

Linux Loadable Kernel Module Based Rootkit Detector

Andrew Stone

Description:

The rootkit detector will be a loadable kernel module for the Linux kernel which detects both user space and kernel space rootkits. While user space rootkits can be detected by programs such as Tripwire, with binary hash checks, kernel based rootkits are more insidious. Kernel based rootkits usually do not modify the kernel executable file. Instead they patch the kernel in memory. While patching the kernel, the memory locations of system calls are either replaced with locations of the patched calls or the first few instructions of the system calls are replaced with jump instructions to the root kit code. This code often acts to hide the presence of certain files in the filesystem, malicious processes, as well as malicious sockets and other network activity.

The first type of kernel rootkit can be detected by comparison between the System.map file and the memory locations of the system calls in the running binary. This is done through the use of a debugger such as gdb, requires manual intervention and can be extremely tedious. It is also possible to automatically do this by comparing locations in System.map and /dev/kmem. There is no known way to easily detect the other type of kernel rootkit, in which jumps inside the syscalls are used to redirect to the rootkit code.

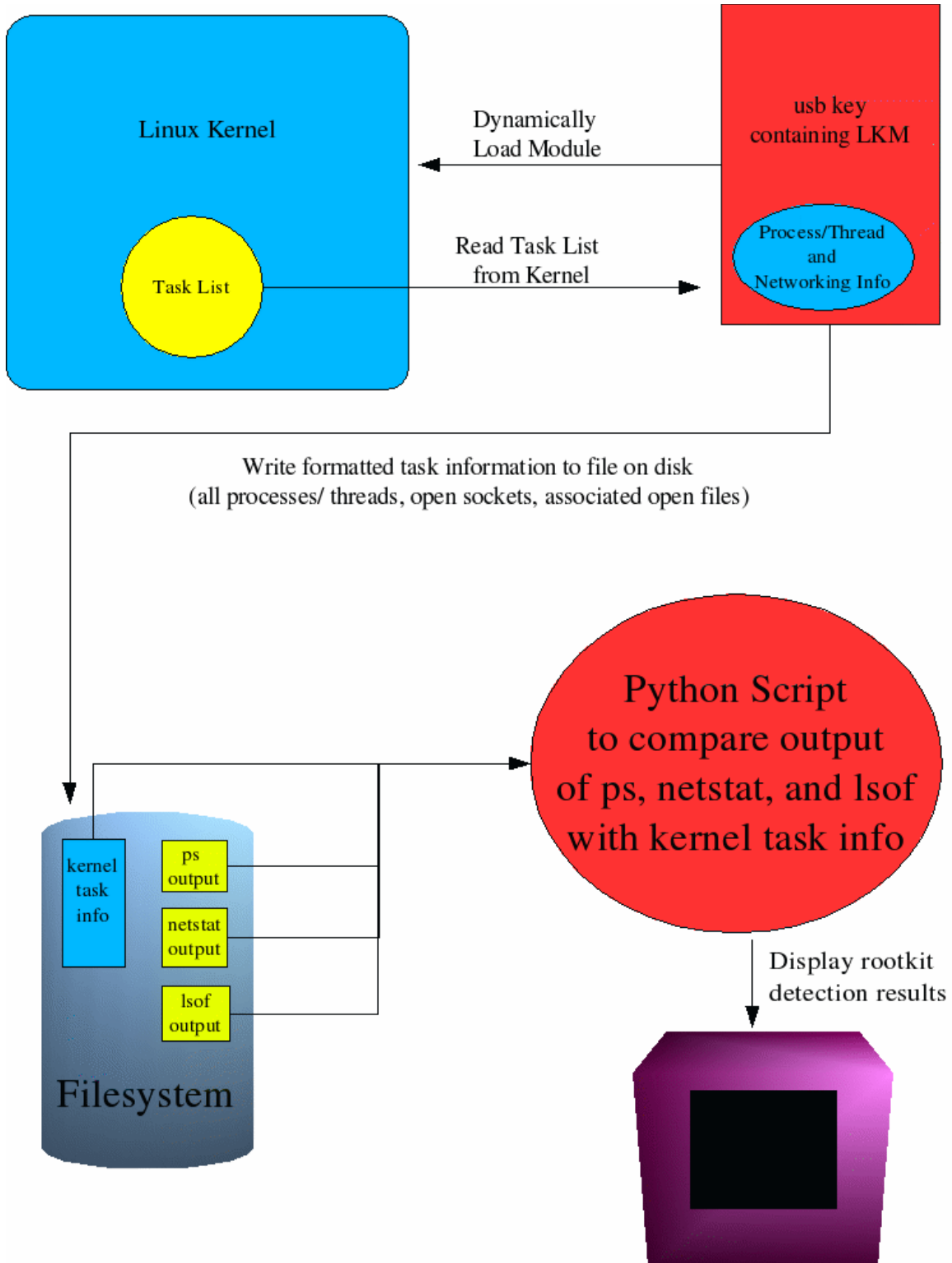
Thus a different method of checking rootkits must be used. We must go into the kernel and check the data structures themselves. By looking at the task list in the kernel, all processes and threads, as well as open file descriptors and network sockets associated with said processes can be identified. If unknown processes exist, or there are conflicts between what is in the task list and the output of programs such as ps, netstat, and lsof, we know that we have been subverted by a rootkit.

