Cyberpsychosis (reverse engineering)

Challenge

The challenge presents a `.ko` file, which is a **kernel module**. Kernel modules are pieces of code that can be added to the kernel at runtime to provide additional functionality, such as device drivers, file systems, or other kernel features. [1]

The file is named `diamorphine.ko`, and is a **rootkit**. Rootkits are a type of malware that are designed to provide a set of functions to an attacker, while hiding their presence on the system. They are often used to maintain access to a system, or to hide other malicious software. [2,3]

As described in the challenge, our goal is to analyze the rootkit (reverse engineering) and disarm it to find the flag.

Solution

By opening the rootkit with Ghidra, we can guess that the initialization function is `diamorphine_init`, which in the assembly code is called as `init_module`. The `init_module` function is necessary for a kernel module to "start" when it is loaded into the kernel (`insmod`). Additionally, also the `cleanup_module` function (`diamorphine_cleanup`) is needed to remove the module from the kernel (`rmmod`). [4]

The goal of the `init_module` is to retrieve the pointer to the **system call table** and replace the `sys_getdents` and `sys_kill` functions with the rootkit's own functions.

```
int diamorphine_init(void)

{
  long lVar1;
  ulong *puVar2;
  int iVar3;
  long in_GS_OFFSET;
  ulong __force_order;

  lVar1 = *(long *)(in_GS_OFFSET + 0x28);
  __sys_call_table = get_syscall_table_bf();
  iVar3 = -1;
  if (__sys_call_table != (ulong *)0x0) {
    cr0 = (*_commit_creds)();
    module_hide();
    kfree(__this_module.sect_attrs);
    puVar2 = __sys_call_table;
    __this_module.sect_attrs = (module_sect_attrs *)0x0;
    orig_getdents = (t_syscall)__sys_call_table[0x4e];
    orig_getdents64 = (t_syscall)__sys_call_table[0xd9];
    orig_kill = (t_syscall)__sys_call_table[0x3e];
    __sys_call_table[0x4e] = (ulong)hacked_getdents;
    puVar2[0xd9] = (ulong)hacked_getdents64;
    puVar2[0x3e] = (ulong)hacked_kill;
    iVar3 = 0;
  }
  if (lVar1 != *(long *)(in_GS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return iVar3;
}
```

- `sys_getdents` and `sys_getdents64` is a system call that retrieves directory entries (`linux_dirent`) from a file descriptor. [5]
- `sys_kill` is a system call that sends a signal to a process.

The function `hacked_getdents` changes the behavior of the `sys_getdents` to hide a specific folder named `psychosis`. This name can be found by analyzing the hex value of `0x69736f68637973` (in ASCII, `isohcysp`) which is in little-endian format. Slightly below there is a check for the character `s` which when joined with the previous characters, forms the word `psychosis`.

```
LAB_00100245:
      uVar13 = (int)uVar6 - (uint)*(ushort *)((long)__dest + 0x1
      0);
      uVar6 = (ulong)uVar13;
      __n = (ulong)(int)uVar13;
      memmove(__dest,(void *)((long)__dest + (ulong)*(ushort *)(
      (long)__dest + 0x10)),__n);
    }
    else {
      if ((*(long *)((long)pvVar1 + 0x12) == 0x69736f68637973
      70) &&
         (*(char *)((long)pvVar1 + 0x1a) == 's')) {
        if (pvVar1 == __dest) goto LAB_00100245;
```

Once found out which is the folder we want to look for, we need to understand the behavior of the `hacked_kill` function.

We know that the `pt_regs->si` holds the signal number, the function behaves as follows:

- Signal `0x2a` (46): hide or unhide the module from the list of loaded modules (`lsmod` command).
- Signal `0x40` (64): escalate the privileges to the root user.
- Signal `0x1f` (31): hide or unhide a specific process from the list of running processes (`ps`). For example, `kill -31 <PID>` will hide the process with the given PID.

Now that we know how the rootkit behaves, we can disarm it by sending the appropriate signals to the rootkit. However, kernel modules do not have a PID, so we need to send the signals to any process by using `kill -<SIGNAL> -1`.

First we need to unhide the rootkit from the list of loaded modules:

```
~ $ lsmod
lsmod
~ $ kill -46 -1
kill -46 -1
~ $ lsmod
lsmod
diamorphine 16384 0 - Live 0x0000000000000000 (OE)
```

Then we can escalate our privileges and unload the module to make the `psychosis` folder visible:

```
~ $ whoami
whoami
whoami: unknown uid 1000
~ $ kill -64 -1
kill -64 -1
~ # whoami
whoami
root
~ # find / -name psychosis
find / -name psychosis
~ # rmmod diamorphine
rmmod diamorphine
~ # find / -name psychosis
find / -name psychosis
/opt/psychosis
~ # ls /opt/psychosis
ls /opt/psychosis
diamorphine.ko  flag.txt
```

Sources

- [1] Kernel Modules: https://linux-kernel-labs.github.io/refs/heads/master/labs/kernel_modules.html
- [2] Rootkits: https://www.malwarebytes.com/rootkit
- [3] Rootkits - 2: https://www.fortinet.com/resources/cyberglossary/rootkit
- [4] Init module: https://sysprog21.github.io/lkmpg/
- [5] sys_getdents: https://man7.org/linux/man-pages/man2/getdents64.2.html