SECURITY NOW!

Transcript of Episode #878

The ZuoRAT

Description: This week we look at Chrome's fourth zero-day of the year and at another welcome privacy-enhancing bump from Firefox. We share the disclosure and forensic investigation of the bug bounty clearinghouse HackerOne's discovery of a malicious, now ex-, employee among their ranks. 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 smart home 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" pizza or two 5" pizzas. As always, another gripping episode of Security Now!.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-878.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-878-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Another zero-day exploit for Chrome. Some new settings you might want to turn on in Firefox. We'll walk you through it. And HackerOne has a malicious, now ex-, employee they want you to know about. Plus the latest in router exploitation with ZuoRAT. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 878, recorded Tuesday, July 5th, 2022: The ZuoRAT.

It's time for Security Now!, the show where we cover the latest security advances, things you should know to protect yourself, your network, your company, privacy information, and a little bit of how computers work and science fiction thrown in for good measure. All thanks to this guy. It's really the mind of Steve Gibson on display. Hello, Steve.

Steve Gibson: Oh, that's a sad statement, but it is true.

Leo: Thanks to Jason Howell. Did Jason do the show last week, or Mikah? Jason did it.

Steve: Yup, he did it last week, and we're going to see him two weeks when you're off sunning yourself, cruising in the Pacific.

Leo: No, it's Alaska. Well, it is the Pacific, but I don't think there'll be a lot of sun. I mean, there will be sunshine.

Steve: Not a lot of sun?

Leo: It's Alaska, so there will be glaciers, that's for sure.

Steve: Anyway, it was great working with Jason.

Leo: Yeah, he's wonderful.

Steve: He's comfortable with the podcast since I guess he's the producer of the podcast normally.

Leo: Yes.

Steve: And he also listens to it, so he was sort of up to speed on things. I didn't have to say, okay, now, we talked about this three weeks ago, and here's what that is. So he had the background. That was great.

So we are at our first podcast for July, as we start into the second half of the year, 878. And no one knows how to pronounce this, unfortunately. We're going with ZuoRAT?

Leo: Zuo, ZuoRAT. It's a Chinese word.

Steve: Yes. For those who are as phonetically impaired as I am, it's spelled Z-U-O-R-A-T, ZuoRAT. Of course RAT is Remote Access Trojan.

Leo: Oh.

Steve: And it's on our radar because a state-sponsored actor, apparently in China, has managed to install this bad remote access trojan in many SOHO routers, taking advantage of the fact that COVID drove people home. And so somebody realized, hey, we've got all - and this is U.S. and European targeted routers have been infected with this as a means of getting into the corporate network through a remotely located employee. So the world in which we live today.

But first we're going to take a look at Chrome's fourth zero-day of the year and at another welcome privacy enhancing bump that Firefox just received from Mozilla. We're also going to share the disclosure and forensic investigation of the bug bounty clearinghouse we've talked about often, HackerOne, and their discovery of a malicious, now ex-employee among their ranks. Can happen to everyone. And we don't talk about insider threats as much as we probably should. So I thought this would be a good opportunity to sort of take a look at that.

We also have some listener feedback which has drawn us into a discussion, or will, of the nature of the vulnerabilities of connecting, and this is a term that we discussed last week, Leo. We used to talk about SCADA devices. Now they're sort of more generically being called OT, or Operational Technology systems.

Leo: Ah, yeah.

Steve: So the nature of the vulnerabilities of these OT Operational Technology systems when they're merged onto the Internet. We also have some hope for the future amalgamation of the currently still fragmented smart home IoT industry. And before we start into a deep dive into some of this ZuoRAT prolific malware, we're going to consider whether we'd rather have one 9" or two 5" pizzas. So as always, another gripping episode of Security Now!.

Leo: All I know is pi r squared. So if you just would make the pizza square, we'd know. Right? It'd be easier.

Steve: Yeah, it'd be much, much easier to deal with.

Leo: Much easier. Picture of the Week time.

Steve: So we have the debut of a good friend of the show by the name of Evan Katz.

Leo: Know him very well. Yeah, he's a great guy.

Steve: Yeah, he is. He sent me this picture that I thought was really apropos of what we've been talking about with all these cookie notices. And what struck me was that for a retailer to put this sign up in their store demonstrates just how pervasive this issue is.

Leo: Everybody knows about cookies now, yeah.

Steve: Yes. And I can't identify the store. It looks like of like a Pottery Barn kind of place. It looks like it's in New York City, which is I know where Evan hangs out because it says NYC Tote for 20 bucks that's not related to this. But anyway, Evan's pointing to this big plaque that says - and it's a display of plates. And so the plaque says "These plates accept all cookies." Uh-huh. And then there's two buttons down below, also printed on this plaque. On the left it says "Manage Cookies," and the right has an old-style sort of Macintosh-like pointy finger saying "Accept all cookies." And that's what's being pressed here. So anyway, sort of fun for the retailer to do this, but also look at the degree to which this has permeated just common experience in the world. It's like, yeah.

Leo: And that smiling face there is Evan.

Steve: Indeed.

Leo: Pointing at that sign. We love Evan, yeah. Thank you, Evan.

Steve: Yup. So for our zero-day watch we have Chrome having moved to v103.0.5060.114. Yesterday for Windows users Chrome jumped to that version. It was an emergency update to address a high-severity zero-day vulnerability which was being exploited by attackers in the wild. It's also the fourth Chrome zero-day patch so far this year. But really, four is not bad. They're like, this is better than they were at this time last year. Because here we are already in the second half of the year, and this is only the fourth problem that they've had.

