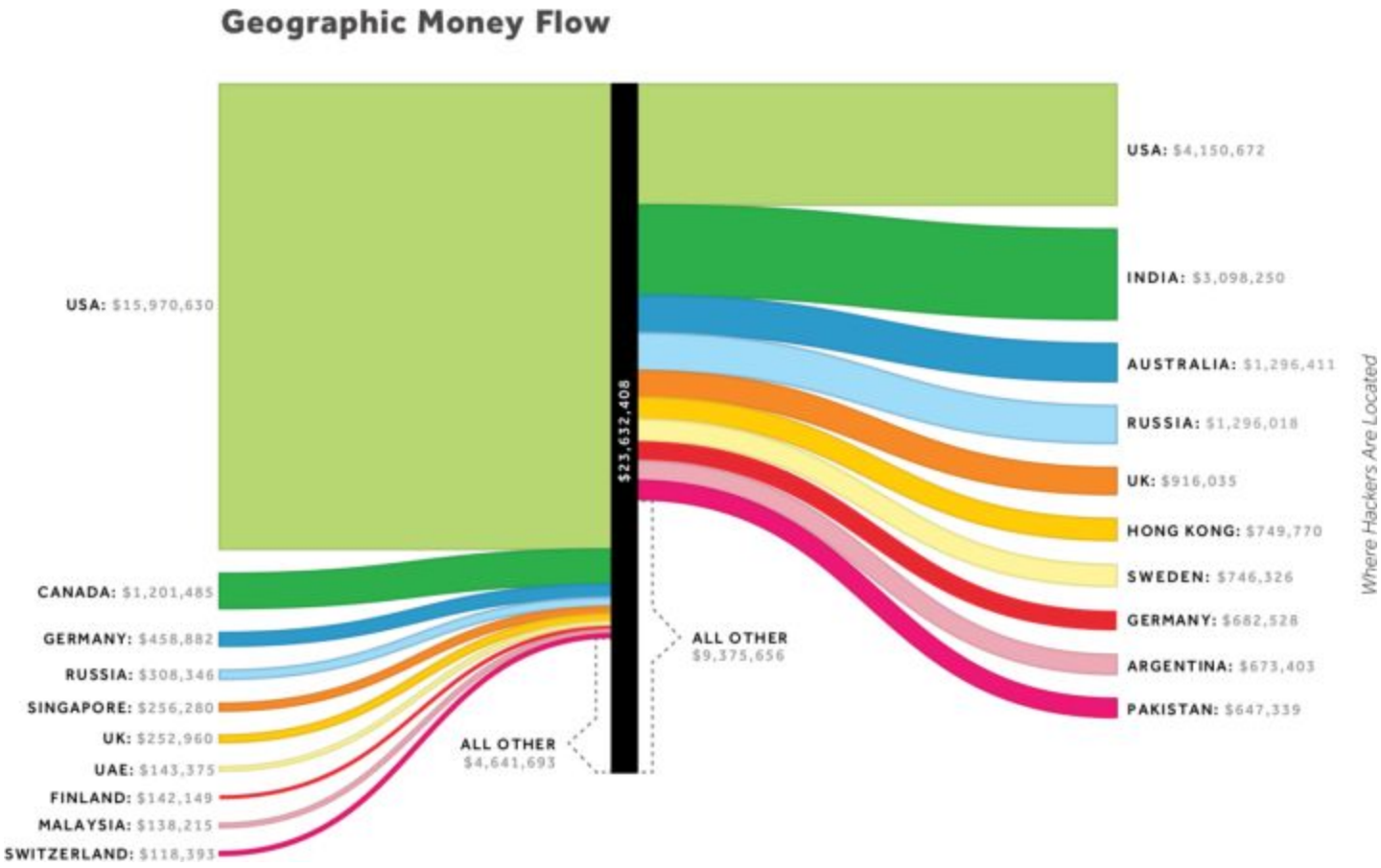
As a part of responsible disclosure, ESET reported the discovered issues to D-Link on 22nd August 2018, including vulnerable unencrypted cloud communication, insufficient cloud message authentication and unencrypted LAN communication.

D-Link responded immediately, informing ESET that the vulnerability reports had been forwarded to their research and development teams, promising follow-up.

