

Security Now! #878 - 07-05-22

The ZuoRAT

This week on Security Now!

This week we look at Chrome's 4th 0-day of the year and at another welcome privacy-enhancing bump from Firefox. And also share the disclosure and forensic investigation of the bug bounty clearinghouse HackerOne's discovery of a malicious (now ex-) employee among their ranks. And some listener feedback draws us into a discussion of the nature of the vulnerabilities of connecting Operation Technology systems to the Internet, and also some hope for the future amalgamation of the currently-fragmented SmartHome IoT industry. And before we start into our deep dive into some new and worrisomely prolific malware, we're going to consider whether we'd rather have one 9-inch pizza or two 5-inch pizzas? As always, another gripping episode of Security Now!

**Good friend of the show, Evan Katz, points out just
How pervasive "Cookie Notices" have become...**



0-day Watch

Chrome: v103.0.5060.114 (Official Build) (64-bit)

Yesterday, for Windows users, Chrome jumped to 103.0.5060.114. This was an emergency update to address a high-severity 0-day vulnerability which was being exploited by attackers in the wild. And it's also the fourth Chrome 0-day patched so far this year.

As usual, very little is being said at this point. And there's really little reason to say more. The only thing we know is that Google was made aware of this by the Avast Threat Intelligence team last Friday, July 1st. So I have no problem with the idea that they were told of this on Friday before the long US 4th of July weekend, and that my Chrome browser has already updated itself as I'm writing this on Monday evening on the 4th of July. Thank goodness Microsoft didn't win the browser wars with their track record of slow or nonexistent responses to known critical problems.

We do know that the trouble was a high severity heap-based buffer overflow weakness found within Chrome's WebRTC (the Web Real-Time Communications) component. And we also know that this reported problem was promptly assigned a CVE-2022-2294, even though it didn't require any user involvement. We recently learned that Microsoft doesn't bother assigning their problems CVEs if no user action is required. Is that a rug? Oh, good. Just lift the corner and we'll sweep it under.

Browser News

Mozilla released Firefox v102 last week which includes a new privacy-enhancing feature: it strips some of the URL parameter crap from the ends of URLs that are used to track us around the web. We talk a lot about tracking cookies. But another more blatant way of tracking is to customize individual URLs, embedding a unique tracking tag into the end of the URL. Many companies, including Facebook, Marketo, Olytics, Drip, Vero and HubSpot, utilize these custom URL query parameters to track the clicks of their links. When the click comes in, the URL closes the loop to let them know where, when and to whom it was issued.

Facebook appends a big long hairy query parameter named "fbclid" that looks like this:

<https://www.facebook.com/{blah-blah-blah}?fbclid=IwAR4HesRZLT-fxhhh3nZ7WKsOpaiFzsg4nH0K4WLRHw1h467GdRjaLiWbLs>

So now, though it is reportedly not as complete and thorough as the Brave browser's URL stripping, Firefox v102, which is out now, has added this new 'Query Parameter Stripping' feature. It will automatically strip various known and recognized query parameters which are used for tracking from URLs as they are used. Firefox will strip the following tracking parameters from URLs when you click on links or paste an URL into the address bar:

Olytics: oly_enc_id=, oly_anon_id=
Drip: __s=
Vero: vero_id=
HubSpot: _hsenc=

Marketo: mkt_tok=
Facebook: fbclid=, mc_eid=

This is great. But modifying URLs is error-prone and feels a bit messy to me. Presumably, Facebook could change the name of their tracking parameter and then it would be allowed until Firefox was updated.

In any event, I'm still using Firefox and I'm glad to have that incremental improvement. Since my Enhanced Tracking Protection was already set to "Strict" — as I would imagine is the case for most of our Firefox-using listeners — this was already enabled for me when Firefox updated v102. Oddly, however, until and unless additional tweaks are made, these tracking parameters will NOT be stripped when using Firefox in its Private Mode, even with Strict mode enabled.

To also enable the feature in Private Browsing (pb) mode, goto Firefox's "about:config" by placing that in the address bar. Then narrow down the gazillion items by searching for "strip". You should see two adjacent lines one already set to true, and a second one you will want to set to true:

privacy.query_stripping.enabled	true
privacy.query_stripping.enabled.pbmode	false <— Set this to TRUE, too!

And, finally, since URL stripping is messy, it's possible, though unlikely, that some sites might object or malfunction. So keep in mind that if something doesn't work you can flip it off briefly.

Security News

HackerOne discloses a malicious insider incident

One thing this podcast probably doesn't spend enough time focusing upon is the very real threat from insiders gone bad. It doesn't matter how strong one's perimeter defenses may be if someone with malicious intent walks right through the front door, smiles at the security guard, swipes his badge, and then settles down in his cubicle or office with full internal access to the corporate network.

