Condi DDoS Botnet Spreads via TP-Link's CVE-2023-1389 | FortiGuard Labs

By Joie Salvio and Roy Tay

Published: 2023-06-20 · Archived: 2026-04-05 23:03:16 UTC

Affected platforms: Linux
Impacted parties: Any organization
Impact: Remote attackers gain control of the vulnerable systems
Severity level: Critical

FortiGuard Labs encountered recent samples of a DDoS-as-a-service botnet calling itself Condi. It attempted to spread by exploiting TP-Link Archer AX21 (AX1800) routers vulnerable to CVE-2023-1389, which was disclosed in mid-March of this year. We have additionally observed an increasing number of Condi samples collected from our monitoring systems since the end of May 2023, indicating an active attempt to expand the botnet.

This blog details the capabilities of this botnet.

Condi Botnet: Buy or Rent

While pivoting from the Command and Control (C2) domain cdn2[.]duc3k[.]com in one of the malware samples, FortiGuard Labs researchers found a sibling domain admin[.]duc3k[.]com that previously displayed the message "contact @zxcr9999 telegram”. A quick search revealed a Telegram channel, Condi Network, advertising a Condi botnet with capabilities matching those observed in our sample (Figure 1).

The Telegram channel was started in May 2022, and the threat actor has been monetizing its botnet by providing DDoS-as-a-service and selling the malware source code (Figure 2).

We provide a technical analysis of the ARM malware sample *509f5bb6bcc0f2da762847364f7c433d1179fb2b2f4828eefb30828c485a3084* in the following sections:

Killing off the Competition

This malware employs several techniques to keep itself running in an infected system. At the same time, it also prevents infections from other botnets by attempting to terminate their processes.

Typical to Mirai-based botnets, this malware cannot survive a system reboot. Because of this, it deletes the following binaries used to shut down or reboot the system.

- /usr/sbin/reboot
- /usr/bin/reboot
- /usr/sbin/shutdown
- /usr/bin/shutdown

- /usr/sbin/poweroff
- /usr/bin/poweroff
- /usr/sbin/halt
- /usr/bin/halt

It also reads the /proc/<PID>/status for each running process and compares the Name field to the following strings to kill any processes with matching names:

- /bin/busybox
- /bin/systemd
- /usr/bin
- test
- /tmp/condi
- /tmp/zxcr9999
- /tmp/condinetwork
- /var/condibot
- /var/zxcr9999
- /var/CondiBot
- /var/condinet
- /bin/watchdog

We assess that the developer intended to kill off older versions of Condi currently running on an infected device together with selected system processes. However, the implementation is flawed as the Name field only contains the executable names of processes and not their full paths.

Additionally, it kills any processes with binary filenames containing the following extensions commonly used by other botnets:

- x86
- x86_64
- arm
- arm5
- arm6
- arm7
- mips
- mipsel
- sh4
- ppc

It also generates a random string of at least ten characters from the custom alphanumeric character set "lvrvup9w0zwi6nuqf0kilumln8ox5vgv@" and attempts to kill any process with this string in its command line, however it is near certain this process will not exist. Which process the malware developer intended to terminate with this code is unclear.

Finally, it generates two numbers (one between 12 and 32, the other between 12 and 20) and kills any processes with a command line length matching either number. Killing off random processes based on their command line length is likely to wreak havoc and prevent the infected device from functioning correctly if the malware happens to terminate system processes.

Botnet Propagation

Unlike most DDoS botnets, this sample does not propagate by trying different credentials. Instead, it embeds a simple scanner modified from Mirai's original Telnet scanner to scan for any public IPs with open ports 80 or 8080 (commonly used for HTTP servers) and then sends a hardcoded exploitation request (Figure 3) to download and execute a remote shell script at hxxp://cdn2[.]duc3k[.]com/t, which will infect the device with Condi if it is a vulnerable TP-Link Archer AX21 device.

The remote shell script is typical of Mirai-based loaders that try to download and execute binaries of each architecture in turn (Figure 4). The first command-line argument provided to the malware binary, "0days", in this case, is referred to as "id" ("source" in the original Mirai code), which DDoS botnet operators commonly use to identify the method used to replicate the malware.

While the sample we analyzed only contained the scanner for CVE-2023-1389, other Condi botnet samples were also seen exploiting other vulnerabilities to propagate. The publicly available source code for older versions also includes scanners for known vulnerabilities exploited by other Mirai variants.

We also observed shell scripts hosted on the same IP with different sources in the execution commands. Figure 5 shows a script with an "adb" source, which refers to Android Debug Bridge (ADB).

We found source code for an older version of Condi that scans for devices with an open Android Debug Bridge port (TCP/5555), so it is possible that the botnet is currently being propagated via this means.

C2 Protocol and Command List

The binary protocol used by Condi to communicate with the C2 server is a modified version of that initially implemented in Mirai.

The initial registration packet sent by the bot to the C2 contains the bytes *\x33\x66\x99*, commonly associated with Moobot, another Mirai variant. These bytes are followed by a one-byte length of the "id". In the case of Condi, "id" defaults to "c" if none was specified, or in our case, of an infection via CVE-2023-1389, "0days". This signals the C2 server that the malware is ready to receive commands.

