CMSC 412 - Fall 2010
William Arbaugh
11/19/2010

L A B 6 - C A P S T O N E

Introduction

The purpose of this lab is tie everything together and build an application that involves kernel and user land components..

In this lab, you will be building a kernel mode rootkit detector that uses both semantic integrity checks (Petroni Jr & Hicks, 2007), and crossview checks (Wang & Beck, 2005). You will need to use Native Windows API functions as documented in Nebbett (Nebbett, 2000). The TA's will have a copy of this book during the discussion sections for you. You may also want to review a copy of the Rootkits book by Hoglund (Hoglund & Butler, 2006).

Rootkit detection is not perfect. An adversary will usually be able to circumvent your detection methods once known and understood. I want you to learn how to research, design, and implement a realistic kernel mode driver and user-land application.

Assignment

You will build upon Lab 5 for this project. If you were unable to complete Lab 5, do so immediately. If needed, see me or a TA for assistance.

This lab is a phased. You will complete the sections needed to receive the desired grade. Completing the base requirements will result in obtaining a grade of at most 79 (C). Completing the next level will result in a grade of at most 89 (B), and completing the last level will result in a grade of at most 99 (A). Points may be deducted from the base grade based on style, performance, and build problems. Failure to complete the base requirements will result in a most a grade of 59 (F), and will likely result in a much lower grade depending on your submission. Obviously, the lack of a submission will result in a 0.

Your user-land application need only be a text based console application.

You may want to look at GMER (www.gmer.net) or rootkit revealer (http://technet.microsoft.com/en-us/sysinternals/bb897445.aspx) for examples of more sophisticated rootkit detectors.

Base Level Requirement

This requirement builds on previous labs to detect hidden processes. Recall from class room discussions, that an adversary may hide a process by "unlinking" the process from the process list. The scheduler, however, uses the thread list to schedule time on the CPU core(s).

To meet this requirement, you must detect hidden processes in the above manner. The technique to do so is walk the thread list and ensure that each thread has a corresponding entry in the process list.

Note 1: You MAY NOT use any native Window's API's to find and walk the thread and process lists.

NOTE 2: There is a race condition here. Think about how you might handle it to prevent false positives (locking is not the right approach here for performance reasons).

B Level Requirement

You must implement a hidden file detector using cross-view analysis by comparing the results of user-land Windows API's to those found with Window's kernel mode API's.

A Level Requirement

You must implement a hidden registry key detector using cross-view analysis by comparing the results of user-land Windows API's to those found with Window's kernel mode API's.

Testing

Programs (not malicious) will be provided in the discussion sections to allow you to test completion of your requirements. The actual grading will utilize these programs along with additional tests.

Submission

Submit the source for your user mode application, and driver along with ALL of your build/project files in a single tar file to the submit server. More information on the submission requirements will be provided by the TA's during discussions sections.

Grading

Meeting the requirement by passing the provided tests will result in at least the base grade. Failing additional tests,style, and/or build problems will result in the deduction of points not to exceed 10%.

Due Date

This project is due 12/10/2010 at midnight.

Bibliography

Hoglund, G., & Butler, J. (2006). *Rootkits : Subverting the windows kernel* (illustrated ed.). Upper Saddle River, NJ: Addison-Wesley.

Nebbett, G. (2000). *Windows NT/2000 native API reference circle series* (illustrated ed.). Indianapolis, IN: Sams Publishing.

Petroni Jr, N. L., & Hicks, M. (2007). Automated detection of persistent kernel control-flow attacks. In *Proceedings of the 14th ACM conference on computer and communications security.*

Wang, & Beck. (2005). Fast user-mode rootkit scanner for the enterprise. In *Proceedings of the 19th conference on large installation system administration conference - volume 19.* USENIX Association.