And as usual, Chrome/Google is saying very little at this point. And there's really no reason for them to say more except fix it. 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 U.S. holiday Fourth of July weekend, and that my Chrome browser was already updated, had already updated itself as I'm writing this Monday evening of July 4th. And I just 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 that was found within Chrome's WebRTC. That's the Web Real-Time Communications component. We also know that this reported problem was promptly assigned a CVE, which is nice to know, 2022-2294, even though it didn't require any user involvement. And of course we recently learned that Microsoft doesn't bother assigning CVEs to their problems if no user action is required. You know, is that a rug? Oh, good. Just lift the corner, and we'll sweep it under. So anyway, Google did the right thing, and they're also cruising along well with only having, you know, this being the fourth zero-day so far, and they made it to the second half of the year.

Mozilla released Firefox 102 last week which includes a new privacy-enhancing feature which I have mixed feelings about. It strips some of the URL parameter crap from the ends of URLs that are used to track us around the web. Certainly it's good for any of that to be stripped. The problem is that it's going to be kind of problematic. We talk a lot about tracking cookies. But another more blatant way of tracking us is to customize each individual URL to embed in the link a unique tracking tag which will just get passed along. Many companies, including Facebook, Marketo, Olytics, Drip, Vero, and HubSpot - and more, that's just some - utilize, they embed these custom URL query parameters because it's going to go wherever you click it. So when the click comes in, the URL essentially closes the tracking loop to let them know where, when, and whom it was issued to.

And for example, Facebook appends a big long hairy query parameter named "fbclid" that looks, you know, I put an example of it. I mean, it's like, after the question mark, "fbclid," and it's, god knows, looks like it's about 50 characters of gibberish.

Leo: It's a GUID; right? Or something like a GUID; right?

Steve: It's very much like that. This looks like about two GUIDs worth.

Leo: Yeah, it's long, yeah.

Steve: I mean, it's this big.

Leo: But they have three billion users. They've got to have long GUIDs.

Steve: That's right. You've got a lot of those people to differentiate among.

Leo: Yeah.

Steve: So, now, though it's reportedly not as complete and as thorough as the Brave browser's current URL stripping, at least Firefox v102, which I already had, so it was updated last week, it's added this new Query Parameter Stripping feature. It will automatically strip various known and recognized query parameters, and this is part of the problem, which are used for tracking from the URLs where they appear.

And for example, we could generically refer to third-party cookies as being bad, regardless of what the cookie is. I mean, like there's no good reason for a third-party cookie to follow you across domains, whereas it's necessary to specifically recognize fbclid in order to strip it from a URL. And it also wasn't clear that it's specifically Facebook fbclids, that is, on Facebook.com links. It looks like it's maybe any domain that has the misfortune of naming part of their URL tail fbclid.

So maybe this explains why Firefox is reportedly not doing as thorough a job as Brave is. They're not wanting to break things because this is going to tend to be a little bit flaky. This Olytics company embeds something called oly_enc_id= and then a big blob of crap. And then there's also oly_anon_id=. Drip, the company Drip has __s= and then some blob. Vero is vero_id. HubSpot uses _hsenc= and then something. Marketo, mkt_tok=. And Facebook actually has two, fbclid and also mc_eid. Anyway, okay. It's good, as I said, that Firefox is doing this. But stripping stuff from URLs is error prone, and to me it feels a bit messy. And if Facebook were to change the name of their fbclid parameter to fbclid2 or something, presumably that wouldn't match, and Firefox would let those go until they updated themselves. So it doesn't really grab me.

Okay. In any event, I'm still using Firefox, and I'm glad to have that incremental improvement. Since my Enhanced Tracking Protection is 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 itself to v102 last week. What's weird, however, is that until and unless an additional tweak is made, this tracking parameter stripping will not be enabled when using Firefox in its private browsing mode. It's like, what? Why would you not want it there? I don't get it. And that's even when Strict mode is enabled.

So Firefox-using listeners go to about:config. And about:config has a gazillion individual little things in there now. So search for the string "strip," S-T-R-I-P. And that'll reduce the number to something manageable, I don't know, looked like about eight, as I recall. You'll see one that's already enabled and set to "true," and that's privacy.query_stripping.enabled. And that was true probably because I'm in Strict browsing mode. And so that was probably - that's something that the Strict browsing mode setting turns on. But the one directly below it is privacy.query_stripping.enabled.pbmode, which is doubtless private browsing mode. It was not enabled. And it should be. So you want to double-click on that to flip it to true. It'll get bold, and then you should be good. And did you not find it? Oh, yeah, there it is, yeah. It's right above the cursor. So there's enabled, and the one below that, Leo, is enabled dot and then pbmode over - and now it's true? Yes, good. And that's what you want.

Leo: Just that one. I don't need to do the stripping enabler [crosstalk]?

Steve: Yeah. I wonder why - I had the one above it, stripping.enabled, the non-pbmode. Mine was already set to true because probably, I assumed, because I was in enhanced tracking protection as a main feature. You definitely want that, you do want that turned on.

Leo: They should all be true, in other words.

