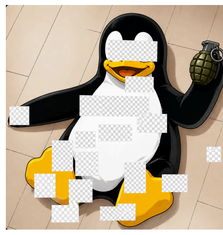
stong add pics 532994f · 13 minutes ago

Name	Last commit message	Last commit date
..		
README.md	add pics	13 minutes ago
fragnesia.c	Fragnesia	5 hours ago

README.md

Fragnesia



Abstract

vuln.mp4

0:00 / 0:17

Fragnesia is a universal Linux local privilege escalation exploit, discovered with V12 by William Bowling with the V12 team. Fragnesia is a member of the Dirty Frag vulnerability class. This is a **separate bug** in the ESP/XFRM from dirtyfrag which has received its own patch. However, it is in the same surface and the mitigation is the same as for dirtyfrag.

It abuses a logic bug in the Linux XFRM ESP-in-TCP subsystem to achieve arbitrary byte writes into the kernel page cache of read-only files, without requiring any race condition.

The technique extends the page-cache write bug class that includes Dirty Pipe: when a TCP socket transitions to `espintcp` ULP mode after data has already been spliced from a file into the receive queue, the kernel processes the queued file pages as ESP ciphertext. The AES-GCM keystream byte at counter block position 2, byte 0 is XORed directly into the cached file page. By selecting the IV nonce to produce a desired keystream byte, any target byte in the file can be set to any value — one byte per trigger invocation.

The exploit builds a 256-entry lookup table mapping each possible keystream byte to its corresponding nonce, then iterates over a payload, firing the splice/ULP race for each byte that needs changing. It writes a small position-independent ELF stub (setresuid/setresgid/execve `/bin/sh`) over the first 192 bytes of `/usr/bin/su` in the page cache, then calls `execve("/usr/bin/su")` to obtain a root shell. The page cache modification is not backed to disk; the on-disk binary is untouched.

"Fragnesia"?

Yes, because the core bug is: the skb "forgets" that a frag is shared during coalescing.

Exploitation

One-line special:

```
git clone https://github.com/v12-security/pocs.git && cd pocs/fragnesia && gcc -c
```

Ubuntu note: AppArmor restricts unprivileged user namespaces by default. You must first run:

```
sudo sysctl -w kernel.apparmor_restrict_unprivileged_userns=0
```

You can chain other bugs to bypass this requirement but this is out of scope for this vulnerability.

The exploit targets `/usr/bin/su` directly. On success it drops into a root shell.

Critical Cleanup Warning

After the exploit runs, `/usr/bin/su` in the page cache contains the injected stub. Any subsequent execution of `su` will re-spawn a shell until the page is evicted. Drop the cache or reboot before leaving the machine:

```
echo 1 | tee /proc/sys/vm/drop_caches
```

Mitigation

Same as dirtyfrag.

```
rmmod esp4 esp6 rxrpc
printf 'install esp4 /bin/false\ninstall esp6 /bin/false\ninstall rxrpc /bin/fals
```

Affected Versions

All versions affected by dirtyfrag are affected.

Any versions without this patch: https://lists.openwall.net/netdev/2026/05/13/79, so Linux kernels before May 13 2026.

Confirmed working on `Linux localhost 6.8.0-111-generic #111-Ubuntu SMP PREEMPT_DYNAMIC Sat Apr 11 23:16:02 UTC 2026 x86_64 x86_64 x86_64 GNU/Linux` (vps purchased from Linode)

How It Works

1. **User + network namespace setup.** The exploit calls `unshare(CLONE_NEWUSER | CLONE_NEWNET)` to obtain a namespace where it holds `CAP_NET_ADMIN` without any real privileges on the host.
2. **XFRM SA installation.** Inside the network namespace, a transport-mode ESP-in-TCP security association is installed via `NETLINK_XFRM` using AES-128-GCM with a known key and SPI `0x100`.
3. **Keystream table.** The 16-byte AES-GCM counter block for sequence position 2 is `[salt || IV || 00000002]`. Encrypting it under the known key yields a 16-byte keystream block; only byte 0 is used. By varying the lower 32 bits of the 8-byte IV (nonce), all 256 possible stream-byte values are reachable within the first 65536 nonces. The table is built once via `AF_ALG` and reused for every byte flip.
4. **Splice-then-ULP trigger.** A sender/receiver pair is forked. The sender splices 4096 bytes from the target file (starting at the byte to corrupt) into the TCP stream, prepended by an ESP-in-TCP length word and an ESP header constructed with the chosen IV. The receiver delays `TCP_ULP espintcp` installation until the data is in the socket buffer. When ULP is enabled, the kernel attempts to decrypt the queued ESP record in-place, XORing the GCM keystream into the splice-mapped file page — the same physical page backing the VFS page cache entry.
5. **Byte-by-byte payload write.** Steps 3–4 repeat for each byte in the payload that does not already hold the desired value. Each iteration recomputes the required keystream byte, looks up the corresponding IV nonce, increments the ESP sequence number, and fires a fresh trigger pair.

6. **Execution.** After all 192 payload bytes are verified, `execve("/usr/bin/su")` runs the now-modified in-cache binary, which calls `setresuid(0,0,0)` / `setresgid(0,0,0)` and execs `/bin/sh`.

Credit

Found with V12 by William Bowling on the V12 team: v12.sh — dangerously powerful agentic security.