INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:35 UTC

BPFdoor Upgraded: Red Menshen Hardens Its Telecom Espionage Implant Against Detection

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0115
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Linux-based systems with BPF-capable kernels; global telecommunications providers
Published	2026-03-27
Discovery Source	Rss

Executive Summary

Red Menshen, a Chinese state-sponsored threat group, has deployed an upgraded variant of BPFdoor, a kernel-level Linux backdoor, against telecommunications providers globally. The implant operates below the application layer, bypasses firewalls and port scanners without binding to any listening port, and activates only on receipt of a covert trigger packet, making it invisible to most conventional security controls. Organizations transiting sensitive communications data through telecom infrastructure face a high risk of long-term, undetected espionage.

Technical Analysis

BPFdoor is a passive, portless Linux backdoor that hooks into the kernel's Berkeley Packet Filter (BPF) subsystem to intercept network traffic before firewall rules are applied. The upgraded variant attributed to Red Menshen (also tracked as Earth Bluecrow) hardens the implant against behavioral and signature-based detection by operating entirely within kernel space, leaving no open ports visible to netstat, ss, or external scanners. Activation requires a specially crafted 'magic packet', a traffic-based trigger the implant monitors passively, which initiates a reverse shell or command channel. No discrete vulnerability is exploited; BPFdoor abuses legitimate OS functionality (BPF hooks, raw socket access), placing it outside the CVE model because CVEs require identifiable flaws, not misuse of intended features. This is categorized as a rootkit technique (T1014) rather than a vulnerability. Relevant CWEs: CWE-284 (Improper Access Control) and CWE-269 (Improper Privilege Management), reflecting abuse of privileged kernel interfaces. MITRE ATT&CK coverage includes T1205.002 (Traffic Signaling: Socket Filters), T1014 (Rootkit), T1543 (Create or Modify System Process), T1059.004 (Unix Shell), T1071 (Application Layer Protocol), and T1049 (System Network Connections Discovery), among others. Affected systems: Linux hosts with BPF-capable kernels, effectively all

modern Linux distributions running on telecom infrastructure. There is no vendor patch; the implant abuses legitimate kernel functionality. Defensive posture depends entirely on proactive detection and kernel-level monitoring.