We've briefly touched on the notion that our government's three letter agencies might bring an existing employee or two who have critical access into their fold under the auspices of patriotism and national security. Clearly, private company employees were aware and silent when the NSA was establishing its massive Internet taps at the major domestic network exchange points. And we have deep suspicions that some older crypto designs may have been influenced under the guise of "this will be good for Uncle Sam." In these instances, people knew and were keeping quiet. And then there's the insidious possibility that a truly malicious foreign agency might bribe someone, or perhaps extort someone who has an embarrassing weakness to "just wipe all of your fingerprints off this little USB device, then plug it into the back of the break room printer and walk away. We'll know when you have, and a parcel with \$50,000 in crisp \$100 bills will be delivered to you privately."

Another possibility is the self-corruption of a trusted employee who's able to subvert their employer's own business to make money on the side. And that's what was revealed last Friday by the famous bug bounty program, HackerOne.

Because it's possible, and often appears to happen, that the same bugs are found by different researchers who happen to be looking in the same place around the same time, bug collision reports — and bounty collisions — are not uncommon. So it's not unusual for a hacker to complain when a bug they confidentially submit for consideration of a bounty is denied and paid to someone else. It can happen when no one is at fault. But to HackerOne's credit, they investigate such reports, and in this case, once clear evidence of an insider gone bad was found, they went public with the details.

Where's what HackerOne's disclosure explained:

On June 22nd, 2022, a customer asked us to investigate a suspicious vulnerability disclosure made outside of the HackerOne platform. The submitter of this off-platform disclosure reportedly used intimidating language in communication with our customer. Additionally, the submitter's disclosure was similar to an existing disclosure previously submitted through HackerOne. Bug collisions and duplicates, where multiple security researchers independently discover a single vulnerability, commonly occur in bug bounty platforms. However, this customer expressed skepticism that this was a genuine collision and provided detailed reasoning. The HackerOne security team took these claims seriously and immediately began an investigation.

Upon investigation by the HackerOne Security team, we discovered a then-employee had improperly accessed security reports for personal gain. The person anonymously disclosed this vulnerability information outside the HackerOne platform with the goal of claiming additional bounties. This is a clear violation of our values, our culture, our policies, and our employment contracts. In under 24 hours, we worked quickly to contain the incident by identifying the then-employee and cutting off access to data. We have since terminated the employee, and further bolstered our defenses to avoid similar situations in the future. Subject to our review with counsel, we will also decide whether criminal referral of this matter is appropriate. Our full discussion of the incident is below.

They then lay out a detailed 9-day timeline, from June 22nd to last Friday July 1st, which I'll skip. But I think that the discussion of their investigation is interesting:

Our investigation has concluded that a (now former) HackerOne employee improperly accessed vulnerability data of customers to re-submit duplicate vulnerabilities to those same customers for personal gain.

The investigation began after a customer notified us of reportedly receiving a threatening communication, outside the HackerOne platform, about a vulnerability disclosure. We immediately launched an investigation. Within 30 minutes of the investigation, additional evidence surfaced that caused us to escalate the priority of the incident. We began to run down every scenario of a possible exposure to disclosure data, including potential exploitation of our application, a remote compromise of the hacker, customer, or analyst, a leak by misconfiguration, and others. There was information to support only one of our hypotheses, an internal threat actor.

Upon this discovery, we began a separate investigation into the insider threat with a contained group. These steps were necessary as we worked to investigate and eliminate the prospect of multiple insiders. We are now confident that this incident was limited to a single employee who improperly accessed information in clear violation of our values, our culture, our policies, and our employment contracts.

Within 24 hours of the tip from our customer, we took steps to terminate that employee's system access and remotely locked their laptop pending further investigation. We were able to reach our conclusion quickly using the following methods:

Our internal logging monitors employee access to customer disclosures for regular business operations, namely vulnerability intake and triage. Analysis of this log data suggested a likely actor soon after our internal investigation kicked off. Only a single employee had accessed each disclosure that our customers suspected of being re-disclosed by the threat actor.

The threat actor [had] created a HackerOne sockpuppet account and had received bounties in a handful of disclosures. After identifying these bounties as likely improper, HackerOne reached out to the relevant payment providers, who worked cooperatively with us to provide additional information. Following the money trail, we received confirmation that the threat actor's bounty was linked to an account that financially benefited a then-HackerOne employee. Analysis of the threat actor's network traffic provided supplemental evidence connecting the threat actor's primary and sockpuppet accounts.

We identified seven customers who received direct communication from the threat actor. We notified each of the customers of our investigation and asked for information related to their interactions. We are grateful for their involvement in our investigation and we thank them for their willing cooperation. Facts shared from their own investigations corroborated the conclusion from our investigation and allowed us to act more quickly.

We have issued platform bans for the employee's known HackerOne accounts. We have terminated the employee for violating our values, our culture, our policies, and our employment contracts. Subject to review with counsel, we will decide whether criminal referral of this matter is appropriate. We continue forensic analysis on the logs produced and devices used by the former employee. We are reaching out to other bug bounty platforms to share details in case their customers received similar communications from "rzlr". The threat actor's motives appear to be financial in nature.

The threat actor had access to HackerOne systems between April 4th and June 23rd of 2022.

