SPYWARE ANALYSIS REPORT

Preparer: Vəli Qasımlı

History: 23.02.2025

Content

1. Static analysis
2. Dynamic analysis

Malware Analysis is the process implemented to analyze the impact, behavior and spread of malicious programs. This analysis is carried out to protect systems and strengthen safety measures.

We use the strings command to analyze statics.

The following critical functions and system calls were detected in the file:

dlopen, dlclose, dlsym → Dynamic library loading and running operations.

fork, waitpid → Mechanisms for creating and controlling new processes.

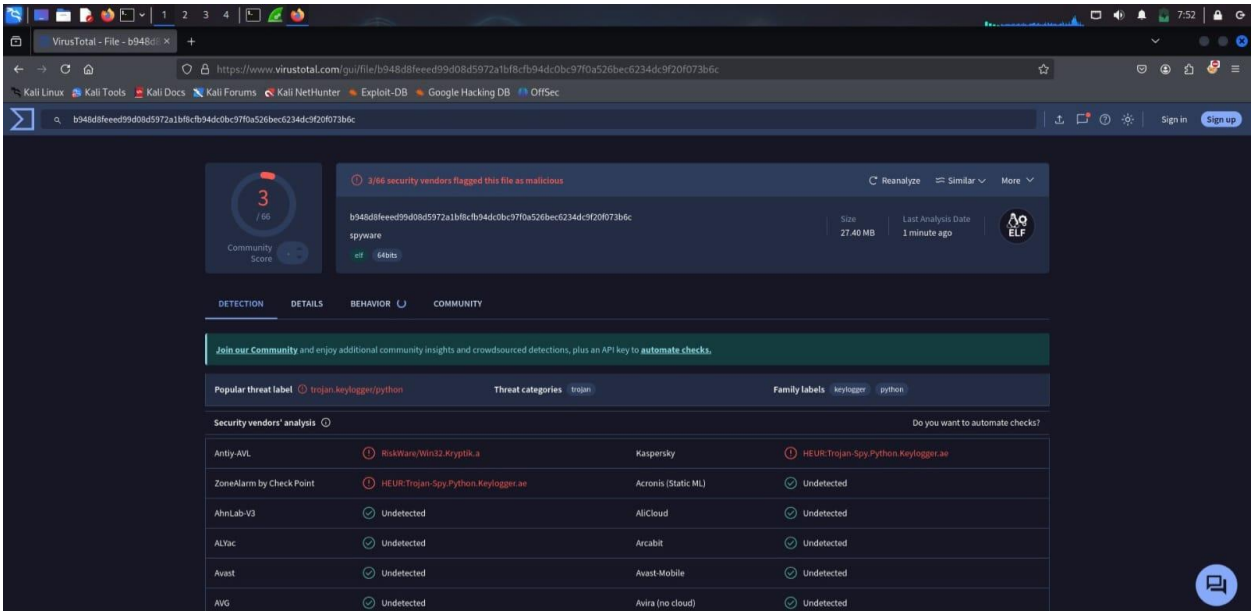
fopen, readdir, unlink, mkdir → File reading, listing directory contents, file deletion, directory creation operations.

strcpy, strncpy, strdup → String copying and memory operations (there may be a risk of Buffer Overflow).

signal → Process management and signal capture (e.g. interrupt signals such as Ctrl+C).

chmod, lstat → Changing file permissions and reading file statistics.

inflate, inflateInit, inflateEnd → Decompressing data (Possibly there is data encryption or storage involved).



This analysis contains the results of a file scan performed on the VirusTotal platform. The file is an ELF (Executable and Linkable Format) 64-bit executable file.

Scanned by 66 antivirus engines.

3 antivirus engines marked this file as malicious.

File name: spyware

This analysis involves reverse engineering of the spyware file, performed using the Radare2 (r2) tool.

WARN: Relocs has not been applied.

Relocation information has not been implemented yet. Full analysis can be performed with the -e bin.relocs.apply=true or -e bin.cache=true option.

INFO: Various analysis operations were performed:

Imports & Entry Point: Imported functions and entry points analyzed.

Symbols & Functions: Symbols, functions and variables in the file were examined.

Function Calls & Arguments: All function calls and arguments analyzed.

Binary Parsing: Structures such as C++ vtable (virtual function tables) were detected.

Local Variables & Propagation: Local variables were recovered and noreturn propagated.

String Scanning: Strings hidden in the code were scanned.