Action Checklist

1. Step 1: Containment, Identify Linux hosts on telecom backbone and carrier-grade infrastructure with BPF-capable kernels. Restrict CAP_NET_ADMIN and CAP_SYS_ADMIN capabilities to explicitly authorized processes using capability bounding sets. Where possible, enforce seccomp profiles that block bpf() syscall for non-privileged workloads. Isolate any host exhibiting anomalous raw socket activity or unexpected BPF program loads pending investigation.
2. Step 2: Detection, Audit loaded BPF programs on all Linux hosts using 'bpftool prog list' and 'bpftool map list'; flag any programs not associated with known, authorized network monitoring agents. Note: bpftool availability varies across distributions. On systems without bpftool, rely on auditd logging of bpf() syscalls (audit rule: -a always,exit -F arch=b64 -S bpf -k bpf_monitoring) and kernel log inspection (dmesg | grep bpf) as alternatives. Review /proc/net/packet and /proc/net/raw for unexpected raw socket holders. Inspect kernel logs (dmesg, /var/log/kern.log) and auditd records for bpf() syscall invocations from non-standard processes. Hunt for processes with no associated listening port that nonetheless receive inbound packets, BPFdoor's defining behavioral signature. Cross-reference against MITRE T1205.002 and T1014 hunting logic.
3. Step 3: Eradication, No vendor patch exists; eradication requires host reimaging after confirming compromise. Before reimaging, collect a full memory image and disk forensic snapshot for incident analysis. Remove any unauthorized BPF programs identified during detection using 'bpftool prog unload' as an interim measure only, this does not guarantee full removal if persistence mechanisms (T1543.002: Systemd Service) are present. Audit systemd unit files, init scripts, and cron jobs for entries referencing unfamiliar binaries or paths.
4. Step 4: Recovery, After reimaging, validate that BPF program inventory returns to known-good baseline using 'bpftool prog list'. Reconfirm that no raw sockets are held by unauthorized processes. Restore systems from verified clean backups only. Deploy continuous kernel telemetry (e.g., Falco, Tracee, or eBPF-based EDR) to monitor for BPF program loads, raw socket creation, and capability abuse going forward. Monitor outbound traffic patterns from restored hosts for anomalous application-layer communication consistent with T1071.
5. Step 5: Post-Incident, This campaign exposes gaps in kernel-level visibility, BPF program inventory controls, and Linux privilege management. Conduct a capability audit across all Linux infrastructure, CAP_NET_ADMIN and CAP_SYS_ADMIN should be explicitly granted, logged, and reviewed. Evaluate deployment of eBPF-based runtime security tools that can observe kernel activity in real time. Map detection coverage against MITRE T1205.002, T1014, T1543, and T1059.004 to identify gaps in current SIEM and EDR rules. Engage threat hunting cadence specifically targeting passive backdoor indicators on high-value Linux hosts.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate immediately to CISO, legal counsel, and relevant national CERT (e.g., CISA for US-nexus carriers) if any BPFdoor indicator is confirmed on a host transiting subscriber call records, lawful intercept infrastructure, or inter-carrier signaling (SS7/Diameter) — presence on telecom backbone infrastructure constitutes a potential CALEA compliance event and may trigger mandatory reporting obligations under FCC breach notification rules (47 CFR Part 64) and applicable national telecommunications security regulations.
Recovery Notes	Restored hosts must not be returned to production telecom backbone roles until a minimum 72-hour clean observation window is completed with Falco or Tracee actively monitoring for bpf() syscall activity and raw socket creation, with alerts confirmed functioning via a test rule. Re-compromise risk is elevated given Red Menshen's demonstrated pattern of re-targeting previously identified victims; treat initial access vector (credential theft, supply chain, or vulnerable management interface) as unresolved until root cause analysis is complete. Maintain packet capture on restored host egress for 30 days and retain all auditd and kernel logs for a minimum of 12 months to support potential regulatory review.
Forensic Artifacts	BPF program inventory snapshot: output of 'bpftool prog list --json' capturing any SOCKET_FILTER type programs not associated with authorized monitoring agents — BPFdoor attaches as a SOCKET_FILTER to intercept packets matching its magic trigger value without binding a port, leaving this as the primary host-based forensic indicator. | Raw socket holder records: contents of /proc/net/packet and /proc/net/raw at time of detection, correlated to PIDs via 'ls -la /proc/[PID]/fd | grep socket' — BPFdoor's passive listener holds a raw socket with no associated bound port, which is anomalous for legitimate telecom host processes. | LiME memory image: full RAM capture taken before any eradication action — BPFdoor's BPF bytecode, its magic packet trigger value (used to activate the implant), encryption keys, and in-memory C2 configuration are stored only in kernel memory and will not survive a reboot or process termination. | Auditd bpf() syscall log: /var/log/audit/audit.log entries matching the 'bpf_syscall_monitor' auditd rule, preserving the process name, UID, timestamp, and arguments of every bpf() invocation — these records establish the timeline of BPF program loading and map creation by the Red Menshen implant. | Deleted-on-disk binary evidence: output of 'readlink /proc/[PID]/exe' for the suspect process — Red Menshen BPFdoor variants have been observed deleting their on-disk binary after execution, leaving a running process whose /proc/[PID]/exe symlink resolves with a '(deleted)' suffix, which is a high-confidence indicator of deliberate anti-forensic tradecraft specific to this implant family.

Per-Action IR Details

Step 1: Containment — Identify Linux hosts on telecom backbone and carrier-grade infrastructure with BPF-capable kernels. Restrict CAP_NET_ADMIN and CAP_SYS_ADMIN capabilities to explicitly authorized processes using capability bounding sets. Where possible, enforce seccomp profiles that block bpf() syscall for non-privileged workloads. Isolate any host exhibiting anomalous raw socket activity or unexpected BPF program loads pending investigation.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), NIST CM-7 (Least Functionality), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: For teams without enterprise tooling: run 'capsh --print' on each host to enumerate effective capabilities and identify any process holding CAP_NET_ADMIN or CAP_SYS_ADMIN outside of expected network daemons. Use 'ss -s' and 'cat /proc/net/packet' to identify raw socket holders. Apply a seccomp profile blocking bpf() syscall using a minimal JSON seccomp filter deployed via systemd's SystemCallFilter directive — example: 'SystemCallFilter=~bpf' in the unit file for non-privileged services. Network-isolate suspect hosts by pushing a null-route for their management IPs via the upstream carrier router or by applying iptables rules dropping all non-management

traffic: 'iptables -I INPUT -j DROP; iptables -I OUTPUT -j DROP' with explicit management allow rules prepended.