Steve: They should all be true. And just as an aside, because this is sort of a problematic thing, on the other hand I don't know what else we're going to do if we don't have our browsers stripping tracking crap from URLs because that's a thing. It could tend to break something. So if you turn this stuff on, and a website doesn't work the way you expect it to, then you might try turning it off briefly to see if that fixes the problem. Again, knowing Mozilla, they are tiptoeing very cautiously into this. I'm sure they didn't turn it on for everybody. I had it turned on for me because I was already in the highest level of strict browsing protection. I don't think they would turn it on if it was causing problems. So I'm sure it's been well vetted, and it's safe to enable. You just also want to enable that private browsing mode, as well.

Okay. One thing this podcast probably doesn't spend enough time focusing on, probably because we don't get enough information about it, is the very real threat from insiders gone bad. Naturally, nobody wants to advertise, no enterprise wants to advertise when they find someone being malicious inside their organization. They're quickly escorted out, and that's the end of it. So consequently it doesn't make the news.

Of course, 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 their badge, and then settles down in their cubicle, now with full internal access to the corporate network.

We've briefly touched on the notion that our government's own three-letter agencies might bring an existing employee of some other company or two who have critical access into their fold under the auspices of patriotism and national security. And certainly it's the case that private company employees were aware and stayed silent when the NSA was establishing those massive Internet listening post taps at the major domestic network exchange points. You know, that had to be known by the employees who worked in those facilities.

And we have deep suspicions that some of the older crypto designs - remember the DRB pseudo PRNG, that pseudorandom number generator that was like the weakest of all possible versions that could be chosen, and for some reason that was the RSA default, which always sort of raised some eyebrows. Which made us wonder whether maybe there was some influence under the guise of "This will be good for Uncle Sam." In these instances, people knew and were keeping quiet, it seems.

And then there's the insidious possibility that a truly malicious foreign agency might bribe someone, or perhaps extort them if there's some embarrassing weakness and say, okay, just wipe all of your fingerprints off of this little USB device that we're going to give you, then plug it into the back of the break room's printer and just walk away. We'll know when you have, and a parcel with $50,000 in crisp $100 bills will be delivered to you privately.

Leo: Awesome. Where do I get that job? I'd like to do that.

Steve: Just make sure you wipe your fingerprints off quite thoroughly.

Leo: Oh, yeah. Absolutely.

Steve: That's right. So yet another possibility is the self-corruption of a trusted employee who's able to subvert their employer's own business model 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 nearly at the same time by different researchers who just happen to be looking at the same place, 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 for a bounty is denied and paid instead to someone else. It can happen when no one is at fault. But to HackerOne's credit, they investigate all such reports. And in this case, once clear evidence of an insider gone bad was found, they went public with the details.

So here's what HackerOne's disclosure started out explaining. They said: "On June 22nd, 2022, a customer" - now, but when they say "customer," they mean someone who is using them to manage their company's bug bounty payments, payouts and so forth. "A customer asked us to investigate a suspicious vulnerability disclosure made outside of the HackerOne platform." Okay. So that meant that this customer was informed of a vulnerability, not through HackerOne, but through some outside agency.

"The submitter of this off-platform disclosure reportedly used intimidating language, as in like extorting this customer, in communication," they said, "with our customer. Additionally, the submitter's disclosure was similar to an existing disclosure previously submitted through HackerOne." They said: "Bug collisions and duplicates, where multiple security researchers independently discover a single vulnerability, commonly occur in bug bounty programs. 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 his 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."

Then they lay out a detailed nine-day blow-by-blow timeline starting June 22nd and finishing last Friday, the 1st of July. And I'm going to skip that. But I think that the discussion of their investigation would be interesting to our listeners. So they said: "Our investigation has concluded that a now former HackerOne employee improperly accessed vulnerability data of customers to resubmit duplicate vulnerabilities to those same customers for their 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 [what they described as] a contained group. These steps were necessary as we worked to investigate and eliminate the prospect of multiple insiders." Thus they immediately went to a constrained investigation, with only a few people being on the inside. They said: "We are now confident that this incident was limited to a single employee who improperly accessed information in clear violation of our values, culture, policies, and contracts." And then they repeated: "Within 24 hours of the tip from our customer, we took steps to terminate that employee's system access and remotely locked their laptop out 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 [what they described as a] 'sockpuppet account'" - meaning an alternative identity as if they were an external vulnerability researcher - "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 their 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've issued platform bans for the employee's known HackerOne accounts. We've terminated the employee for violating our values, culture, policies, and 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.'" That was the handle used by this guy, the sockpuppet account, "rzlr."

"The threat actor's motives appear to be purely financial in nature. The threat actor had access to HackerOne systems between April 4th and June 23rd of 2022." So not a long-term employee, somebody that they hired at the beginning of April who went south or sour pretty quickly. Maybe that was the entire intent of the employment was to get on the inside and then grab these things and try to get additional bounty payments.

So 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, it's clear that 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 being logged, an enterprise's network and the traffic it carries is not the sovereign property of its employees. It belongs to the corporation. So a casual reminder of the fact that what's done on the enterprise's network is being monitored can go a long way toward preventing such abuse from occurring.

And Leo, let's take a break. I'm going to prevent further abuse of my vocal cords by giving them a bit of hydration.

Leo: No hydration abuse here. No.

Steve: We're going to use some of our closing-the-loop feedback from our listeners to bring us into some interesting discussions.

Leo: Great. Awesome.

Steve: So someone on Twitter calling himself ReliefTwitcher, he said: "@SGgrc I thought of you today when I reached around to the back of my computer tower," he says, "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..."

Leo: Ah, didn't like it.

Steve: "...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." He said: "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 wrong way the first time anyway."

Leo: Wow.

Steve: I'm just sayin'.