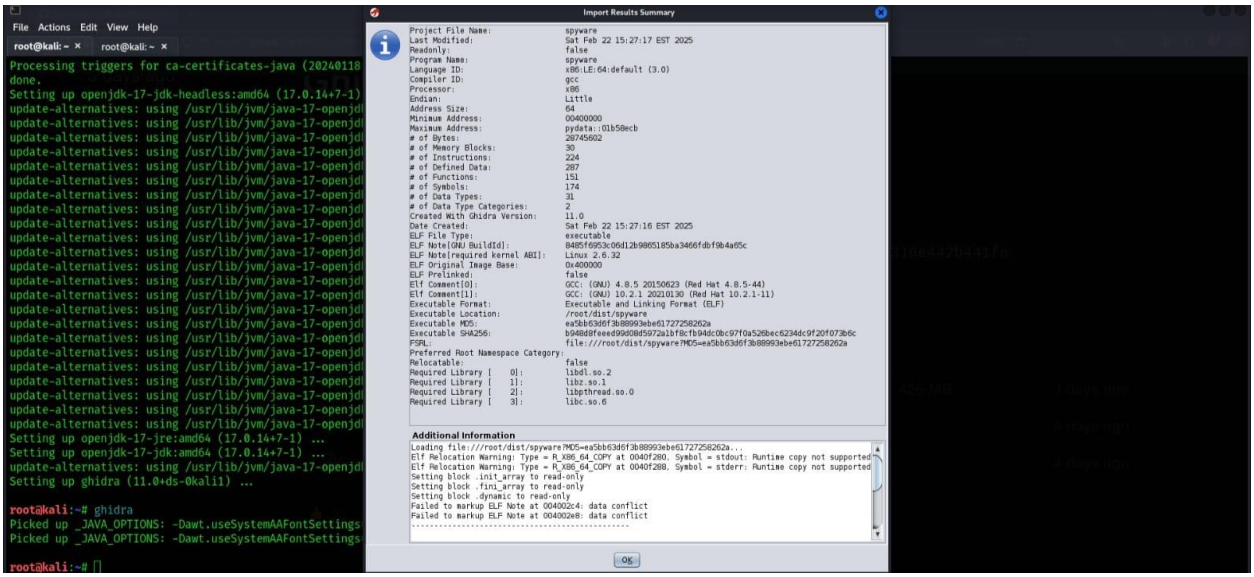
Function Prologues: Function beginnings were examined.

Experimental Type Propagation: anal.types.constraint enabled.

The analyzed processes show that the file has a complex structure and probably contains malicious functions.

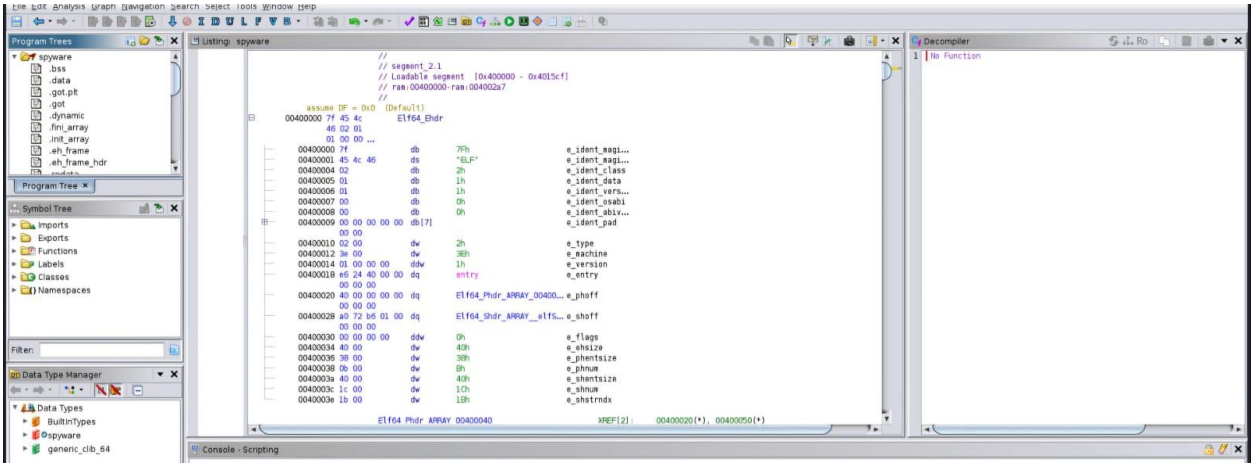
Ghidra is a reverse engineering tool used for static analysis. It extracts the assembler code and C-like code of the program.

Program Tree: Contains sections such as .bss, .data, .got, .rodata. These sections contain global variables, constant data, function tables and program codes.