Evidence: Before restricting capabilities, capture the full output of 'bpftool prog list --json > bpf_prog_inventory_$(hostname)_$(date +%s).json' and 'bpftool map list --json' to document all loaded BPF programs and maps — BPFdoor's packet filter program will appear as a SOCKET_FILTER type program with no associated authorized agent. Capture 'cat /proc/net/packet > raw_socket_holders.txt' and 'cat /proc/net/raw >> raw_socket_holders.txt' to record all raw socket holders at time of isolation. Preserve 'ls -laR /proc/[0-9]*/fd | grep socket' output to correlate socket file descriptors to PIDs before any process termination.

Step 2: Detection — Audit loaded BPF programs on all Linux hosts using 'bpftool prog list' and 'bpftool map list'; flag any programs not associated with known, authorized network monitoring agents. Review /proc/net/packet and /proc/net/raw for unexpected raw socket holders. Inspect kernel logs (dmesg, /var/log/kern.log) and auditd records for bpf() syscall invocations from non-standard processes. Hunt for processes with no associated listening port that nonetheless receive inbound packets — BPFdoor's defining behavioral signature. Cross-reference against MITRE T1205.002 and T1014 hunting logic.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy auditd rules to capture bpf() syscall invocations by adding '-a always,exit -F arch=b64 -S bpf -k bpf_syscall_monitor' to /etc/audit/rules.d/bpfdoor.rules, then 'auditctl -R /etc/audit/rules.d/bpfdoor.rules'. Query captured events with 'ausearch -k bpf_syscall_monitor -i | grep -v authorized_agent_name'. For hunting BPFdoor's passive listener signature, use osquery with the query: 'SELECT pid, name, cmdline FROM processes WHERE pid IN (SELECT pid FROM process_open_sockets WHERE family=17 AND local_port=0);' — this surfaces SOCK_RAW holders with no bound port, BPFdoor's defining trait. For kernel-level BPF tracing without EDR, deploy Falco with rule: 'syscall bpf evt_type=enter proc_name!=(expected_agents) → priority CRITICAL'. Use the community Sigma rule 'linux_bpf_program_load_unknown' mapped to T1205.002 as a detection baseline.

Evidence: Capture 'dmesg -T | grep -i bpf > dmesg_bpf_$(date +%s).txt' and 'journalctl -k --since=boot | grep -i bpf' to identify kernel-level BPF events since last boot — Red Menshen's BPFdoor variant loads its SOCKET_FILTER program at implant startup. Extract auditd logs with 'aureport --syscall | grep bpf' and preserve the full audit.log for forensic analysis. Run 'strace -e bpf -p $(pgrep -d, suspicious_proc)' if a suspect process is identified live. Collect 'ss -nlp' and 'netstat -anlp' output to confirm no bound listening port exists for the suspicious process — the absence of a listening port combined with raw socket presence is the primary BPFdoor behavioral indicator per MITRE T1205.002.

Step 3: Eradication — No vendor patch exists; eradication requires host reimaging after confirming compromise. Before reimaging, collect a full memory image and disk forensic snapshot for incident analysis. Remove any unauthorized BPF programs identified during detection using 'bpftool prog unload' as an interim measure only — this does not guarantee full removal if persistence mechanisms (T1543.002: Systemd Service) are present. Audit systemd unit files, init scripts, and cron jobs for entries referencing unfamiliar binaries or paths.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software)

Compensating: Memory acquisition on a resource-limited team: use LiME (Linux Memory Extractor) kernel module — 'insmod lime.ko path=/mnt/evidence/mem_$(hostname)_$(date +%s).lime format=lime' — to capture a full RAM image before any eradication action. For disk imaging use 'dc3dd if=/dev/sda hash=sha256 hlog=/mnt/evidence/hash.log of=/mnt/evidence/disk_$(hostname).dd'. For systemd persistence audit, run 'systemctl list-units --type=service --all | grep -v known_services' and 'find /etc/systemd /usr/lib/systemd /run/systemd -name "*.service" -newer

/var/log/dpkg.log' to surface recently created unit files consistent with BPFdoor's T1543.002 persistence. Check cron with 'for u in $(cut -f1 -d: /etc/passwd); do crontab -u $u -l 2>/dev/null; done' and inspect '/etc/cron.d/', '/etc/cron.daily/', '/var/spool/cron/'.