Leo: There you go. Or it's Schrodinger's USB key. It could be that, as well.

Steve: That's right. Yes, actually we did have, I think I shared it a couple weeks ago...

Leo: Yeah, a couple weeks ago, yeah.

Steve: The notion of it existing in both states until you observe it.

Leo: Exactly.

Steve: And attempt to plug it in. And then it says, oh, no, no, no, no, not so fast.

So Stefan Bang, he said: "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 I also think I can fix reality for you," he said. "So regarding Operational Technology," he said, "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. Most protocols like Modbus over TCP or BACnet over IP are really old, from the '70s and '80s," he says, "RS-485 protocols that have been made to work over Ethernet.

"These protocols don't use any authentication, and BACnet/IP is even a UDP protocol. But they're 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're 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." He said: "The security of the supervisory system is easier to maintain as the server can auto update, and the firewall only needs to allow HTTPS traffic. A site-to-site 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."

He says: "I'll quote a wise man: 'Security is hard,' and 'Complexity is the enemy of security.'" He said: "By using a VPN, the only things that need to be secure is the VPN and the web server of the supervisory system. This is easier and much less complex than securing every single OT protocol, as there are a lot of them."

And then he just finished on his fixing reality for me. He said: "Regarding reality, I don't think the simulation is broken. I think the person who writes and maintains the simulation listens to Security Now!."

Leo: Oh, that would be a nightmare.

Steve: Oh, my god. "The reason why you need to turn the USB-A plug three to four 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."

Leo: Yeah?

Steve: So apparently, if that's the case, it is a Security Now! listening simulating overlord.

Leo: It's all about port-knocking.

Steve: Yeah, that's right. Okay. So 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. Now, I heard what Stefan said. I get it that in his mind sequestering a local or a private network behind a firewall or behind a VPN, or behind an HTTP-only, an HTTPS server is the solution. But those packets want to get out, Leo. They want to be free.

Leo: Data wants to be free, yeah.

Steve: Yes. And they're going to find a way, just like those Jurassic Park dinosaurs that were all female, they somehow - they reproduced anyway. Nature will find a way. So the problem is that the global insecure network is IP. While that pressure regulating valve was using the RS-485 protocol for its 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 that two wires rather than one were used, where the data was contained in the difference in the voltage on the two wires, rather than their absolute voltages.

This creates a great deal of environmental electrical and magnetic noise immunity which can be a problem in heavy industrial environments. And whereas RS-232 is a simple 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 a serial bus architecture. So it existed. It was secure. There's no way any, I mean, it didn't use packets. So they couldn't have escaped if they wanted to because there weren't any.

But the world back then was suddenly flooded with Ethernet cabling and Ethernet switches and routers whose cost dropped to near nothing as high manufacturing volumes and massive inter-vendor competition fought for market share. Wired Ethernet is also a highly 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 IP protocols, remember, what was it...

Leo: UDP? No.

Steve: No, I'm thinking that other - Novell. Novell had, you know, Novell Networking was on Ethernet. It just wasn't IP. So you could have had all of the advantages of all of those economies of scale without using a protocol that could escape. So other non-routable, non-IP protocols have been carried over Ethernet, and they would have been a far safer choice. But ultimately, being non-IP protocols, they too would have 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 completely overtaken the world. There was, I mean, I agree there was probably no way to resist the allure of that near zero cost.

Okay. But now our mission-critical, pressure-regulating valves are actually connected to, and potentially reachable from, Russia. As remember, as Khan was famously heard to say in the second Star Trek movie, when he was explaining why Kirk didn't raise his own ship's shields, he said, "We're all one big happy Federation," just before he blew the crap out of Kirk's ship.

Leo: Khan!!! Khan!!!

Steve: Now, with Operational Technology systems supporting IP protocols, we're all one big happy global network. What could possibly go wrong?

Leo: Yeah.

Steve: So there is a simple and elegant solution to this. I've mentioned this a few times in the past when its application has been appropriate. But it's just sort of like deriving one, like deriving all websites' passkeys from a single master key rather than just using random number generators. Yeah, okay. The same IP technology that creates this problem in the first place offers a solution. Operational Technology equipment, or for that matter any equipment that doesn't want or need to be reachable from across the globe, that is, that is designed for local network access only, 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 incoming TTL, its Time To Live value, which is contained in the header of that packet. If that TTL decrements to zero, meaning if it was one when it arrived at the router, the router decrements it, it hits zero, no router on the Internet will forward that packet. Period. Nothing would kill the Internet faster than packets which refused 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, just to be safe, will never affect any local networking use of such equipment, while at the same time flatly, elegantly, and completely preventing them from ever reaching foreign soil.

This use of deliberately short TTLs is something, as I said, that never seems to gain any traction, but it's a beautiful solution when adding another security failsafe to a system might be useful. Especially when you've got a system designed for local-only use, and its packets should never be allowed to escape. Right? In fact, we talked about this a long time ago because there was a time when TTLs were set to 32. Just sort of arbitrarily early on.

Leo: Is that seconds?

Steve: No. It's hops, 32 hops.

Leo: 32 hops.

Steve: So, for example, when we do a trace route, the way the trace route works is it deliberately sends out a packet with a TTL of 1, which is returned by the router that rejects it, and thus we get that router's IP. Then it sends a packet with a TTL of 2, so that first router says okay, fine, it's got some time left, and forwards it to the second router, which decrements the TTL from 1, which the first router set it to, to 0, which the second router sent it to. Now it dies at the second router, so that second router sends back a sorry, your packet died. So by successively emitting packets with greater and greater TTLs, we're able to trace or trace route the router-to-router hops that packets take.