Insider threats are always a real possibility. The best outcome is to prevent them from happening in the first place. But if they do anyway, having extensive previous access and traffic logging becomes invaluable for post-incident forensics. And casually disclosing through water cooler conversation that such logging is always being done can go a long way toward preventing such abuse in the first place. We've talked a lot about the fact that while no one likes the idea of being watched and their activities logged, an enterprise's network and the traffic it carries is NOT the sovereign property of its employees. It belongs to the corporation. A casual reminder of the

fact that what's done on the enterprise's network is being monitored can prevent such abuse from occurring.

Closing The Loop

ReliefTwitcher / @ReliefTwitcher

@SGgrc I thought of you today when I reached around to the back of my computer tower (yes, I still have one) to plug in a USB-A cable. There is a USB 3.0 expansion card back there and I know the orientation of the ports from years of experience.

I thought, "I know for sure this port is 'upside down.' Time to prove Gibson wrong!" But then, as I turned the plug to the correct orientation as I knew it to be, I noticed that the cable seemed to want to lay in an unnatural way that I had not noticed when it had been connected last.

This caused me to doubt my own certainty, so I turned it over. The cable lay more naturally this way. Shaking my head, I thought, "Maybe it's time to prove Gibson RIGHT!" I moved to plug it in and, sure enough, it did not go in. I turned it over again, the way I had it the first time, and in it went.

It seems that even when I know the correct orientation, the bug in the simulation introduces a chaotic variable to make me try the wrong way first!

I'm just sayin'...

Stefan Bang / @I_dont_like_MS

I rarely have the chance to bring something useful to the table, but I have some insights to the OT (Operational Technology) subject of last week, which I hope will not waste your time and also I think I can fix reality for you.

Operational Technology: *I work as a developer in the OT business in Denmark and I think it makes a lot of sense that OT protocols are insecure, they should be. Mostly protocols like Modbus over TCP or BACnet over IP are really old (from the 70s and 80s) RS485 protocols that have been made to work over ethernet.*

These protocols don't use any authentication, and BACnetIP is even a UDP protocol. But they are fast and reliable - and if I need to close a valve in order for a heat exchanger to not explode, I'd rather have it be insecure on a local isolated network, than have people hurt because a certificate had expired and no one had noticed.

The control units are also probably insecure because they are rarely updated. Most OT controllers are installed and forgotten in kindergartens and homes and such. They don't need to be secure if they are not accessible from the internet - because if you have physical access to the device, then it's easier to pull the plug on the pump than to break into the system.

When an OT system is accessible from the internet, it is often through a supervisory system on a server. The server is likely to be located in the cloud or on-prem at a service partner (like the company I work for). The security of the supervisory system is easier to maintain as the server can auto update and the firewall only needs to allow http(s) traffic. A site-to-server VPN will eliminate any protocol insecurities and is much easier to maintain, instead of securing and maintaining the security of every SCADA protocol (renewing certificates, securing private keys and so on).

I'll quote a wise man: "security is hard!" and "complexity is the enemy of security".

By using a VPN, the only things that need to be secure is the VPN and the webserver of the supervisory system. This is easier and much less complex, than securing every single OT protocol – as there are A LOT of them.

Reality: *I don't think the simulation is broken. I think the person who writes and maintains the simulation listens to SN. The reason why you need to turn the USB-A plug 3-4 times for it to be allowed to enter the socket, is because you need to brute-force the physical port knocking sequence of the USB-A socket.*

*I wish you the best
Stefan Bang*

The trouble with Operational Technology over IP is that it's a classic case of convenience trumping security. I completely agree with Stefan that worrying about things like expiring certificates is a nightmare. But I think that the proper way to think about security is that it's the cost that's incurred when any non-IP technology wants to become an IP technology that shares the Internet's inherently routable IP network.

While that pressure regulating valve was using RS-485 protocol signaling, it was speaking an entirely different language and was fundamentally inaccessible to hackers in Russia. Computer users are more familiar with RS-232. RS-485 is closely related. It used differential signaling, meaning two wires rather than one where the data is contained in the difference in their voltages rather than in their absolute voltages. This creates a great deal of environmental electrical and magnetic noise immunity that can be a problem in heavy industrial environments. And whereas RS-232 is a point-to-point system, RS-485 uses device addressing to allow multiple devices to share a so-called "multi-drop" connection. So it was a very early form of serial bus architecture.

But the world was suddenly flooded with Ethernet cabling and Ethernet switches and routers whose cost dropped to near nothing as high manufacturing volume and massive inter-vendor competition fought for market share. Wired Ethernet is also a noise-immune differential signaling system, so it only made sense to move to it. However, it wasn't absolutely necessary to also move to IP. Other non-routable non-IP protocols have been carried over Ethernet and they would have been a far safer choice. But ultimately, as non-IP protocols, they would have also suffered the same fate as non-Ethernet RS-485 by ultimately being more expensive to deploy because they were in any way different from the IP technology that has overtaken the world. There was probably no way to resist the allure of near zero cost.

