A short and incomplete history of ELF binary format manipulation

vrzh, sblip

The Morris Worm

Not ELF, not Linux, but a good case study of the CFAA

- **Morris Worm (Internet Worm)**
  Author: Robert Tappan Morris, Cornell graduate student
  Date Released: November 2, 1988

- **Intended Goal**: Measure the size of the Internet (ARPANET).
  Actual Outcome: Triggered the first major, widespread computer network outage.

- **Systems Targeted**: UNIX systems — mainly 4BSD variants on VAX and Sun-3 machines.
  Impact: Infected roughly 6,000 systems (about 10 % of the Internet at the time), causing slowdowns and crashes from excessive replication. Did **not** destroy files.

- **Legal Precedent**: First conviction under the Computer Fraud and Abuse Act of 1986 (CFAA).

STAOG

The first known Linux virus (disputable), discovered October 20, 1996. Written in Assembly by Quantum/VLAD, it targeted ELF binaries and attempted to gain root privileges by exploiting three separate kernel/utility vulnerabilities to remain memory-resident. Its purpose was primarily a proof-of-concept.

10/20/1996, Quantum/VLAD

Linux.Bliss

A follow-up ELF file infector from a member of the 29A group. It was a more direct file infector, appending the viral code to the existing executable and modifying the entry point (`e_entry`). It was notable for attempting to spread in the wild, though it was easily patched.

The virus was created by an individual using the pseudonym "Electric Eel" and was first publicly released as an alpha version on September 29, 1996, with a more stable version released on February 5, 1997.

The file is executed, the original program is extracted to the /tmp directory under the process ID name with the prefix ".bliss-tmp." added to it. Infected files will run correctly, though shell scripts may complain about them.

2/5/1997, Electric Eel

Unix ELF Parasites and Virus

Exploits the memory alignment requirements of the ELF format to find "padding" space at the end of the text segment.

The Technique: The parasite code is inserted into this padding area.

The file's internal structures (specifically the program headers and the section header table) are then carefully modified to:

- 1. Increase the size of the text segment to encompass the new parasite code.
- 2. Redirect the Entry Point of the executable to jump to the newly inserted parasite code first.
- 3. The parasite code executes its malicious payload and then transfers control back to the original program’s entry point, allowing the host program to run normally.

- Limitation: This method typically limits the size of the parasite to the size of a memory page (e.g., 4KB), as extending beyond this would require major file and segment shifts.

8/1998, Silvio

The publication of this paper sparked the creation of the UNIX virus mailing list, and more mainstream interest in ELF viruses and advanced manipulation. Previously viruses were almost exclusively worked on in the Vx community.

Runtime kernel kmem patching

Runtime kernel kmem patching is a powerful, low-level technique used to modify the code or data of a running kernel directly in memory, which, historically, was often accomplished via the now-restricted /dev/kmem device file. This technique is chiefly employed by kernel-level rootkits for malicious purposes, such as hiding processes and files by hooking system calls (like lsmod or the System Call Table), but is also the fundamental concept underlying modern, legitimate live-patching systems (like Ksplice) designed to apply critical security updates without requiring a system reboot.

1998, Silvio, published in Mundo Toxico 1

Diesel

Diesel is a direct action infector [virus] for Linux that places its code in the middle of files it infects while preserving the part in the middle it would otherwise overwrite. It was created by [Paddingx] in France in 1999 and it appeared in the fourth issue of 29a.

Notably, this virus used code caves in the text segment to hide its code. (Petter Ferrie, 2025)

3/13/1999, PaddingX (Paddingten, haha)

Shared Library Call Redirection via ELF PLT infection

Shared Library Call Redirection via ELF PLT infection is a malware technique that hijacks the normal flow of execution for functions imported from dynamic libraries. The attack exploits the Procedure Linkage Table (PLT), which is an array of small code stubs, and the associated Global Offset Table (GOT), which contains the actual addresses for shared library functions. By overwriting a target function's GOT entry—which is initially written to by the dynamic linker at runtime—the attacker redirects the call from the legitimate library function to their own malicious code (the 'parasite'), typically one they have injected into the executable, thereby subverting program execution to achieve goals like privilege escalation or code execution persistence. Published in Phrack Magazine (Issue 56, December 2000)

12/2000, Silvio

The Cerberus Interface

The Cerberus Interface describes a collection of sophisticated, on-disk infection techniques targeting the ELF (Executable and Linkable Format) binary structure on Linux/UNIX systems. The primary goal of these methods is to achieve stealthy, persistent code injection, often by manipulating the dynamic linker's behavior. A key technique involves modifying the ELF's Dynamic Segment, specifically by overriding or extending the **DT_NEEDED** section to force the executable to load a malicious shared library (.so) at runtime, essentially creating a backdoor for malware or rootkits without altering the program's main execution code. This method allows an attacker to hijack the application's runtime dependencies and inject arbitrary code into the process's memory space.

5/2000, Mayhem

W32.Winux / Lindose

a single binary file written in x86 assembly language that contained code for both platforms. When executed, it checked the operating system environment and infected local executables using the native format's infection method.

It was the first publicly known virus to successfully and natively infect executables on both the Windows and Linux operating systems from a single file, demonstrating true cross-platform binary infection capability.

Note: It’s a bit of a stretch to call this a fully cross-platform virus, as it does not execute on Linux natively, at least until the first iteration has run.

3/27/2001, Benny - 29a

ERESI Research Project

The ERESI (ERESI Research Software Interface) framework emerges (early development dates vary, but core visibility and use formalize around this time). Developed by Julien "juri" Voisin, it is a comprehensive, open-source toolset for deep binary analysis, editing, and reverse engineering, enabling researchers to perform the structural and symbolic changes necessary for advanced ELF hijacking.

2005, juri

Exploiting the hard working DWARF

The paper demonstrates that DWARF (Debugging With Attributed Record Format) data used for exception handling in GCC-compiled C++ binaries contains a **Turing-complete** bytecode virtual machine** that is executed during stack unwinding.

Attackers can modify this DWARF bytecode (specifically in the .eh_frame or .debug_frame sections of the ELF file) to implant a trojan payload that contains no native executable code. This payload is executed by the operating system's exception handling routine (the DWARF VM) when a specific error or exception is triggered, allowing for sophisticated control-flow hijacking without touching the main executable code segment.

August, 2011 - James Oakley, Sergey Bratus

Back to the Backdoor Factory

The article "Back to the Backdoor Factory" combines classic **on-disk ELF binary infection** techniques with modern network tools for a complete attack chain. While the core methods involve static file manipulation (like Text Segment Padding) to persistently embed a malicious payload inside an executable, the final step involves using a network tool like **Bettercap** to deliver or control the backdoor. Specifically, an infected binary can be used in a malicious context, and the attacker would then use **Bettercap** to intercept and modify network traffic or perform local Man-in-the-Middle (MITM) attacks to facilitate the infection's delivery to a target host or to receive outbound command-and-control (C2) communications from the running backdoor. This approach highlights how persistent file-based backdoors can be leveraged within sophisticated network-based exploitation frameworks.

5/17/2020 – awgh, sblip - https://youtu.be/NaT8rxmfThg

tmp.out

The tmp.out research collective launched its first volume in April 2021. This collective carries the torch of low-level ELF research, publishing highly technical papers on modern topics like ELF binary "golfing," new infection algorithms for 64-bit and PIE (Position-Independent Executable) binaries (e.g., Linux.Nasty) Linux.Kropotkine, and other advanced fileless and stealth techniques.

4/2021 – tmp.out staff

tmp.0ut Volume 1 - April 2021

CONTENTS