Implementation:

I will need to write a loadable kernel module for the Linux kernel. The kernel module will be loaded off a usb key and will gather information from the task list in the kernel. I will need to write the task list information to a file on disk. Information written will consist:

- all processes and threads running on the system
- what files are associated with what processes
- what ports are associated with what processes

I will also need to write a script in python which runs as root the commands **ps -eFL**, **netstat -aep -inet**, and **lsof** and writes this data to a file or files. The script will then compare the data collected from the kernel module with that from the output of ps, netstat, and lsof. Any discrepancies will be noted and shown to the user to indicate the presence of a rootkit. The following diagram shows the rootkit detection method and identifies the main components of the rootkit detection system.

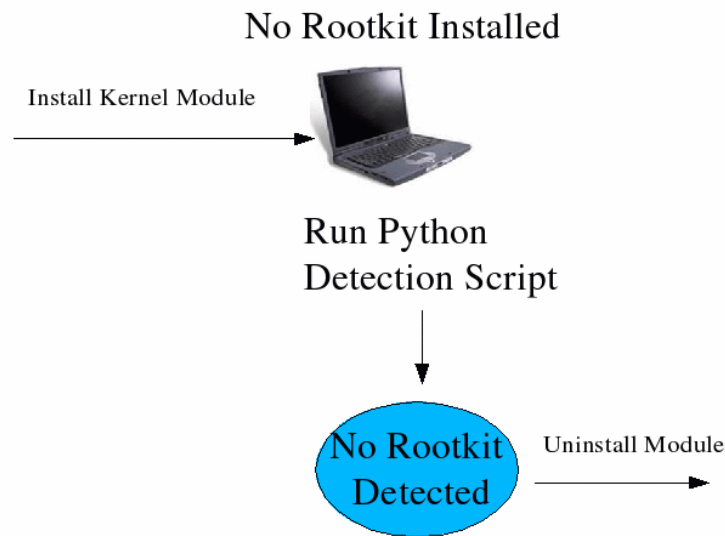


Demonstration:

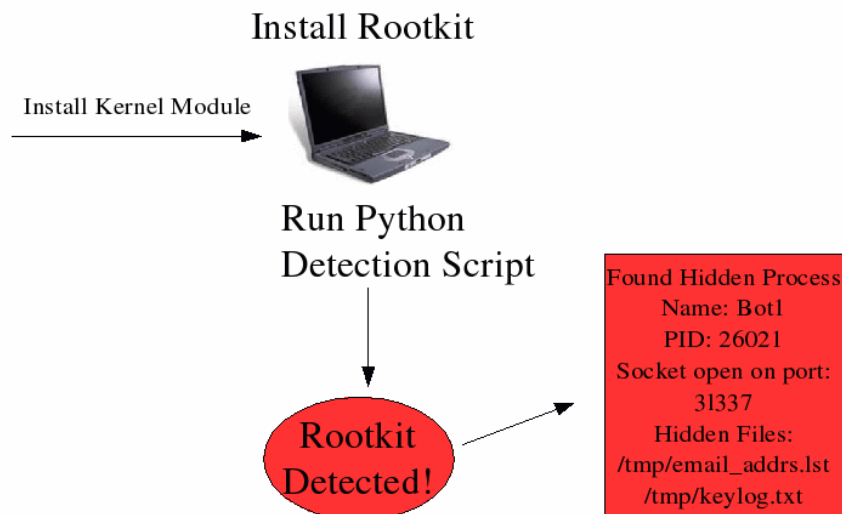
I will demonstrate my project on a Linux based laptop. I will install my rootkit detector off a usb drive as a Linux kernel module, and a python script. The initial configuration on the machine will not consist of a rootkit. When the rootkit detector is installed and run it will show that no rootkits have been detected. I will then uninstall the module. I will then install a rootkit, most likely a kernel based one, if I can find one that works on Linux 2.6. Currently I have the Adore rootkit, but it is for the 2.4 kernel. If I can not get a kernel based rootkit, I will use a user based one which replaces ps, netstat, and lsof with trojaned versions.

I will then re-install the rootkit detector module and run the detection script. The python script will acknowledge that a rootkit has been found and will list the processes, threads, open sockets, and files associated with the processes and threads that have been hidden via the rootkit. The diagram below will detail the setup.

Test1



Test2



<u>Equipment</u>	<u>Status</u>
1 laptop	Existing
1 usb drive	Existing
1 rootkit	Needed

References:

Linux Kernel Development, Robert Love

The Linux Kernel Module Programming Guide, Peter Jay Salzman, Michael Burian, Ori Pomerantz

Kernel Rootkits, Dino Dai Zovi

Runtime Kernel Kmem Patching, Silvio Cesare <http://vx.netlux.org/lib/vsc07.html>

Analysis of the t0rn rootkit, Toby Miller <http://www.sans.org/y2k/t0rn.htm>

<http://lwn.net/Articles/75990/>

Analysis of the Knark Rootkit, Toby Miller <http://www.ouah.org/tobyknark.html>

<http://www.ossec.net/rootkits/studies/knark-README.txt>