But now our mission-critical pressure-regulating valves are actually connected to, and reachable from, Russia. As “Kahn” was famously heard to say in the second Star Trek movie, explaining why Kirk wasn’t raising his own ship’s shields: “We’re all one big happy federation”, just before he blew the crap out of Kirk’s ship. Now, with Operational Technology systems supporting IP protocols, we’re all one big happy global network. What could possibly go wrong?

There is a simple and elegant solution to this, courtesy of the same IP technology that creates this problem in the first place: Operational Technology equipment, or for that matter any equipment that doesn’t want or need to be reachable from across the globe simply needs to set the TTL value of the packets they emit to a very low number. The absolutely, universally-honored fact of IP packet routing, is that every router which encounters any IP packet, first decrements the packet’s TTL — the Time To Live value contained in that packet. If that TTL is decremented to zero, NO ROUTER on the Internet will forward that packet. Period. Nothing would kill the Internet faster than packets which refuse to die. So every router honors that requirement before any other. Therefore, the simple expedient of emitting packets with a TTL of, say, two or three, will never affect any local networking use of such equipment, while at the same time flatly, elegantly and completely preventing them from ever reaching any foreign land.

This use of deliberately short TTLs is something that never seems to gain any traction, but it’s a beautiful solution when adding another security failsafe might be useful.

Dr. Nathan B. Gibson / @drnathanpgibson

I haven't gotten into smart home devices, among other reasons because of all the security issues you talk about with them. I was wondering about two things:

(1) When you talk about isolating your smart home network from your Internet network, it sounds like a lot of work. Are there any secure smart home hubs that can do this for you? I.e., you talk to the hub on your regular network but it sets up another air-gapped network and talks to the other devices on that? My FritzBox router actually has a setting for smart home devices, but I'm not really sure what it does.

My feeling is that we are still on the “don’t have it yet” side of the SmartHome compatibility revolution. So I haven’t yet invested in any single vendor’s hub technology because they’ve all been hoping to own the market for themselves. (Reminiscent of not synching passkeys across vendors.) So, I have individual hubless IoT WiFi devices running on their own isolated guest IoT network, courtesy of my ASUS router which offers up to four individual isolated guest networks. No matter how the IoT market sorts itself out, I would strongly recommend that anyone who purchases a WiFi router in the future be certain that it supports isolating guest WiFi networks.

And all that said, there is encouraging news on the home IoT front. Last summer, Ben Patterson for TechHive wrote:

We’ve been eagerly awaiting the arrival of Matter, the new, open-source, and platform-unifying standard that promises to make our various smart home devices play nicer with each other, and now comes word that we’ll have to wait a little longer.

*Initially, the Connected Standards Alliance (formerly the Zigbee Alliance) had announced that we might see the first Matter-enabled smart products by the end of the year. But as **Stacey Higginbotham at Stacey on IoT** reports, the CSA now says that a software development kit for Matter won't be finalized until the first half of 2022, which means the first Matter devices won't arrive until sometime next year.*

In a release announcing the delay, CSA CEO Tobin Richardson cited the "need to get it right" in terms of ensuring the upcoming Matter specification and SDK are "stable, deployable at scale, and meet market expectations for quality and interoperability."

[Amen to that! By all means, please, let's take our time to get it right.]

According to Stacey, CSA's CEO Tobin Richardson also blamed the resurgence of COVID, as well as the addition of nearly 30 more companies to the Matter group. Formerly known as Project CHIP (Connected Home over IP), Matter is an IP-based protocol that's compatible with Wi-Fi, ethernet, and Thread. Matter has the backing of some of the biggest names in the smart home market, including Amazon, Google, Signify (owner of the Philips Hue smart lighting brand), and Samsung's SmartThings.

Just last month, Amazon announced that all of its current, Alexa-enabled Echo speakers and displays will support Matter, while Google had previously said its Nest speakers and displays will support Matter, too. Apple's HomePod mini comes with its own integrated Matter radio.

Matter promises to unify the thicket of competing smart home platforms, as Matter-certified devices will be able to recognize each other and work seamlessly together across different ecosystems, including Apple's HomeKit, Amazon's Alexa, and Google's Assistant-powered Nest platform.

In other words, if you buy a Matter-certified smart gadget, you (ideally) will be able to control it with Alexa, Google Assistant, and Siri, and it should also work with any other Matter-enabled devices you own.

That's a welcome prospect for anyone who's pulled their hair out trying to make different makes and models of smart home devices work well together. For now, however, it looks like we'll have to keep coddling our stubborn smart gadgets through the end of the year, and probably even longer.

I also noticed DigiCert and Wemo's names associated with Matter. Wikipedia has this to say:

Matter, formerly Project Connected Home over IP (CHIP), is a royalty-free home automation connectivity standard, with manufacturers only incurring certification costs. Announced on December 18, 2019, Matter aims to reduce fragmentation across different vendors, and achieve interoperability among smart home devices and Internet of things (IoT) platforms from different providers. The project group was launched and introduced by Amazon, Apple, Google, Comcast and the Zigbee Alliance, now Connectivity Standards Alliance (CSA). Subsequent members include IKEA, Huawei, and Schneider. Matter-compatible products and

software updates for existing products are expected to be released in 2022. Although the Matter code repository is open-source under the Apache license, the Matter specification is licensed by CSA.

