Software Security Assignment 1 - Trusted Path in the Linux System Kernel

Mats Authen

September 3, 2014

1 Introduction

1.1 The assignment

The assignment that was given:

1. Choose an operating system that does not have "Microsoft Windows" in its name, for instance Mac OS, Linux, iOS, Android, Multics, Plan 9, Turaya, Solaris
2. Find out and describe how the OS implements the concept of a trusted path (if it does).
3. Include references to the sources that support your findings (scientific articles, developer documentation).

I chose to find out how **Solaris** implements trusted path.

1.2 Trusted path definition

The most common way to define trusted path, is to say that it is a way for the system to authenticate itself for the user. The user can then be certain that he/she is interacting with the system safely, and not through for instance a false interface.
Linux is an open source operating system, or more specifically, a whole lot of operating systems with bigger or smaller differences, that's all based on the same kernel. Therefore, I will focus on mechanics in the kernel itself, as well as some well used versions and extensions.

2 Trusted Path in Linux

Trusted path is in no way implemented in the Linux kernel by default. There are a few methods for it though, but none of them is standardized. I will go through some of the common mechanisms further on.

3 Secure Attention Key

The secure attention key (SAK), is one of the mechanisms sometimes used in Linux to help implement the trusted path. SAK is a sequence of keys, that when pressed interacts directly with the kernel, enabling the system to authenticate itself.

The most known SAK is the one in windows: CTRL+ALT+DEL. In Linux however, there is no standard for this. There exists two variants, CTRL+ALT+PAUSE, and SysRq+K. It is not recommended to use SysRq+K, because in order for this to work, the kernel must be compiled with support for the SysRq key. Instead, CTRL+ALT+PAUSE is regarded as the best choice.

However, in Linux it is possible to customize the SAK by using "loadkeys", thereby eliminating problems related to SysRq.

4 What the SAK does

Effects of pressing the SAK vary from OS to OS. In Windows it takes the user to an isolated desktop, and gives a list of options, among them is logging out and locking the system. On Linux, the SAK will always work correctly when the keyboard is in raw mode. The SAK will then kill the X server, taking the user away from fakeable GUI, to the core system CLI. If the run level of the system is 5, the X server will be restarted as well. According to documentation, this is ideal.

5 Other alternatives

Some extensions and open source software exists that implement trusted path execution in Linux systems. One of these is grsecurity. The implementation of TPE in grsecurity allows an authenticated user to add a GID to a group of users that is "untrusted". These users can only attempt to execute/write files that are root owned and executable and writable only by

root (which they can never execute or write).

This is one of many security features provided by grsecurity. However, they do not provide any solution for trusted path or SAK.

6 Sources

- Secure by Design - Input Processing (UI) - Hanno Langweg
- The Linux Kernel Archives - Andrew Morton
  https://www.kernel.org/doc/Documentation/SAK.txt
- grsecurity: Documentation - grsecurity dev team
  http://en.wikibooks.org/wiki/Grsecurity/Appendix/Grsecurity_and_PaX_Configuration_Options#Trusted_Path_Execution_.28TPE.29