Evidence: Before reimaging, preserve: (1) Full LiME memory image — BPFdoor operates in kernel space and its BPF bytecode, magic packet trigger value, and C2 configuration may only exist in memory. (2) Full disk image via dc3dd with hash verification. (3) Artifact-specific: run 'find / -name ".*" -o -name "*bpf*" -o -name "*door*" 2>/dev/null | grep -v proc | grep -v sys > hidden_file_candidates.txt' — Red Menshen variants have historically used dot-prefixed hidden filenames. (4) Collect 'find /etc/systemd /usr/lib/systemd -name "*.service" -exec md5sum {} \; > systemd_unit_hashes.txt' and compare against known-good baseline. (5) Capture 'ls -la /proc/$(pgrep suspicious_proc)/exe' and 'readlink /proc/$(pgrep suspicious_proc)/exe' — BPFdoor has been observed with deleted-on-disk binaries that remain running, indicated by '(deleted)' in the symlink target.

Step 4: Recovery — After reimaging, validate that BPF program inventory returns to known-good baseline using 'bpftool prog list'. Reconfirm that no raw sockets are held by unauthorized processes. Restore systems from verified clean backups only. Deploy continuous kernel telemetry (e.g., Falco, Tracee, or eBPF-based EDR) to monitor for BPF program loads, raw socket creation, and capability abuse going forward. Monitor outbound traffic patterns from restored hosts for anomalous application-layer communication consistent with T1071.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CP-10 (System Recovery and Reconstitution), NIST AU-2 (Event Logging), NIST SI-4 (System Monitoring), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 8.2 (Collect Audit Logs)

Compensating: Post-reimage BPF baseline verification: immediately after OS restore, run 'bpftool prog list --json | python3 -c "import json,sys; progs=json.load(sys.stdin); [print(p) for p in progs if p.get(\"type\") == \"socket_filter\"]"' — any SOCKET_FILTER type program on a freshly imaged host without authorized monitoring agents is an immediate re-compromise indicator. Deploy Falco (free, CNCF project) with the default ruleset plus a custom rule alerting on bpf() syscall from any process not in an explicit allowlist. For outbound C2 traffic monitoring consistent with T1071, deploy Zeek (formerly Bro) on a network tap and alert on unusual application-layer protocols from restored telecom hosts — BPFdoor C2 sessions over TCP/UDP on non-standard ports would appear in 'conn.log' with long durations and low byte counts characteristic of command-and-control beaconing.

Evidence: Post-recovery monitoring artifacts to establish: (1) Snapshot 'bpftool prog list --json' output immediately post-reimage as the authorized baseline document, signed and stored in change management. (2) Enable auditd bpf() syscall rules (as defined in Step 2) on restored hosts immediately — first 72 hours of logs are critical for detecting re-compromise attempts by Red Menshen, which has demonstrated persistence and re-targeting of previously compromised telecom infrastructure. (3) Configure Zeek or tcpdump packet capture on host egress for 30 days post-recovery, filtering for traffic from the restored host IP to flag any re-establishment of BPFdoor's covert channel communication.

Step 5: Post-Incident — This campaign exposes gaps in kernel-level visibility, BPF program inventory controls, and Linux privilege management. Conduct a capability audit across all Linux infrastructure — CAP_NET_ADMIN and CAP_SYS_ADMIN should be explicitly granted, logged, and reviewed. Evaluate deployment of eBPF-based runtime security tools that can observe kernel activity in real time. Map detection coverage against MITRE T1205.002, T1014, T1543, and T1059.004 to identify gaps in current SIEM and EDR rules. Engage threat hunting cadence specifically targeting passive backdoor indicators on high-value Linux hosts.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CM-7 (Least Functionality), NIST RA-3 (Risk Assessment), NIST SI-4 (System

Monitoring), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Capability audit across Linux fleet without enterprise tooling: deploy an osquery scheduled query pack with 'SELECT pid, name, uid, capabilities FROM processes WHERE capabilities LIKE "%cap_net_admin%" OR capabilities LIKE "%cap_sys_admin%";' on a 15-minute interval to maintain a continuous inventory of privileged processes. Author a Sigma rule mapped to T1205.002 targeting the specific behavioral pattern (SOCK_RAW holder + no bound port + inbound packet receipt) and submit to the community Sigma repository for peer validation. For threat hunting cadence, schedule a bi-weekly hunt procedure using osquery + 'bpftool prog list' output diff against the post-reimage baseline — any net-new SOCKET_FILTER or SCHED_CLS type BPF program on a telecom host is a mandatory escalation trigger. Reference Tracee (Aqua Security, Apache 2.0 license) as a zero-cost eBPF runtime security tool deployable without kernel module signing requirements on most distributions.