Since then, some of the vulnerabilities have been mitigated – according to our tests, the "mydlink services" plug-in is now properly secured – although other issues persist. At the time of writing the most recent version of firmware available for download was from November 2016 and did not address the vulnerabilities allowing malicious replacement of the camera's firmware, as well as interception of audio and video streams.

D-Link DCS-2132L camera is available on the market. Current owners of the device are advised to check that port 80 isn't exposed to the public internet and reconsider the use of remote access if the camera is monitoring highly sensitive areas of their household or company.

A visualization of the flow of Software Bug Bounty Money, from those putting up the bounties (on the left) → to those collecting the bounties (on the right):



(This was from an article complaining about the misplaced bounties being offered by the UK. They are focused upon rewards rather than upon working to improve the quality of their code.)

Another User-Agent gotcha with Google Docs and the new Microsoft Edge browser. We were just talking about the mess with user-agent headers and the idea that websites are expected to tune the code they produce for this or that browser.

Frankly, my approach would always be to simply choose the lowest common denominator feature set and write to that so that the same thing works independent of and across all web browsers. But I understand that when what's being delivered is a cutting edge web-based application, the lowest common denominator might well be too low. This is the situation with, for example, Google Docs, which has always presented a warning message when a web browser is not known to be capable of running the online application.

But Microsoft's new and forthcoming Edge browser based upon the common Chromium engine would be expected to be able to run Google Docs without trouble. So observers were understandably puzzled when Edge showed the message "The version of the browser you are using is no longer supported." -- No LONGER? It turned out upon examination and experimentation that Google Docs maintains an explicit whitelist of known-compatible web browsers... and the new Chromium Edge is transmitting a slightly altered User-Agent header that Google Doc's browser checker didn't recognize.

We were previously talking about the user-agent header in the context of Microsoft's Edge presenting different UA headers depending upon the domain being visited by its user. So it would presumably be simple for Microsoft to present a Google Docs compatible User Agent header to Google Docs... Docs could be taught about the User Agent header being produced by Microsoft's forthcoming browser. I'm mildly curious to see how that one turns out. :)

And Google Earth won't run under Microsoft's Edge browser for the same reason?
No. There the reason is different and the incompatibilities are real.

When users try to launch the Google Earth web app with Microsoft's new Chromium Edge, they get the following error: *"Aw snap! Google Earth isn't supported by your browser yet. Try this link in Chrome instead. If you don't have Chrome installed, download it here. Learn more about Google Earth."*

Eric Lawrence, Microsoft's Product Manager for Edge explained on Twitter thread that the issue arises from the fact that the Chromium-based Edge browser does not ship with the Portable Native Client (PNaCl) component. That's the architecture-neutral version of Native Client (NaCl) which was used by Google when they converted Google Earth into a web app back in 2017.

Google's plan is to move Google Earth over into WebAssembly (WASM), but they're running a little behind schedule on that project.

Jordon Mears is Google's Lead Manager for "Google Earth on Web". He said: "It has always been our intention to bring Earth to as many people as possible on as many browsers as possible. Parallel to our efforts in building the first iteration of Earth on web, Google and the Chrome team have been active participants in the W3C process for WebAssembly."

He continued: "WebAssembly also has the advantage of being supported by the four major browsers (Chrome, Edge, Firefox, Safari). So since the April 2017 launch of Earth on web, the Earth team has been working to port Earth on web over to WebAssembly from Native Client."

A demo of Google Earth running in WebAssembly on Chrome, Firefox and Chromium was presented by Google's Ben Galbraith, the leader of the Chrome Web Platform team, and Dion Almaer, Google Engineering Director, during the Chrome Dev Summit 2017.

But it does seem to be taking longer than they planned. What this means is that Microsoft will be waiting a while for Google Earth to be fully finished for WebAssembly and Microsoft apparently won't bother with the effort to bring up the Portable Native Client (NaCl) on Chromium Edge since it's an already deprecated technology.

Edge's planned future...
We're currently in the middle of Microsoft's 3-day Build 2019 and Microsoft's CEO Satya Nadella kicked off the 2019 conference with a keynote that touched on Microsoft's plans for its much anticipated Chromium-powered Edge web browser. On the list of coming goodies was...

Updated Privacy tools
Microsoft Edge will be acquiring some new privacy-centric features to help users understand how their data is being used by sites across the web, and to provide some control over various features of the browser, including the innate ability to block trackers.