And so it wasn't - there was a point during this podcast, because I remember we talked about it, where packets were just, you know, our operating systems were - the TCP/IP stacks set the IP TTLs to 32. And because there was just, you know, it was a nice number that engineers tend to use, 2^5. And that should be plenty; right? Then the Internet kept getting bigger and bigger. And more and more ISPs got sort of like shimmed in between others. And what happened was the so-called, the transit diameter of the Internet in some locations became greater than 32. There were two points on the Internet greater than 32 hops apart. And they were unreachable to each other.

And so it was necessary to go, oops, and just simply we doubled the TTL to 64, or in some cases it went to 127 or 255 because it's an eight-bit count. It doesn't take up very much space at the front of the packet, just a little eight-bit header, or eight-bit field, and it gets decremented. And so if these systems that never wanted to allow anyone in Russia or China to get a hold of them, even if the firewall fails, even if a flaw is found in the web server that they're hiding behind, then just emit packets with a TTL of 2 or 3, and they can't get out. They can't go far. But as far as I know it's never been done for security. And it would have been, you know, I'm a belt and suspenders person. And so it would make sense to do that.

Dr. Nathan P. Gibson, he sent another tweet that I thought was - actually it gave me a perfect segue. He said: "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." And I quoted the first of those two. He said: "When you talk about isolating your smart home network from your Internet network, it sounds like a lot of work. Are there are any secure smart home hubs that can do this for you? For example, you talk about," he says, "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." He says: "My Fritz!Box router actually has a setting for smart home devices, but I'm not really sure what it does."

Well, my feeling is that we're still on the "don't have it yet" side of the smart home 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, currently, 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 from this point forward a WiFi access point or router in the future be certain that it supports isolated guest WiFi networks. Just it's a terrific technology. And it's just trivial to set up. You just turn on a guest network, give it its own SSID and password, and then click a checkbox, typically, where you affirm that you do not want this network to have any access to the other LAN networks, WiFi or wired, and it becomes isolated.

Now, all that said, there is encouraging news on the IoT home 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" - that's CSA, 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..."

Leo: I know her.

Steve: Yes, we do, "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." He was writing this in August of 2021, so he's saying this year, 2022. "In a release announcing the delay, CSA (Connected Standards Alliance) CEO Tobin Richardson cited the 'need to get it right'" - words dear to me - "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 all of that.

"According to Stacey, CSA's CEO Tobin Richardson also blamed the resurgence of COVID, as well as the addition of nearly 30 [three zero] more companies to the Matter group. Formerly known as Project CHIP, which stood for Connected Home over IP, Matter is an IP-based protocol that's compatible with WiFi, Ethernet, and Thread. Matter has the backing of some of the biggest names in the smart home market, including Amazon, Google, Signify" - I didn't see Apple, but Apple's name is somewhere - "Signify is the owner of the Philips Hue smart lighting brand, and Samsung's SmartThings.

"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 also. Apple's HomePod mini comes with its own integrated Matter radio." So it looks like Apple's onboard, too. "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," he finishes, "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" - coddling? Oh, yeah, coddling - "our stubborn smart gadgets..."

Leo: You wrote it, dude.

Steve: "...through the end of the year, and probably even longer." Oh, I did also note that DigiCert and Wemo's names were also associated with Matter. And Wikipedia just had this to say.

They said: "Matter, formerly Project Connected Home over IP (CHIP), is a royalty-free home automation connectivity standard, with manufacturers only incurring certification costs." And so using it costs nothing, thank god. And that gives it a chance to be the chosen solution. "Announced on December 18th, 2019, Matter aims to reduce fragmentation across different vendors and achieve interoperability among smart home devices and Internet of Things platforms from different providers."

Wikipedia says: "The project group was launched and introduced by Amazon, Apple, Google, Comcast, and the Zigbee Alliance, now called the Connectivity Standards Alliance (CSA)." And then they said: "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 the CSA."

So the Connectivity Standards Alliance, for anyone who is interested, is at csa-iot.org. Again, csa-iot.org. And this couldn't be better news. It sounds like we'll be needing to do a podcast on this soon. And of course I won't be able to resist titling it "What's the Matter?"

Leo: Ohhh. We've been talking about Matter for quite some time. Of course Stacey is all over it.

Steve: Yup.

Leo: But this is good. I'm glad they're starting to make some progress instead of just being empty suits.

Steve: And a hope, yes.

Leo: Yeah.

Steve: So that's really great. I mean, basically that's what we've been waiting for. I would not move to a single, I mean, a hub-based solution is going to be better because it's going to integrate much more tightly with your other devices, and potentially solve the security problem. I mean, it is nerve wracking, knowing that my individual light switches and smart plugs each establish a persistent connection to China. That's just like, you know, that's not what you want. So having them instead connect to a centralized hub from a security-conscious company like Apple, Google, or Amazon, that's a way better solution. So looks like that's going to happen soon. And couldn't happen soon enough. Yay.

Ivan Neeson, he tweeted. He said: "Hey, Steve. I have a question about passkeys. 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 log on? So," he says, "so if someone hacks the PC they won't get any passkeys?"

And Ivan, that's a good point, and it's potentially true. I'm not sure how I'll come down on this issue. At the moment, for example, I have all of my TOTP second-factor time-varying 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 for me takes little 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 back then a novel concept. And of course we later renamed it Secure Quick Reliable Login as its application space expanded. And we saw something like that in Apple's presentation, though all the mechanics were not exactly clear.

