Linux Incident Response - ELF sections

Taz Wake | Halkyn Consulting Ltd | Originally published 2023-11-03

In the recent article about the objdump command, the concept of "sections" within an ELF file was mentioned, although without much detail. As a result, this article will try to summarise the most common sections we find in Linux ELF files and give an overview of the most relevant points for an incident responder.

Common sections within an ELF file

1. **.init Section:**
   - **Purpose:** The .init section contains code that is executed when the program is loaded into memory. This code is responsible for performing various initialization tasks required by the program before it starts executing its main logic. It ensures that the program starts in a known and stable state.
   - **Contents:** Functions or routines that set up global variables, establish necessary data structures, and perform other initialization tasks.
   - **Execution Time:** The code in the .init section is executed during the program's loading phase, specifically after the program has been loaded into memory but before its main() function (or equivalent) is called.
2. **.data Section:**
   - **Purpose:** The .data section contains initialized global and static variables. These variables have predefined values specified by the programmer and are initialized before the program starts executing. Any variable with an initial value, like int x = 10;, will be stored in the .data section.
   - **Contents:** Initialized global and static variables with predefined values.
   - **Execution Time:** The values in the .data section are set during the program's loading phase, just like the variables in the .init section. However, the .data section is primarily concerned with storing data rather than code.
3. .**text Section:**
   - **Purpose:** The .text section contains the executable code of the program, including the instructions that the CPU interprets and executes. It represents the main body of the program, encompassing functions, control flow, and actual operations.
   - **Contents:** Assembly instructions and machine code generated from the program's source code.
   - **Execution Time:** The code in the .text section is executed during the program's runtime. It represents the core logic of the program and is responsible for performing the intended operations specified by the programmer.

4. **.bss Section:**

- **Purpose:** The .bss section, also known as the Block Started by Symbol section, stores uninitialized or zero-initialized global and static variables. These variables do not have specific values assigned in the source code but are initialized to zero or null values during program execution.
- **Contents:** Uninitialized or zero-initialized global and static variables.
- **Execution Time:** The variables in the .bss section are initialized to zero or null values during the program's loading phase, just before the program starts executing. They are ready for use but do not consume space in the executable file, as their values are set dynamically in memory.

5. **.rodata Section:**

- **Purpose:** The .rodata section, short for Read-Only Data, contains constants and read-only variables used by the program. These values remain constant during the program's execution and cannot be modified.
- **Contents:** Constants, read-only variables, and other immutable data.
- **Execution Time:** The data in the .rodata section is accessible throughout the program's runtime but cannot be modified. It is typically used for storing constant values, strings, and other data that should not be altered during program execution.

6. **.symtab Section:**

- **Purpose:** The .symtab section, or Symbol Table, holds information about symbols used in the program, including function and variable names. It provides a mapping between symbol names and their corresponding memory addresses, aiding in the linking and debugging processes.
- **Contents:** Symbol names, types (function, object, etc.), memory addresses, and other related information.
- **Execution Time:** The .symtab section is primarily used during the linking phase to resolve symbols and establish memory addresses for various program elements. While not directly involved in the program's execution, it is crucial for proper linking and debugging of the executable.

#linux #elf #executable #malware #incidentresponse #dfir #infosec #cybersecurity #cyber #security

---

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858