The first three bytes of the C2 response indicate the command for the Condi bot:

1. *\x99\x66\x33*: Likely to check if the malware is still active, in which case the malware sends a packet to C2 with \x66\x99\x66\x04 followed by "ping"
2. *\x99\x66\x66*: Terminate the bot
3. *\x33\x66\x66*: Start the webserver for serving malware binaries

4. *\x33\x66\x33*: Update binaries served by the webserver

5. *\x33\x66\x99*: Send the webserver port. Malware responds with \x66\x99\x66 followed by a length of the next string and “CondiiNeett webserv:<PORT>"

6. *\x66\x66\x99*: Sets an unused *lockdown* flag, which might indicate a feature in development.

Once it receives the *\x33\x66\x66* command used to start the webserver, this malware downloads bot binaries from a hardcoded IP and port. After that, it starts a basic HTTP server on a random port number above 1024 to host these binaries. GET, POST, and HEAD requests to this server for the */arm*, */arm7*, */mips*, */mipsel*, */x86_64*, */sh4*, */ppc*, and */m68k* URLs will serve these binaries if they were downloaded previously. This HTTP server masquerades as a legitimate Apache HTTP server by responding with the “Server: Apache” header when any URLs are requested.

From then on, the threat actor can issue the *\x33\x66\x33* command to download the latest binaries from the same hardcoded IP and port so that the webserver serves the most updated version of the malware.

If the first byte of the C2 response is not *\x33*, *\x66*, or *\x99*, the bot parses it as an attack command in the same way as Mirai.

Below is this sample's list of attack functions and a description of the implemented attack method.

- *attack_tcp_syn*: Similar to Mirai’s TCP SYN flood
- *attack_tcp_ack*: Similar to Mirai’s TCP ACK flood
- *attack_tcp_socket*: TCP flood using 5000 threads against a single targeted IP
- *attack_tcp_thread*: TCP flood using 100 threads shared among targeted IPs
- *attack_tcp_bypass*: Similar to Mirai’s TCP STOMP flood
- *attack_udp_plain*: Similar to Mirai’s UDP PLAIN flood
- *attack_udp_thread*: Similar to *attack_udp_plain*, but uses two threads per target IP
- *attack_udp_smart*: Similar to *attack_udp_plain* with extra error handling for connection failures

As the attack methods are consistent with the descriptions in the Telegram advertisement (Figure 1), this particular sample was likely built by the bot developer or someone with access to the malware source code.

This sample did not contain any HTTP attack methods observed in older Condi versions.

Conclusion

Malware campaigns, especially botnets, are always looking for ways to expand. Exploiting recently discovered (or published) vulnerabilities has always been one of their favored methods, as we highlighted above for the Condi botnet. Thus, it is strongly recommended to always apply the latest security patches and updates as soon as possible.

As always, FortiGuard Labs will continue to monitor these campaigns.

Fortinet Protections

Fortinet customers are already protected from this malware through FortiGuard's Web Filtering, AntiVirus, FortiClient, and FortiEDR services, as follows:

The following (AV) signature detects the malware samples mentioned in this blog:

- **Linux/Mirai.REAL!tr**
- **Linux/Mirai.CDB!tr**

The FortiGuard AntiVirus service is supported by FortiGate, FortiClient, and FortiEDR. Fortinet EPP customers running current AntiVirus updates are also protected.

The FortiGuard Web Filtering Service blocks the C2 servers and download URLs.

FortiGuard Labs provides IPS signatures against attacks exploiting the following vulnerability:

- CVE-2023-1389: TP-Link.Archer.AX21.Unauthenticated.Command.Injection

For a comprehensive list of protections from FortiGuard Labs for this vulnerability, please visit the Outbreak Alert page for further details.

The FortiGuard IP Reputation and Anti-Botnet Security Service proactively blocks these attacks by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our Global FortiGuard Incident Response Team.

IOCs

Files

091d1aca4fcd399102610265a57f5a6016f06b1947f86382a2bf2a668912554f
291e6383284d38f958fb90d56780536b03bcc321f1177713d3834495f64a3144
449ad6e25b703b85fb0849a234cbb62770653e6518cf1584a94a52cca31b1190
4e3fa5fa2dcc6328c71fed84c9d18dfdbd34f8688c6bee1526fd22ee1d749e5a
509f5bb6bcc0f2da762847364f7c433d1179fb2b2f4828eefb30828c485a3084
593e75b5809591469dbf57a7f76f93cb256471d89267c3800f855cabefe49315
5e841db73f5faefe97e38c131433689cb2df6f024466081f26c07c4901fdf612
cbff9c7b5eea051188cfd0c47bd7f5fe51983fba0b237f400522f22ab91d2772
ccda8a68a412eb1bc468e82dda12eb9a7c9d186fabf0bbdc3f24cd0fb20458cc
e7a4aae413d4742d9c0e25066997153b844789a1409fd0aecce8cc6868729a15
f7fb5f3dc06aebcb56f7a9550b005c2c4fc6b2e2a50430d64389914f882d67cf

Download URLs

hxxp://85[.]217[.]144[.]35/arm
hxxp://85[.]217[.]144[.]35/arm5
hxxp://85[.]217[.]144[.]35/arm6
hxxp://85[.]217[.]144[.]35/arm7
hxxp://85[.]217[.]144[.]35/m68k
hxxp://85[.]217[.]144[.]35/mips
hxxp://85[.]217[.]144[.]35/mpsl
hxxp://85[.]217[.]144[.]35/ppc
hxxp://85[.]217[.]144[.]35/sh4
hxxp://85[.]217[.]144[.]35/x86
hxxp://85[.]217[.]144[.]35/x86_64
hxxp://85[.]217[.]144[.]35/abc3.sh
hxxp://cdn2[.]duc3k[.]com/t

C2s

85[.]217[.]144[.]35
cdn2[.]duc3k[.]com