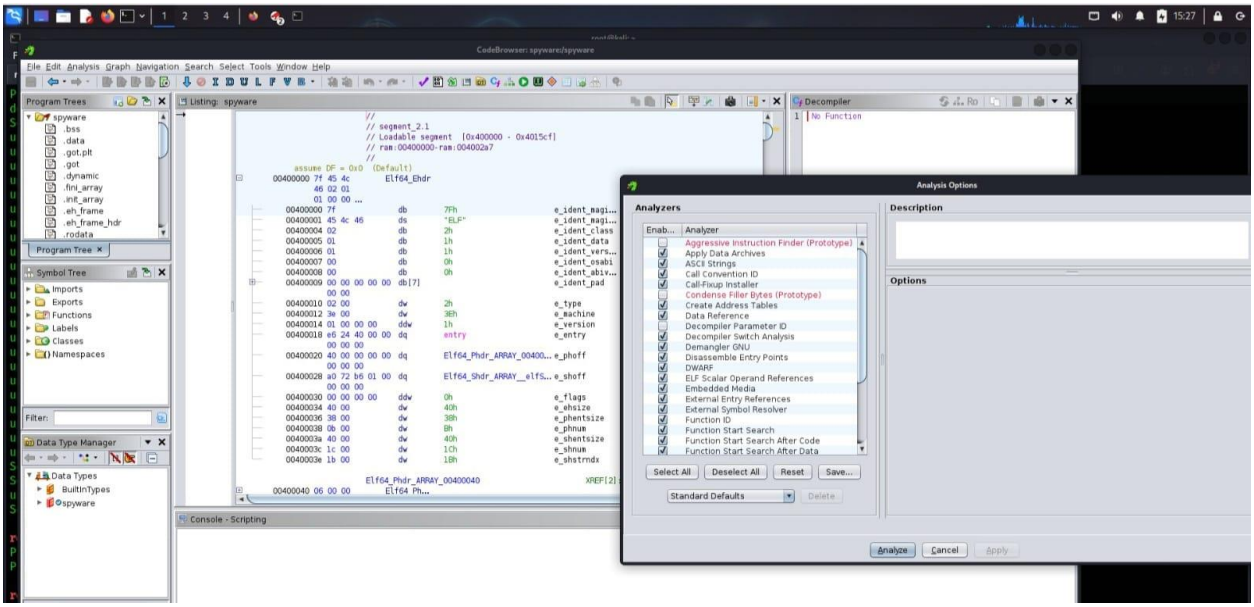
Symbol Tree: Imports: Contains functions called from external libraries. For example, if there are network-related calls such as socket(), connect(), send(), this may be a backdoor or data leaking software.

Exports: Shows the functions that the file offers to the outside world.



Functions: Lists all functions available in the file.

Labels, Classes, Namespaces: Contains details about variables and objects within the program.



GNU Build ID:
8485f6935c06d1b2985b813e3466dfb9b4a95c

GCC Version: 8.5.0

File Format: ELF (Executable and Linking Format)

Original Image Base: Linux 2.6.32

This information shows that the file named spyware was compiled with GCC on a Linux distribution such as Red Hat/CentOS and is an executable file. GNU Build ID is the unique ID of the file and can be used in hash-based analysis.

Relocation Warning:

R_X86_64_COPY relocation error occurred. This indicates that a copy operation related to standard output, such as stdout, was run unexpectedly.

Text Block Warnings: The .init_array section has been put into read-only mode. This may contain potentially malicious code because it contains functions that can be run while the program is starting. ELF Note Markup Errors:

Address: 0x400042c - "data conflict"

Address: 0x4000282 - "data conflict"

These errors indicate that there may have been a conflict or memory manipulation in the ELF note sections.

2.Dynamic analysis

Analysis of Strace Output

1. Running the Program and Memory Management

execve("./spy", ["./spy"], 0x7ffe220fb100 /* 57 vars */) = 0

The execve call indicates that the executable named ./spy has been started.

brk(NULL) = 0x5644d2cb1000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdbbec30000

The brk and mmap calls are related to memory management.

With mmap, the memory area is reserved, it has PROT_READ|PROT_WRITE permissions, so it can be both read and written.

2. Interaction with System Libraries

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=96070, ...}) = 0

close(3)

access("/etc/ld.so.preload") error:

The program tries to check the dynamic library preload file (/etc/ld.so.preload), but the file does not exist.

/etc/ld.so.cache opened:

The program examines the cache file to see which shared libraries are installed on the system.

3. Installing the libc Library

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3pread64(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0", 64, 0) = 64

The libc.so.6 library has been opened and read.

libc (C standard library) provides system calls and basic functions.

4. Suspicious Memory Accesses

mmap(NULL, 2055604, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdbbea4a000

mmap(0x7fdbbeaf0000, 1462272, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fdbbeaf0000 mmap(0x7fdbbec05000, 325276, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e2000) = 0x7fdbbec05000

mmap(0x7fdbbec0b000, 52696, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdbbec0b000

Doubtful points:

Anonymous writing to memory (MAP_ANONYMOUS) is performed.

With MAP_FIXED, memory is allocated to certain addresses with PROT_EXEC permission → Code injection may occur.

5. Reading User Information and Process Manipulation

arch_prctl(ARCH_SET_FS, 0x7fdbbea1f740) = 0

set_tid_address(0x7fdbbea1fa10) = 2

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

The arch_prctl call is related to thread management.

The call to set_tid_address determines the ID of the thread.

Stack size is adjusted with prlimit64.

Some malware uses this to bypass memory restrictions.