The Connectivity Standards Alliance is at: <https://csa-iot.org/> So it sounds like we'll be needing to do a podcast on this soon. And I won't be able to resist titling it: "What's the Matter?"

Ivan Neeson @zzymyn

Hey Steve I have a question about pass keys. Why would I want my passkeys on my PC at all? Isn't the whole point to keep the keys on the phone with its high security chip and use that to logon? So if someone hacks the PC they won't get any pass keys?

That's a good point, and it's true. I'm not sure how I'll come down on this issue. At the moment, for example, I have all of my many TOTP 2nd-factor keys on an iPhone that I have sitting next to me. I use the iOS app "OTP Auth" which I like a lot. So, obtaining a token takes no time. We're going to need to see how the flow works with smartphone cross-device authentication. This was, of course, the original mode for SQRL. When I first described SQRL to this podcast's audience "SQRL" stood for Secure QR-code Login, which was a novel concept back then. We saw something like that in Apple's presentation, though all of the mechanics were not clear. If it's super easy to present a website's FIDO2 passkey QR code to a smartphone to have it authenticate, that might indeed be sufficient and ideal. And perhaps we would only have our PCs storing passkeys that are less critical. We'll need to see how this all shakes out.

Jonathan / @sophtwhere

Noted that math geeks make difficult customers and pointed me to a Twitter thread...

I ordered a 9-inch personal Pizza. After a while, the waiter brought two 5-inch pizzas and said, the 9-inch pizza was not available and he was giving me two 5-inche Pizzas instead, and that I am getting 1 inch more for free! [We know where this is going.] I requested the waiter to call the owner over. I showed the owner the mathematical formula to calculate the area of a circle. Circle Area = $3.1415926 r^2$ So, the area of a 9-inch pizza is 63.62 sq.in, whereas a 5-inch pizza has only 19.63 sq.in. The two 5-inch pizza areas add up to 39.26 sq.in. I said that even if he gave me three 5-inch pizzas, I would still lose-out. So, how can you say you are giving me an extra inch for free? The owner was speechless and gave me 4 pizzas.

The ZuoRAT

Zuo is pronounced a bit like “sewer”, but more like: SO’-ah. Thus SO’-ah RAT. https://dictionary.hantrainerpro.com/chinese-english/translation-zuo_make.htm
ZuoRAT gets its name from the Chinese word for “left” because the RAT’s filename is “asdf.a” suggesting a keyboard walk across of the left home keys.

Last Tuesday, Black Lotus Labs, the threat intelligence arm of Lumen Technologies (formerly known as CenturyLink), has been tracking elements of what they say appears to be a sophisticated campaign leveraging infected SOHO routers to target predominantly North American and European networks of interest.

<https://blog.lumen.com/zuorat-hijacks-soho-routers-to-silently-stalk-networks/>

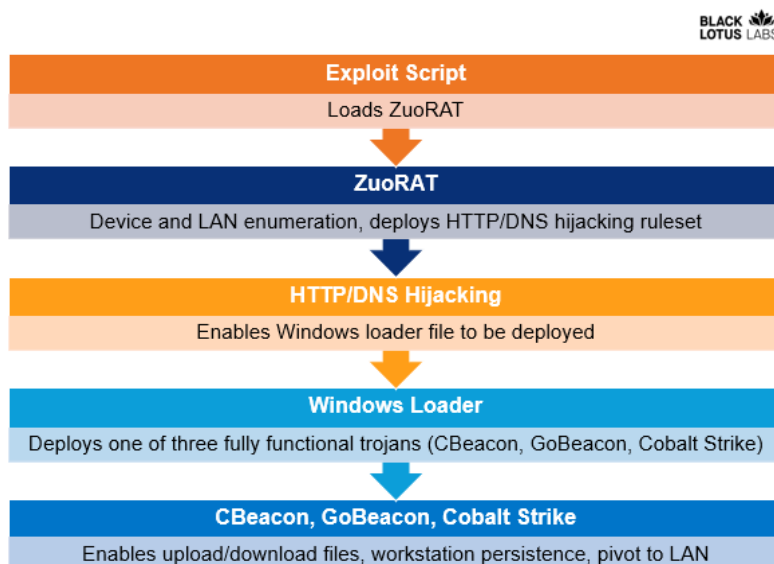
Generically, RATs (Remote Access Trojan's) are a dime a dozen and wouldn't usually command much of this podcast's prolonged attention. But in one of those “it's obvious (after the fact) that this would happen” disclosures, Black Lotus Labs revealed that they had uncovered a complex campaign that went undetected for nearly two years. And that the trigger for the campaign was the COVID-driven shift to working from home. How do remote well-financed nation-state actor bad guys get into well-protected corporate networks? They enter through the avenue of least protection and resistance: less secure remote employee networks.