Edge's privacy panel will allow users to select from among different levels of information sharing. If they choose a privacy-focused predefined level, the system will automatically configure the browser to protect data and provide options to configure the browser's behavior to third-party trackers. This sounds similar to Firefox's Content Blocking feature, but it may not (at least yet) provide the degree of customization possible to allow mixing and matching of different settings. Microsoft likely views this as a control-complexity reduction.

IE11 Mode
We've touched on this before, but some enterprises must have IE11 for their internal apps. So Microsoft formally confirmed that Edge will also offer an Internet Explorer mode to bring full IE11 compatibility to Microsoft Edge for compatibility with legacy sites and applications.

Developer Tools
And Microsoft Edge will also offer powerful developer tools built upon the familiar Chromium DevTools. Developers will be able to use the built-in tools to inspect and debug any Microsoft Edge-powered web content, including Progressive Web Apps and WebView with a consistent UI experience across all of those targets.

Miscellany

https://send.firefox.com/
Filemail imposes a two files per day per IP. Firefox Send - no such limit.

Post-Coinhive Cryptojacking

So what happened after the Coinhive shutdown?
As we previously discussed, the highly controversial Coinhive browser-based Monero coin mining facility voluntarily closed its doors one month ago, on March 8th. At the time it was expected that the vacuum would be quickly filled with alternative mining solution. So what has happened??

To address and answer that question, Jérôme Segura, Head of Threat Intelligence for MalwareBytes examined the state of Cryptojacking now that Coinhive is no more.

https://blog.malwarebytes.com/cybercrime/2019/05/cryptojacking-in-the-post-coinhive-era/

September 2017 is widely recognized as the month in which the phenomenon that became cryptojacking began. The idea that website owners could monetize their traffic by having visitors mine for cryptocurrencies in their browser was not new, but this time around it became mainstream, thanks to an entity known as Coinhive.

The mining service became a household name overnight, and quickly drew ire for its original API, whose implementation failed to take into account user approval and CPU consumption. As a result, threat actors were quick to abuse it by turning compromised [web]sites and routers into a large illegal mining business.

The ride was wild but, as we came to see, short-lived, as Coinhive shut its doors in March 2019 following months of steady decline and loss of interest in browser-based mining.

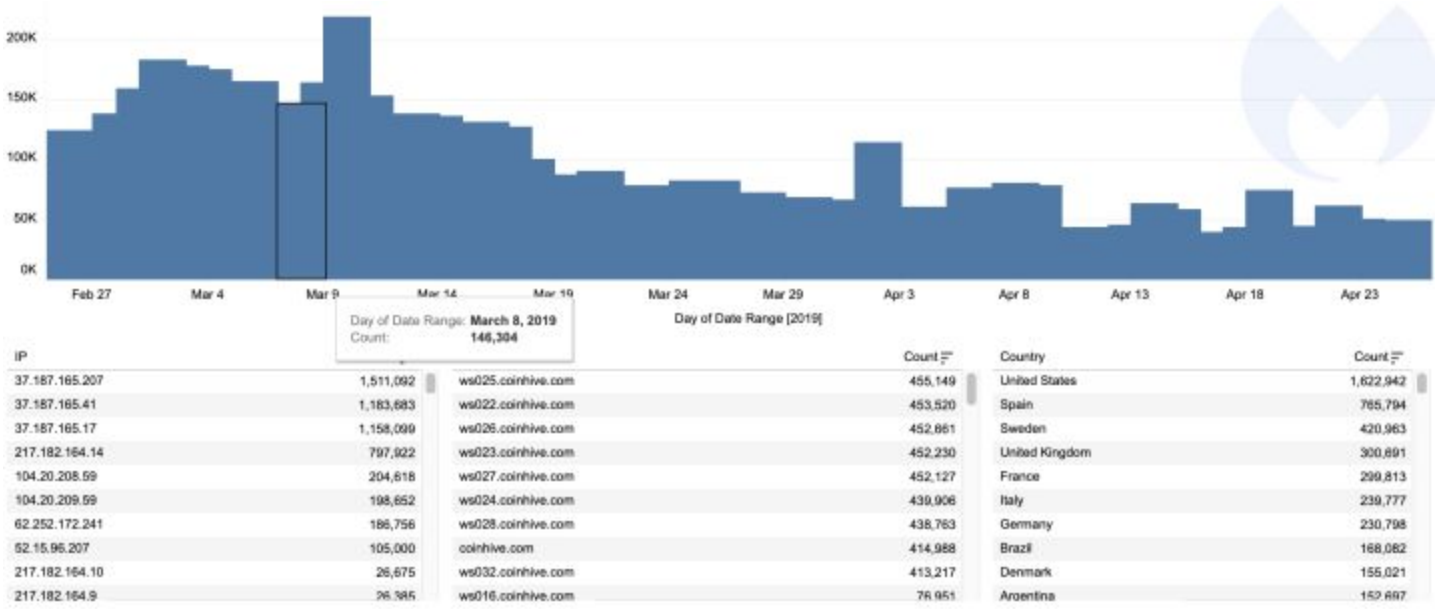
[I'll interject here that, as we talked about at the time, the general collapse in cryptocurrency evaluations from their highs of a few years ago did much to take the wind out of the market for monetizing the stolen CPU cycles of innocent web users. It just wasn't worth it any longer.]

