Summer Research Project Tentative Outline

Windows Rootkit Detection and Removal Algorithm

Christopher Wood

Faculty Mentor: Professor R. K. Raj

Department of Computer Science, Rochester Institute of Technology

April 13, 2009

Overview

To hunt down any virus or malicious program inside a computer system requires an intimate knowledge of how the program functions. It would be pointless for an anti-virus program to attempt to scan a computer for potentially harmful programs when the program does not know what it is looking for or where to look for it. Instead, anti-virus programs are able to effectively detect and remove malicious programs because they can successfully identify discrepancies and system processes and modifications that may trace back to such programs by looking in the appropriate places. The same logic must be applied to the detection and removal of rootkits. Essentially, a reverse engineering practice must be implemented to track the steps of a rootkit and remove it from the system. By knowing where the rootkit may reside, what it might be doing, how it operates, and why it is there to begin with one can, in theory, successfully locate and remove the rootkit. Therefore, it is my goal to play the role of both the rootkit and anti-rootkit program developer in order to deal with rootkits. By designing a rootkit that takes advantage of the Windows operating system and all it has to offer and then designing a method or algorithm to effectively counter such functionality step by step I believe I can root out many rootkits that are in the system.

However, the implemented rootkit and algorithms aren't the most important aspect of this project. While they are a necessary part of it, the purpose of this project is for me to formulate new ideas in the rootkit and computer security field in general. Therefore, during the implementation stage of this project I will spend a great deal of time on the conceptualization of new ideas that I may or may not have the ability or time to integrate into the working rootkit and rootkit detection process. It is my goal to publish such ideas in the form of a written and/or poster paper that I will also be working on throughout the course of the project to be presented at the RIT Undergraduate Research Symposium and any selected conferences that follow.

Rootkit Development Process

1.) Create a very basic, working rootkit to tie into the system.

2.) Add “userland” hooks to the rootkit to hide footsteps.

   a. Import Address Table (IAT) hook.

    b. Inline function hooking.
    c. Injecting a DLL into processes.

3.) Add kernel hooks so that the rootkit operates at the same level as the rootkit detection program.
    a. System Service Descriptor Table (SSDT) hook.
    b. Interrupt Descriptor Table (IDT) hook.
    c. Major I/O Request Packet function table in the Device Driver Object.

4.) Add runtime patching functionality to the rootkit (code injection, detours, etc.).
    a. Reroute program control flow.
    b. Use NonPagedPool Memory.
    c. Jump templates.

5.) Implement a layered driver approach to installing the rootkit for rootkit to easily manage and manipulate data.
    a. I/O Request Packet (IRP).
    b. Stack locations.

6.) Utilize Direct Kernel Object Manipulation (DKOM) to further hide the rootkit's presence.
    a. Process hiding.
    b. Device-Driver hiding.
    c. Handle synchronization issues (multi core CPU systems as well).

Anti-Rootkit Detection Development Process

1.) Develop an algorithm to detect both "userland" and kernel hooks by rootkits. This will involve an intricate scan of the system tables used by the operating system, e.g. the System Service Descriptor Table (SSDT).

2.) Develop an algorithm to monitor runtime execution detours and potential code manipulation.

3.) Develop an algorithm to detect non-authorized access to data between layered drivers in the system.

4.) If possible, develop a way to detect changes with kernel objects (such as the “removal” of a process in the EPROCESS linked list).