Or, as Black Lotus explained:

The rapid shift to remote work in spring of 2020 presented a fresh opportunity for threat actors to subvert traditional defense-in-depth protections by targeting the weakest points of the new network perimeter — devices which are routinely purchased by consumers but rarely monitored or patched — small office/home office (SOHO) routers. Actors can leverage SOHO router access to maintain a low-detection presence on the target network and exploit sensitive information transiting the LAN. Black Lotus Labs is currently tracking elements of what appears to be a sophisticated campaign leveraging infected SOHO routers to target predominantly North American and European networks of interest. We identified a multistage remote access trojan (RAT) developed for SOHO devices that grants the actor the ability to pivot into the local network and gain access to additional systems on the LAN by hijacking network communications to maintain an undetected foothold. While we currently have a narrow view of the full extent of the actor’s capabilities due to the limited state of SOHO device monitoring in general, using proprietary telemetry from the Lumen global IP backbone, we have enumerated some of the command-and-control (C2) infrastructure associated with this activity and identified some of the targets. We assess with high confidence the elements we are tracking are part of a broader campaign.

Okay. So some powers of darkness realize early in the COVID lockdown that large numbers of employees of North American and European enterprises will be working from home. And that many of those employees will be inherently weakening their employer’s security. Of course, this

also occurred to many of those enterprises whose employees suddenly needed to have access to corporate resources which were once protected by virtue of having everyone physically located within the same internal LAN. And we talked about this at the time... that this sudden work from home shift was going to be straining enterprise security. Obviously, this occurred to others as well.



ZuoRAT may not have been built explicitly and expressly to fulfill this agenda, but it appears to have been perfectly designed for its role. The campaign consists of several components:

1. The first stage is a RAT that was developed for SOHO routers. We'll come back to take a closer look at it in a minute.
2. There's a simple loader for Windows machines compiled in C++.
3. And then there are three separate fully functional agents – two of which were custom-developed – and the one that wasn't is the Cobalt Strike Beacon which we previously covered in some details. Together and independently these full-function agents allow for full enumeration of the infected device, downloading and uploading files, network communication hijacking, process injection and more.

ZuoRAT is a MIPS file which will run on routers from ASUS, Cisco, DrayTek and NETGEAR. That executable can enumerate a host and internal LAN, capture packets being transmitted over the infected device and perform man-in-the-middle attacks, including DNS and HTTPS hijacking driven by predefined rules.

We've often talked about the threat and power of deliberate DNS corruption. As we know, DNS still transits over unsecured and unencrypted UDP packets. The reason Dan Kaminsky was able to get the entire Internet to update DNS overnight when he realized (and quietly shared) how vulnerable it was to spoofing, was because DNS really has no other protection than relying upon the goodwill and good behavior of all interconnecting networks. Therefore, if malware is able to set up shop on the router that's linking a residential LAN to the Internet, a great deal of damage can be done.

Black Lotus notes that what they found surprised them because attacks as severe as these, true DNS and HTTPS hijacking, have mostly been theoretical and rare. This is what they wrote:

While compromising SOHO routers as an access vector to gain access to an adjacent LAN is not a novel technique, it has seldom been reported. Similarly, reports of man-in-the-middle style attacks, such as DNS and HTTP hijacking, are even rarer and a mark of a complex and targeted operation. The use of these two techniques congruently demonstrated a high level of sophistication by a threat actor, indicating that this campaign was possibly performed by a state-sponsored organization.

The Windows loader that they analyzed reached out to obtain a remote resource and then ran it on the host machine. They assessed that it was used to load one of three fully functional second-stage agents, depending on the environment. There was:

- **CBeacon** – A custom developed RAT written in C++, which had the ability to upload and download files, run arbitrary commands and persist on the infected machine via a component object model (COM) hijacking method. And it was Windows-only.
- **GoBeacon** – A custom-developed RAT written in Go. This trojan had almost the same functionality as CBeacon, but also allowed for cross-compiling on Linux and MacOS devices.
- **Cobalt Strike** – We observed that in some cases this readily available remote access framework was used in lieu of either CBeacon or GoBeacon.

They said:

Analysis of multiple Windows samples revealed the consistent use of the same program database (PDB) paths, some of which contained Chinese characters, while others referenced a possible name or Chinese locality. Additionally, there was a second set of actor-controlled C2 infrastructure used to interact with the Windows RATs that was hosted on Internet services from China-based organizations, namely Alibaba and Tencent. Given the age of the first observed router sample, which was first submitted to VirusTotal in December 2020, as well as a sampling from Black Lotus Labs telemetry over a period of nine months, we estimate this multi-year campaign has impacted at least 80 targets, likely many more.

During their investigation of ZuoRAT's activity, they observed telemetry indicating infections stemming from numerous SOHO router manufacturers, including, as I noted before, ASUS, Cisco, DrayTek and NETGEAR. But they were less lucky in capturing any code, only being able to obtain the exploit script for a model Q20 router manufactured by JCG. In that case, the actor was found to be exploiting known CVEs 2020-26878 and -26879 by using a Python-compiled Windows EXE that referenced a proof of concept called ruckus151021.py. I found that python proof of concept over on Github from before it had been weaponized to obtain credentials and load ZuoRAT.

