PA1: Programming Assignment #1
CSCI 3753: Operating Systems
John Black

Due: Friday, Sep 19th, 11:55pm

Introduction

You can work individually or in groups of 2 to 3 people. No more than 3. If you work as a group, submit the project only once under your "leader's" moodle account. Here, the "leader" is the person least likely to forget, put it in the wrong place, submit the wrong thing, or ask for an extension due to whatever reason.

Get this done on time please! It's one of the more straightforward projects.

Description

We're going to play with a very simply rootkit for Linux. Read through the entire problem description and then break the tasks into assignments for each group member.

Resource Requirements

You need to be using 64-bit Ubuntu and you need root on the machine you're working on. One core and a little memory is fine here.

Setup

Obtain the C file called `rkit.c` from the moodle and put it somewhere under your home directory. Then you're ready to begin!

Instructions

This project is mostly about answering questions using clear and precise English sentences. There is a long series of questions below that you should endeavor to answer in a clear concise manner.

Important: It is fine to Google the answers to your heart's content. Read all you can and learn all you can, but do NOT cut and paste or otherwise copy other people's writing. I want this in your own words.

Here we go:

1. Read about "Loadable Kernel Modules." Try Wikipedia to start with and go from there as far as you like. Then write one paragraph on this: "what is one advantage LKMs have over directly modifying the kernel?"

2. `rkit.c` is an LKM. Figure out how to compile it (hint: you'll need to use extensions to make that understand `obj-m` targets). Give your makefile here and show one execution.

3. Read through the source of `rkit.c` and answer the following questions:

    a. We see `ssize_t` used in the code a few times. What is `ssize_t` and how does it differ from `size_t`?
    b. What does `asmlinkage` do?
    c. What does `copy_from_user()` do?
    d. What is the `sys_call_table`? Note that starting in Linux 2.6, the `sys_call_table` is no longer exported, so we can't just reference it. What does `rkit.c` do in order to get around this?
    e. In `rkit_write()` we see `kmalloc()` called instead of plain old `malloc()`. Why?
    f. The code uses `printk()` instead of `printf()`. Why? And where does the output go? (Please show evidence that the output really does go wherever you claim it goes!)
    g. In `rkit_init()`, what are the two calls to `write_cr0()` doing?
    h. Describe in one paragraph, the purpose of the `rkit_init()` function.
    i. Describe in one paragraph, the purpose of the `rkit_write()` function.

4. Take your compiled LKM and install it. This requires sudo, of course. Now, at a shell, type `echo h1dd3n`. Why doesn't that string appear?

5. Type `lsmod` and include the output here. Given what a rootkit is supposed to accomplish, what is the problem here?

6. Remove the module and type `echo h1dd3n` again. Explain why it now appears (you'll need to look at `rkit_exit()`.

7. Uncomment the first two lines of code in `rkit_init()`. Recompile and install the module again. Type `lsmod`. What has changed?

8. How do we remove the rootkit now?

9. Extend `rkit.c` so that calling `setreuid(1337, 1337)` will spawn a root shell. Note: this is probably the hardest part of the assignment, and you will likely want some help. The details are different on a pre-2.6 kernel and later kernels; our class VM has a 3.13 kernel.

Once you're done, print out the new function(s) you added and a demo of the modified rootkit in action.

Turn in all of the above by submitting two documents to the moodle:

1. The document which answers all my questions above and
2. Your modified `rkit.c` source file.