So if it's super easy to present a website's FIDO2 QR code, whatever it is that it's showing, to a smartphone, to then have the smartphone authenticate the user, that indeed might be sufficient and, for many people, ideal. And perhaps we, for example, only have our PCs storing passkeys that are less critical, and keep like our banking passkeys on a non-shared environment. So anyway, we don't know yet. We'll need to see how this all shakes out. And the good news is we'll probably have a couple years to wait for all that to happen.

Jonathan tweeted. He notes that math geeks make difficult customers, and he pointed me to a tweet thread. Tweet thread read: "I ordered a 9" personal pizza. After a while, the waiter brought two 5" pizzas and said the 9" pizza was not available, so he was giving me two 5" pizzas instead, and that I was getting 1" more for free.

Leo: Five plus five equals 10.

Steve: That's right, baby. Of course we know where this is going.

Leo: Yes.

Steve: The person tweeting this said: "I requested the waiter to call the owner over. I showed the owner the mathematical formula to calculate the area of a circle." As we know, the area of a circle is pi times r squared. And you don't actually need to know about pi. You could sort of throw that out; right? You could just use r squared. But this tweet thread says: "So the area of a 9" pizza is 63.62 square inches, whereas a 5" pizza has only 19.63 square inches. The two 5" pizzas together add up to 39.26 square inches. I said to the owner that even if he gave me three 5" pizzas I would still be losing out.

Leo: Ah, this guy is absolutely math literate. I love it. Not the owner, obviously.

Steve: Right. Or the server. "How can you say you're giving me an extra inch for free?"

Leo: I bet it fools everybody except this guy. Absolutely.

Steve: Yeah. "The owner was speechless," he says, "and gave me four 5" pizzas."

Leo: Happy ending.

Steve: Happy ending indeed.

Leo: That's hysterical. That's great.

Steve: And really, as we know, you can do the math easily in your head because you could ignore the pi and just do r squared. Because that's all you really care about.

Leo: That's right.

Steve: Exactly. Although you could say that pizza is a pi.

Leo: Unh-huh.

Steve: So there is that.

Leo: And, well, yeah. You know where we're headed.

Steve: Okay. We're heading to our final sponsor, and then we're going to talk about the hard-to-pronounce...

Leo: ZuoRAT.

Steve: ZuoRAT. Last Tuesday, Black Lotus Labs, which is the threat intelligence arm of Lumen Technologies, which was formerly known as CenturyLink, they've been tracking elements of what they say appears to be a sophisticated campaign leveraging infected SOHO (small office/home office) routers to target predominantly North American and European networks of interest. So generically RATs (Remote Access Trojans) are a dime a dozen and wouldn't usually command much of this podcast's prolonged attention.

But in one of those kind of "It's obvious after the fact that this would happen" disclosures, Black Lotus Labs revealed that they had uncovered a complex campaign that had gone undetected for nearly two years, and that the trigger for the campaign was apparently 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 secured remote employee networks. And from there they pivot onto the corporate LAN.

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've enumerated some of the command-and-control infrastructure associated with this activity and identified some of the targets. We assess with high confidence the elements we're tracking are part of a broader campaign."

Okay. So some powers of darkness realized early in the COVID lockdown that large numbers of employees of North American and European enterprises would begin working from home, and that many of those employees would be inherently weakening their employers' security by effectively extending their previously well-curated and secured network out into the periphery. 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 workforce migration to home was going to be straining enterprise security. Obviously, this occurred to others, as well.

ZuoRAT may not have been explicitly and expressly created to fulfill this agenda, but it appears to have been perfectly designed for this role. The campaign consists of several components. There's the first stage RAT that was developed specifically for SOHO routers. We'll come back to take a closer look at that in a minute. There's a simple loader for Windows machines which was compiled in C++. And then, third, there are three separate fully functional agents, which the RAT and the Windows loader, which is just a simple remote file retriever, downloads. There are three fully functional agents, two of which were custom developed, and the one that isn't is the well-known Cobalt Strike Beacon which we previously covered in some detail. Together and independently, these full-function agents allow for full enumeration of the infected device, downloading and uploading of files, network communication hijacking, process injection, and more. And I'll go into those also in some detail in a minute.

So this ZuoRAT is a MIPS chip file compiled for the MIPS chip which will run on routers from ASUS, Cisco, DrayTek, NETGEAR, and others. That executable can enumerate a host and the 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 and actually softly defined 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 the 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 mischief and 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 remain rare. This is what they wrote. They said: "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."

Now, notice that when we talk about certificate spoofing, one of the concerns is that our computers now trust so many certificate authorities, including foreign actor, like foreign state certificate authorities. So the reason we're not that worried is that the IP, we assume that the IP address we are getting from DNS is correct. So even though a foreign actor, a foreign state could produce a certificate for some high-profile website, like say Facebook, or a bank, our traffic will go to the bank. So it doesn't matter, I mean, we're not going to believe, we're still going to get the bank's certificate.

But if you combine the ability of a high-level state actor to create certificates, with the ability for them to change the IP address that your DNS lookups return, which is exactly what this thing does, now you think you're connecting to your bank. You're instead connecting to a server elsewhere which has a certificate not issued for the bank, but issued under the bank's name for a foreign actor. And so your browser's happy, no alerts are shown, and you're completely compromised. Everything you do on that site is decrypted at that location and can be relayed anywhere. So it's a serious attack, when you get somebody who has the ability, either directly or indirectly, to get certificates, who's also able to redirect traffic.