The weaponized script first performed command line injection to obtain authentication material, and then used the output from the command injection to perform an authentication bypass. This

chain of vulnerabilities allowed the actor to download a binary, then execute it on the host. The script they recovered contained four functions:

Function Name	Description
getpasswd	Sent a specifically formatted request to the remote host (targeted router IP address) then requested the URL “http://{TargetIPAddress}/cgi-bin/luci”, which resulted in the router providing back its password.
getloginsysauth	Used the remote host and previously obtained password from the function above to extract the sysauth cookie and the stok value.
execCmd	Sent a crafted request to the URL using the previously obtained information to invoke the telnet command.
telnet	On the remote router, opened the /tmp directory, removed any files named “asdf.a” (the ZuoRAT), then retrieved the latest version of the payload to run it on the infected machine before supplying the active C2 node.

The final stage of the exploit script downloaded the ZuoRAT agent framework. That framework enables in-depth reconnaissance of target networks, traffic collection & network communication hijacking. It can be divided into two components: the first contains functions that would auto-run upon execution of the file. The second component contains functions that were embedded into the file but were not explicitly called. Black Lotus believes that these additional functions were intended to be called by additional commands. Why are some active and some appear to just be along for the ride? ZuoRAT appears to be a heavily modified version of the Mirai malware.

Component 1: Core Functionality

The first component was designed to obtain information about the router and its locally-connected LAN, to enable packet capture of network traffic and to send the information back to Command & Control. Black Lotus believed that its purpose was to acclimate the threat actor to the targeted router they had landed on, as well as the contents of the router-adjacent LAN to determine whether or not there’s anything worth further exploitation.

The capabilities included functions to ensure only a single instance of the agent was present, and to perform a memory dump that could yield useful data stored in memory such as credentials, routing tables and IP tables, among other information. The file was initially executed by the threat actor via the command line, specifying an IP address and port for the C2 node. If the IP:port was not provided in the exploit script, the ZuoRAT code contained a default C2 hostname listed as cs.memthree[.]com, a domain originally purchased in October 2020.

Upon execution, the agent would launch a new process with a randomly generated, 32-character string using the characters A-Z and 0-9. Next, it gathered host-based information by running the uname command to send to the C2. It also attempted to gather the router’s public IP address by querying the following web services:

- [http://whatsismyip.akamai\[.\]com](http://whatsismyip.akamai[.]com)
- [http://ident\[.\]me](http://ident[.]me)
- [http://myipdnsomatic\[.\]com](http://myipdnsomatic[.]com)
- [http://ipecho\[.\]net](http://ipecho[.]net).

If ZuoRAT was unable to obtain a public IP address, it would delete itself under the assumption that it was being run in an isolated sandbox.

Next, ZuoRAT would connect to the C2 and attempt to bind a "listen" on port 48101. If the bind failed because the port was already in use, it would immediately terminate the current process to ensure that only a single instance of the Trojan was running on the compromised device.

ZuoRAT then used a scan function designed to survey the adjacent LAN's internal IP addresses. Specifically, it scanned for a hardcoded list of open ports, including: 21, 22, 23, 80, 135, 139, 443, 445, 808, 902, 912, 1723, 2323, 3306, 5222, 5269, 5280, 5357, 8080, 8443 and 9001. It performs an internal LAN scan of all machines for those IPs. In doing so, it's on the trusted internal network, inherently trusted by Windows firewall on those machines, and able to bridge traffic from the external command & control to anywhere within the network.

And once that was done, ZuoRAT would send the reconnaissance information it had obtained to the previously supplied C2. If the connection was being established for the first time, it occurred over port 55556; if the connection was being refreshed, communication switched over to port 39500. If the connection was successful, data would be transmitted. If errors were returned, the program slept and repeated the loop.

And finally, in preparation for establishing network capture capabilities, ZuoRAT allocated memory for increased performance and allocated a mutex semaphore insure that only one instance would run at once.

Then, if initiated by subsequent commands, an array of network functions would allow the remote threat actor to collect network traffic on UDP, DNS and some TCP connections where data might be sent in the clear:

- init_http_proto_match_rule
- init_https_proto_match_rule
- init_dns_proto_match_rule
- init_ftp_proto_match_rule
- init_socks_proto_match_rule
- init_scan_flag
- init_http_hij_info
- init_dns_hij_rule_list
- init_catch_file_match_info
- init_ip_port_record_list
- init_banner_record_list
- dns_plug_init
- udp_pcap_init
- pcap_platform_init
- netbroker_init

The functions allowed for fine-tuning over which traffic would be captured and forward to the remote command & control facility.

As its last act, a function is initialized to collect TCP data flowing over ports: 20, 21, 80, 8080, 443 and 8443 to allow the threat actor to obtain any credential which just might be passed in the clear and also to gain insight into the browsing activity performed by the end user behind the compromised router.