Evidence: Post-incident lessons-learned should document: (1) Timeline of initial BPF program load event from auditd records versus first detection — this gap quantifies kernel-level detection blindspot for the after-action report. (2) Comparison of 'bpftool prog list' output from compromised hosts versus known-good baseline to characterize the specific BPF program type, tag, and map associations used by this Red Menshen variant. (3) Zeek conn.log entries or tcpdump captures showing any covert channel traffic patterns used for C2 communication — these should be submitted to sector-specific ISACs (telecom sector: Communications ISAC) as threat intelligence for peer organizations.

Detection Guidance

Primary detection relies on kernel-level and process-level telemetry rather than network scanning, which BPFdoor defeats by design. Key detection methods: (1) Run 'bpftool prog list' and 'bpftool map list' on all Linux hosts; investigate any BPF program not attributable to a known, authorized agent. (2) Check /proc/net/packet and /proc/net/raw for processes holding raw sockets, cross-reference PIDs against running process list and known-good baselines. (3) Enable auditd rules on the bpf() syscall (syscall=321 on x86_64) and alert on invocations from processes outside an approved allowlist. (4) Use Falco or Tracee rules targeting 'bpf' syscall events and raw socket creation from non-root, non-monitoring processes. (5) Hunt for network traffic that triggers a process state change on a host with no corresponding listening port, this is the behavioral signature of magic-packet activation (T1205.002). (6) Inspect /proc/[pid]/exe and /proc/[pid]/maps for deleted-on-disk binaries, a common BPFdoor persistence technique. (7) Review systemd unit files and cron jobs for unfamiliar entries pointing to non-standard binary paths (T1543.002). Note: standard port scans, netstat, and ss will not reveal BPFdoor's presence. Network-layer detection is insufficient without kernel telemetry.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`[refer to Rapid7 and Security Affairs threat research reports for current file hashes]`	BPFdoor upgraded variant — specific hash values should be extracted directly from linked primary research reports; hashes are not reproduced here to avoid transcription error	LOW

Type	Value	Context	Confidence
DOMAIN	`[no confirmed C2 domains publicly attributed at this time]`	BPFdoor is designed to minimize outbound C2 footprint; command channels are initiated via magic-packet trigger rather than persistent outbound beaconing	LOW

Framework Mappings

MITRE-ATTACK

- **T1102** — Web Service
- **T1059.004** — Unix Shell
- **T1014** — Rootkit
- **T1106** — Native API
- **T1205.002** — Socket Filters
- **T1205** — Traffic Signaling
- **T1071** — Application Layer Protocol
- **T1543** — Create or Modify System Process
- **T1057** — Process Discovery
- **T1049** — System Network Connections Discovery
- **T1543.002** — Systemd Service

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AC-3** — Access Enforcement
- **AC-6** — Least Privilege

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

CIS-V8

- **6.1**
- **6.2**
- **5.4**
- **6.8**
- **8.2** — Collect Audit Logs

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.312(e)(1)** — Transmission Security

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1102	Web Service	Command-And-Control
T1059.004	Unix Shell	Execution
T1014	Rootkit	Defense-Evasion
T1106	Native API	Execution
T1205.002	Socket Filters	Defense-Evasion
T1205	Traffic Signaling	Defense-Evasion
T1071	Application Layer Protocol	Command-And-Control
T1543	Create or Modify System Process	Persistence
T1057	Process Discovery	Discovery
T1049	System Network Connections Discovery	Discovery
T1543.002	Systemd Service	Persistence

Sources

Source	URL	Tier
Security News	https://www.darkreading.com/threat-intelligence/china-upgrades-back...	T3
BPFdoor in Telecom Networks: Sleeper Cells in the backbone	https://www.rapid7.com/blog/post/tr-bpfdoor-telecom-networks-sleepe...	T3

Source	URL	Tier
BPFdoor hides deep inside the OS kernel to target telecoms worldwide	https://www.scworld.com/news/bpfdoor-hides-deep-inside-the-os-kerne...	T3
Espionage campaign targets telecom with stealthy Linux-based ...	https://www.cybersecuritydive.com/news/espionage-campaign-telecom-l...	T3
China-linked Red Menshen APT deploys stealthy BPFDoor implants ...	https://securityaffairs.com/190029/malware/china-linked-red-menshen...	T3

Generated 2026-03-29 18:35 UTC by TJS Security Command Center