They said the Windows loader that was 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 those three functionally second-stage agents, and the one that was chosen depended upon the environment. There was something known as CBeacon, which was a customized developed RAT written in C++, which had the ability to upload and download files, run arbitrary commands on the machine where it was installed, and persist on the infected machine through a component object model (COM) hijacking method. Well, the fact that it's COM means that it's Windows only, and CBeacon was Windows only.

However, GoBeacon, the second of those three, is 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. So that one was used to infect non-Windows Linux and macOS.

And then, finally, the third one, Cobalt Strike. They said: "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 is an internal Microsoft development term - "some of which contained Chinese characters, while others referenced a possible name or Chinese locality." So strong evidence that this was Chinese in origin. Although you could have a false-flag operation, too.

"Additionally, there was a second set of actor-controlled command-and-control infrastructure used to interact with the Windows RATs that was hosted on Internet services from China-based organizations Alibaba and Tencent. Given the age of the first observed router sample, which was first submitted to VirusTotal in December of 2020, as well as a sampling from Black Lotus Labs telemetry over a period of nine months," they said, "we estimate this multiyear campaign has impacted at least 80 [eight zero] specific targets, likely many more."

And that was one of the problems was they recognized that due to the structure of the system, they didn't have visibility into everything that was going on. These are individual SOHO routers, so how are they ever going to see what's going on? Well, one of the things they could do because they are Lumen, which is CenturyLink, thus a Tier 1 backbone provider, is they can look at the traffic that at least their own backbone is carrying, which will certainly - it's far short of the global Internet, but it's a big chunk of smaller ISPs who buy their bandwidth from CenturyLink, now Lumen. So, for example, if they see one compromised SOHO router connecting to a specific command-and-control server at a given IP, they can then look for all other traffic connecting to that IP and see what they can learn from it. So they are in a privileged location by being such a large Tier 1 provider.

They said, or actually I said, that during their investigation of ZuoRAT's activity, they observed telemetry indicating infections stemming from numerous SOHO router manufacturers, including, as I noted, ASUS, Cisco, DrayTek, and NETGEAR. But they were less lucky in capturing any actual running code on those. They were only able to obtain the exploit script for a model Q20 of a router manufactured by a company JCG. So it was the JCG-Q20. 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, a Python script. So I looked, and I found that Python proof of concept over on GitHub from before it had been weaponized and it obtained the credentials and - well, it was weaponized to then obtain credentials and to load ZuoRAT.

The weaponized script first performed a command line injection to obtain authentication material. Then it used the output from the command line injection, namely that authentication material, to authenticate itself in order to get the access that it was looking for. So this chain of vulnerabilities allowed the actor to download a binary onto that router, then execute it in order to gain the access they wanted. The script that they were able to obtain contained four functions. There was one called "getpasswd" which sent a specifically formatted request to the remote host requesting its password. There was "getloginsysauth," "execCmd," and "telnet." Basically a little toolkit providing everything that they needed.

The final stage of the exploit script downloaded the ZuoRAT agent framework. That framework enables in-depth reconnaissance of target networks, traffic collection, and network communication hijacking. And that can be divided into two components. The first contains functions that would auto-run upon execution of the file. The second component contains functions, and 2,500 of them believe it or not, 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. And they wondered why are some active and some appear to just be along for the ride. It appears that this ZuoRAT was a heavily modified version of the Mirai malware, which of course was a strain of ransomware that we've talked about before.

So the core functionality of that first component. It 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 the command-and-control servers. Black Lotus believed that its purpose was to acclimate the threat actor to the targeted router they'd landed on, as well as the contents of the router-adjacent LAN to determine whether or not there's anything worthy of further exploitation. They don't know anything about the router that they've managed to get into. They've got to look around and decide whether they want to spend any more time there.

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 in memory such as credentials, routing tables, IP tables, and so forth. The file was initially executed by the threat actor via the command line, specifying an IP address and port for the command-and-control node. So it would execute the command and say, send this package of information to this IP and port. If the IP and port was not provided in the exploit script, the ZuoRAT code contained a default command-and-control hostname listed as cs.memthree.com, which was a domain originally purchased in October of 2020. So coincident with the first development of all this.

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 command-and-control server. It also attempted to gather the router's public IP address by querying the following web services. It would query whatismyip.akamai.com, ident.me, myipdnsomatic.com, and ipecho.net. If ZuoRAT was unable to obtain a public IP address, it would immediately delete itself and terminate under the assumption that it was being run in an isolated sandbox, probably being analyzed.

Next, ZuoRAT would connect to the command-and-control server and attempt to bind a "listen" on port 48101. If the bind failed because the port was already in use, it would also immediately terminate the current process to ensure that only a single instance of the Trojan was running on the compromised device because you're not allowed to locally bind two different applications to be listening on the same port because it's the binding that decides when a packet comes in which process will receive a notification and access to that packet.

ZuoRAT then used a scan function designed to survey the adjacent LAN's internal IP addresses. So it would do an internal scan of the entire behind-the-router LAN. It scanned for a hardcoded list of open listening ports including 21, 22, 23, 80, 135, 139, 443, 445, 808, 902, 912, 1723, 2323, 3306, 5222, 5269, 5280, 5357, 8080, 8443, and 9001. And I know that many of those ports are familiar to our listeners as they are to me. It then performs an internal LAN scan of all machines on the LAN across all the IPs.