Remember that I mentioned the second component which didn't run commands on its own? It contains auxiliary commands sent to the router to be run under the attacker's command by additional modules that downloaded onto the infected machine. The module to download may have been informed by the device and network information gleaned from the first component.

In total, Black Lotus observed approximately 2,500 embedded functions, which included modules ranging from password spraying to USB enumeration and code injection. They focused on the LAN enumeration capability, which provided the actor additional targeting information for the LAN environment, and subsequent DNS and HTTP hijacking capabilities which have the benefit to the attacker of being traditionally quite difficult for defenders to detect.

Several secondary commands supported additional LAN enumeration and the collection of DNS configurations from the infected system. One function would send DNS information to a hard-coded IP address, 202.178.11[.]78. However, Black Lotus did not observe any telemetry from the compromised routers communicating with this IP address. This suggested that either the IP address was manually reconfigured or that it was no longer being used at the time of their analysis. However, it was one of three functions that contained an externally routable C2, like an IP address or a domain name.

Another function would gather host-based DNS and WiFi settings such as the BSSID (basic service set identifier) and the SSID (service set identifier) information. And the agent would gather the internal IP addresses and the MAC addresses of the devices from the ARP table, which can help an actor conduct a highly detailed assessment of a LAN.

Once the threat actor obtained information about the DNS settings and the internal hosts in the local LAN, there were several functions designed to perform DNS hijacking. These functions would look at the DNS requests that were being transmitted through the router and a custom DNS parser, providing statistics on the types of domains being requested by the victim. Other functions allowed the actor to update DNS hijacking rules specifying which domains to hijack, the malicious IP address resulting from the hijack and the number of times to trigger the rule. It would also capture the time at which the new rule was created and its task number, then flag it if the rule was active. In an older sample from 2020, a partial list of domains and IP addresses were hard-coded, and they included publicly routable and internal IP addresses:

91.196.70[.]49	077d.kse[.]com	192.168.100[.]30
202.178.11[.]78	www.baidu[.]com	172.230.88[.]99
2001[:]:A12C	2001[:]:A12D	www.sina[.]com

This should all serve to well establish the idea that this malware represents a huge amount of work on someone's part and that whoever it is, no one wants this installed on one's residential router.

Black Lotus also explains that the extent to which the threat actors have taken pains to hide the command & control infrastructure cannot be overstated. To avoid suspicion, they handed off the initial exploit from a dedicated virtual private server (VPS) that hosted benign content. As we have talked about before, they also leveraged other 3rd-party routers as proxy C2s that hid in plain sight through router-to-router communication to further avoid detection. And they rotated these proxy routers periodically to avoid detection.

One malware sample compiled in December 2021 was hosted at [http://141.98.212\[.\]62/asdfa.a](http://141.98.212[.]62/asdfa.a). Black Lotus' telemetry indicated that the C2 first became active in early September 2021 and was used with at least six waves of exploitation through October 2021. In an attempt to make the staging server appear more legitimate, in case anyone looked, the threat actor uploaded some content written in Arabic script on the hard-coded IP address's default page. They didn't find any subsequent malicious activity associated with the webpage and suspect it was uploaded as a ruse to avert suspicion. This is the sort of caution that Black Lotus has seen from highly sophisticated actors to evade detection.

Based upon their telemetry, they observed 23 devices with a persistent connection to this single command & control server during September and October 2021. All 23 devices were located in the U.S. and Canada. The majority of IP addresses communicated with the C2 over TCP port 9000, but a few communicated over other ports, including 55556, 55558 and 39500. The device types which were infected included, but were not limited to: Cisco RV 320, 325 and 420; Asus RT-AC68U, RT-AC530, RT-AC68P and RT-AC1900U; DrayTek Vigor 3900 and an unspecified range of NETGEAR devices. Based upon their analysis of the router malware and their telemetry, the Trojan first attempted to establish a TCP connection over port 55556 and, as noted above, subsequent connections were made to port 39500.

More recently, they've seen activity from a separate C2 103.140.187[.]131:6666 occurring from Feb. 22, 2022 – May 16, 2022... which has been acting similarly.

Black Lotus' report contained much more detail, but I'm sure that everyone has a good idea now how much work someone, apparently in China, has put into building and deploying a powerful network intrusion capability. And it's not just theoretical. It is in active use.

In summarizing what they would, Black Lotus wrote:

Though advanced actors have long demonstrated the capability and intent to target sensitive networks, the industry has uncovered only a handful of router-based malware specifically designed to covertly target them. The sudden shift to remote work spurred by the pandemic allowed a sophisticated adversary to seize this opportunity to subvert the traditional defense-in-depth posture of many well-established organizations. The capabilities demonstrated in this campaign – gaining access to SOHO devices of different makes and models, collecting host and LAN information to inform targeting, sampling and hijacking network communications to gain potentially persistent access to in-land devices and intentionally stealth command and control infrastructure, leveraging multistage siloed router to router communications – points to a highly sophisticated actor that we hypothesize has been living undetected on the edge of targeted networks for years.

