Anna Kaardal
120918
Obligatory exercise #1
2014-09-02

Trusted path in the Linux Operating System

Introduction

A trusted path is a mechanism which lets the system authenticate to the user. This means that the user can make sure that the input he or she types in, is not read by unauthorized programs, which for example claim to be the log on screen for the user account, but in reality is trying to trick the user into giving away information. An well known example for a trusted path, is the key combination (ctrl+alt+delete) in Windows, which can not be imitated by an malicios program, and takes the user right to the Windows Security Screen.

In this assignment I have choosed to take a closer look at the operating system Linux`s implemetation of trusted path. Unlike Windows, Linux does not have a trusted path for the login procedure[1]. However, there are alternatives.

Secure Attention Key (SAK)

A secure Attention Key is a key sequence which provides secure communication between the user and the system. SAK is not implementet by default in Linux, but there are different ways to do this. When implemented, typing in the SAK will make the x-server reboot and programs that might claim to be the log on procedure, will be killed. A superuser has the opportunity to modify the rc.sysinit or rc.local file and set a specifisc key sequence as the SAK [2]. When modifying, it is important to choose a key sequence that does not interfer with other tasks. For example, the ctrl+alt+delete sequence is already used by Linux for rebooting. A good practice is to use the ctrl+alt+pause sequence, which does not interfere with other tasks. Notice that SAK might kill processes that were not intended to. This might be unfortunate to the user. There is also possible to use setserial from the command line. Setserial has to be installed first, followd by typing in the *setserial sak* command. This will set the break button as SAK[3].

The Linux Security Modules project and Trusted Path Execution (TPE)

The Linux Security Modules (LSM) project provided the Linux kernel with security models. One of theese models is The Trusted Path Execution (TPE)[4]. TPE monitors execution of files and checks if the parent directory of the file is owned by root[4]. If it is, the path is considered trusted. TPE does also use Trusted User ACL[4] to define which users who are considered trusted.
If neither the path nor the user can be considered trusted, access to program execution will be denied. This helps preventing potentially malicious code from running on the system.

User\Path	**Trusted**	**Untrusted**
Trusted	Execution allowed	Execution allowed
Untrusted	Execution allowed	Execution declined

The table above illustrates TPE[4].

Other security mechanisms in Linux
There are also other security mechanisms that might be implemented in the kernel. Two of these, which are both mandatory access control (MAC) mechanisms implemented in the kernel:
- AppArmor, which is a Linux application security system with numbers of default security policies included, and which defines which resources an application will be allowed to access[5]. From Ubuntu 7.10, AppArmor is installed by default[7].
- Security Enhanced Linux (SeLinux) which is a mandatory access controll (MAC) mechanism implemented in the kernel[6]. The users do not get more privileges than they need to be able to perform their tasks, and in that way preventing users (on purpose or by a mistake) from running malicious code.

Conclusion
Linux does not have a secure path, but there are other alternatives, like Secure Attention Key, which communicates directly with the kernel and kills all programs that might try to fool the user, and Trusted Path Execution, which prevents malicious code from running by checking the path and user for trustworthiness. There are also other security mechanisms, like AppArmor and SeLinux, which both implements mandatory access control. Most of theese alternatives are not implemented by default, but anyway, when implemented, they provide some security mechanisms which help protect the system from intruders.

References
(next page).

1 Secure programming for Linux and Unix HOWTO , Set up a Trusted Path, http://www.dwheeler.com/secure-class/Secure-Programs-HOWTO/trusted-path.html, 26.08.14
2 Andrew Morton, Secure Attention Key (SAK) handeling, https://www.kernel.org/doc/Documentation/SAK.txt , 26.08.14
3 Linux man page, setserial, http://linux.die.net/man/8/setserial , 30.08.14
4 Niki A. Rahimi, Trusted Path Execution for the Linux 2.6 Kernel as a Linux Security Module, https://www.usenix.org/legacy/event/usenix04/tech/freenix/full_papers/rahimi/rahimi.pdf, 02.09.14
5 Main Page - AppArmor http://wiki.apparmor.net/index.php/Main_Page, 02.09.14
7 Chris Hoffman, HTG Explains: WhatAppArmor Is and How It Secures Your Ubuntu System, http://www.howtogeek.com/118222/htg-explains-what-apparmor-is-and-how-it-secures-your-ubuntu-system/ , 02.09.14
6 SeLinux http://wiki.centos.org/HowTos/SELinux, 02.09.14