Anyway, Jérôme continues... As such, this blog will strictly focus on web-based miners, which were impacted the most by Coinhive's closure. It will not cover malware (binary-based) coin miners that are still infecting PCs, Macs, and servers.

Coinhive relics left behind

Interestingly, we still detect thousands of attempts for Coinhive-related domain requests, even though the service announced it was shutting down on March 8. Over the past week, our telemetry recorded an average of 50,000 attempts per day.

Digging deeper, we see that a large number of websites and routers have never been cleaned, and the bits of JavaScript requesting the Coinhive library are still there. Evidently, with the service down, the necessary WebSocket that sends and receives data between client and server will fail to connect to the server, resulting in zero mining activity or gain.



Is cryptojacking still a thing?

To answer that question, we go back to the early adopters of browser-based mining: torrent sites. Visiting a proxy for "The Pirate Bay" with our browser we spot something familiar enough —our system's CPU usage is maxed out at 100 percent. So, yeah… apparently in some corners of the web cryptomining itself remains alive and well.

As we'll recall, this is what started the cryptojacking trend back in 2017, when users weren't told about this code running on their machine, let alone that it was hijacking their processor for maximum usage.

In this instance, the mining API was provided by CryptoLoot, which was one of Coinhive's competitors at the time. Malwarebytes reports that while they are seeing nowhere near the same levels of activity as they saw during the fall of 2017 and early 2018, according to their telemetry, they are still blocking more than 1 million requests to CryptoLoot each day!

There are a few other services out there, and it's worth mentioning CoinIMP, which we've seen used more sensibly on file-sharing sites.

Router-based mining still going

While the number of compromised sites loading web miners was going down in 2018, a fresh opportunity presented itself, thanks to serious vulnerabilities affecting MikroTik routers worldwide.

By injecting mining code from a router and serving it to any connected devices behind it, criminals could finally scale the process so it was not limited to visiting a particular website, therefore generating decent revenues.

The number of hacked routers running a miner has greatly decreased. However, today we can still find several hundred that are harboring the old (inactive) Coinhive code, and have also been injected with a newer miner (WebMinePool): https://www.webminepool.com/

Campaigns gone missing
Perhaps the biggest change in cryptojacking-related activity is the lack of new attacks and campaigns in the wild targeting vulnerable websites. For example, in spring 2018, we saw waves of attacks against Drupal sites where web miners were one of the primary payloads.

These days, hacked sites are leveraged in various traffic monetization schemes that include browlocks, fake updates, and malvertising. If the Content Management System (CMS) is Magento or another e-commerce platform, the primary payload is going to be a web skimmer.

We might compare cryptojacking to a gold rush that didn't last too long, as criminals sought more rewarding opportunities. However, we wouldn't rush to call it fully extinct.

We can certainly expect web miners to stick around, especially for sites that generate a lot of traffic. Indeed, miners can provide an additional revenue stream that is, as concluded in this Virus Bulletin paper,"depend[ent] on various factors, including, of course, the value of cryptocurrencies, which historically has been volatile."

The next time cryptocurrencies see an upturn in the market, expect threat actors to do what they do best: exploit the situation for their own profit.

~30~