So in doing so, it's on the trusted internal network. It's inherently trusted by Windows firewall on those machines and able to bridge traffic from the external command-and-control to anywhere within the network, giving this thing a lot of power.

And once that's done, ZuoRAT would send the reconnaissance information it had obtained to the previously supplied command-and-control. If the connection was being established for the first time, it would occur over port 55556. If the connection was being refreshed, communication switched to port 39500. And I have no idea why. If the connection was successful, data would be transmitted. If errors were returned, the program would sleep and then repeat the attempt in a loop.

And finally, in preparation for establishing network capture capabilities, ZuoRAT allocated memory to itself for increased performance in order to have room to capture packets, and also allocated a mutex semaphore to ensure that only one instance would be running at a time. 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.

So anyway, we have a breathtakingly sophisticated and serious remote access trojan which is designed to avoid detection. In their write-up they also talked about something that we had talked about before, how rather than connecting directly to command-and-control servers, they also often bounced through innocent routers that were being used as simple IP proxies. Routers that have universal plug-and-play ports open are able to be asked to serve as public routers. And so the people behind this know where those routers are, and they're able to set them up to bounce traffic through the router, making it much more difficult to track down the location of the command-and-control servers. So they are being very deliberately and carefully protected.

They said, based on their telemetry, they observed 23 devices maintaining a persistent connection to a single command-and-control server during September and October of 2021. All 23 devices were located in the U.S. and Canada. The majority of IP addresses communicated with the command-and-control server 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's RV 320, Cisco's RV 325, and Cisco's RV 420; the Asus RT-AC68U, the Asus RT-AC530, the RT-AC68P, and the RT-AC1900U; also a DrayTek Vigor 3900 and an unspecified bunch 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 I noted above, subsequent connections were made to port 39500.

They said more recently they've seen activity from a separate command-and-control located at 103.140.187.131 on port 6666. That was occurring from February 22nd of 2022 through May 16th of 2022, and that's been acting similarly. So anyway, the Black Lotus report contained much more detail, but I'm sure that everyone now has a good idea how much work someone, apparently in China, has put into building, deploying, and maintaining a powerful network intrusion capability. And it's not just theoretical. It is in active use.

In summarizing what they had done, 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."

Leo: What? Like the Unabomber? I mean, not physically living on the edge. He's just - he's out there somewhere. Probably China; right?

Steve: Well, they're on the edge of targeted networks.

Leo: Right.

Steve: So they are in the SOHO routers of employees of major networks, and they are using that location to get into major enterprise networks.

Leo: I get it. I get it, yeah.

Steve: So it's a stepping-stone.

Leo: Yeah.

Steve: And it actually exists.

Leo: Just like a sewer rat.

Steve: Just like a sewer rat, Leo.

Leo: Living on the edge. Oh, wow. All right. I guess what can you do? What can we do? We can update our routers, I guess.

Steve: Yes. I would say make sure that your routers are running the most recent firmware, and reflashing the firmware when you know you've got good firmware. That will clear them out of the file system. And then just, you know, make sure that your perimeter security is up. Turn off any, you know, Universal Plug and Play should never be bound to the WAN port, and remote management should never be turned on. Just find some other way to do it.

Leo: Yeah. Well, it's a never-ending litany of threats, and that's why we love him. Steve Gibson, GRC.com. If it weren't for him, you might not even know about it; right? So that's why you've got to listen every single week. You can get a copy of this show at Steve's site. In fact, he's got some unique copies, a 16Kb version for the bandwidth impaired. He also has the transcripts, handcrafted by Elaine Farris. Those are great for not only searching, but reading along while you listen. All of that plus 64Kb audio available at GRC.com.

While you're there, pick up SpinRite. Not too late to get v6 of the world's best mass storage maintenance and recovery utility. If you have drives, you need SpinRite. Works on SSDs as well as hard drives. 6.1 is coming soon. And anybody who buys 6.0 now will get 6.1 as a free upgrade when that happens.

Steve: Yup.

Leo: There's also lots of other stuff there that Steve puts up for free. It's a really good site to browse around: GRC.com.

Steve: Does rathole, does that term work?

Leo: Yeah, yeah. It's a rathole.

Steve: It's a rathole.

Leo: We even have a rathole jingle somewhere which I will not take your time up.

Steve: Thank you.

Leo: Digging up. You can also get copies of this show at our website, TWiT.tv/sn for Security Now!. There's a YouTube channel dedicated to Security Now! so you can watch it there, share it from there, which is a good way to share the video if you want to share it with somebody. And I know a lot of times there's stuff in here you want to tell people about, raise the alarum, as it were. You can also subscribe in your favorite podcast player. That's probably the best thing to do. That way you'll get it automatically the minute it's available. You won't miss an episode at all, ever, that way. And you can also leave us a five-star review that way, which is very much appreciated.

We record Security Now! on Tuesdays at about 1:30 Pacific, 4:30 Eastern, 20:30 UTC. You can watch us do it live, if you wish, or listen live at live.twit.tv if you want the very freshest version of Security Now!. If you're watching or listening live, chat live at irc.twit.tv, or in our Discord chatroom open to Club TWiT members. Actually, Club TWiT's a great thing to get because you get ad-free versions of all the shows. You get the Discord. You get the TWiT+ feed. A mere seven bucks a month. Just go to TWiT.tv/clubtwit if you're interested. Thanks in advance. We really appreciate your support.