Implementation of the Trusted Path concept in the Linux Kernel

141627

Oberdorf, Urs

Gjøvik University College

Faculty of Computer Science and Media Technology

IMT 3501 Software Security, Hanno Langweg

Obligatory exercise 01

Autumn semester 2014

Introduction

There are two realizations of the Trusted Path concept for the Linux kernel. One is to secure the login process through a key combination which gets immediately directed to the kernel, the Secure Attention Key. The other one puts itself into effect through a logical file and directory property check, the Trusted Path Execution.

The Secure Attention Key

The idea behind the Secure Attention Key, in short SAK, is to implement a key signal which gets straight to the kernel without the chance of being intercepted by an application. The Background of this idea is a counter measure to the appearance of malware applications which spoofs a login screen to the user and thereby obtains the users password by fraud. Pressing the SAK would redirect the user to the systems login screen on an ensured and trusted path.

Linux approaches the implementation of the SAK in two ways. [1]
The first happens through the Magical System Request. [2] This key provides useful kernel communications in case of an unresponsive program, for example in an hanging X session, the user can sync the devices, unmount them and reboot the system. It should be pointed out here that for this approach, the operating kernel has to be compiled with SysRq. This implementation is used by Debian [3] for example.
The second approach, in which the kernel compile option SySRq is not required, is the control + alt + pause keystroke to execute the SAK signal.

The Trusted Path Execution

This method of securing the system from malware was introduced in 1998 by a kernel patch [4] and extended a few month later [5] by a trusted user feature. After the Linux Security Modules framework was introduced in 2001, [6] it later became part of the grsecurity patch set. [7] There are also Linux Security Modules which implement security extensions to the kernel, like SELinux or AppArmor, but neither SELinux nor AppArmor support Trusted Path Execution. [8]

The basic function is a hook on every file execution a user does. It will check if the parent directory of the executable file is owned by root and if the user is in a list of trusted users. Then the execution is determined by the previous check results. Possible results are trusted user & trusted path, untrusted user & trusted path, trusted user & untrusted path and untrusted user & untrusted path. In the first three cases the execution permission is given, not so in the last and it will result in an error -EACCES. [9]

An example use case shows which goals are achieved by implementing the Trusted Path Execution patch. The user Linus is not on the list of trusted users. He downloads a source code with malicious or buggy software and compiles it into a binary executable in his home directory. When he wants to execute the binary, he will receive an error stating that this is not a trusted path and prevents him from accidentally or intentionally crashing the system. Linus can now ask the system administrator to get an entry for himself in the trusted user list or to put the executable in a directory owned by root for successfully executing it and crashing the system.

[1] Andrew Morton. The Linux Kernel Archives [Internet]. San Jose,California: Linux Kernel Organization, Inc.; March 18th 2001 [cited august 31st 2014]. Available from: https://www.kernel.org/doc/Documentation/SAK.txt

[2] Mydraal(pseudonym). The Linux Kernel Archives [Internet]. San Jose,California: Linux Kernel Organization, Inc.; [date unknown] [updated January 1st 2001; cited august 31st 2014]. Available from: https://www.kernel.org/doc/Documentation/sysrq.txt

[3] Osamu Aoki. Debian Reference version 2. [place unknown]: The Debian Project; 2013 [cited august 31st 2014]. Chapter 9.3.15 Alt-SysRq key. Available from: https://www.debian.org/doc/manuals/debian-reference/ch09.en.html#_alt_sysrq_key

[4] daemon9(pseudonym). Hardening the Linux Kernel. Phrack Magazine. January 26th, 1998: Volume 8 (Issue 52) article 06 of 20. Available from: http://phrack.org/issues/52/6.html#article

[5] Krzysztof G. Baranowski. Linux Trusted Path Execution Redux. Phrack Magazine. July 8th, 1998: Volume 8 (Issue 53) article 08 of 15. Available from: http://phrack.org/issues/53/8.html

[6] Crispin Cowen. MARC Mailing list ARChives [Internet]. KoreLogic Data Center: MARC; April 11th 2001 [cited August 31st 2014] Available from: http://marc.info/?l=linux-kernel&m=98695004126478&w=2

[7] grsecurity [Internet]. [place unknown]: Open Source Security Inc.; [date unknown][cited august 31st 2014]. Available from: https://grsecurity.net/features.php#tpe

[8] grsecurity [Internet]. [place unknown]: Open Source Security Inc.; [date unknown][cited august 31st 2014]. Available from: https://grsecurity.net/compare.php

[9] Niki A. Rahimi. USENIX '04 – Technical Paper, General Track [Internet]. Boston, MA, USA: 2004 USENIX Annual Technical Conference; June 27th – July 2nd 2004 [cited august 31st 2014]. Available from: https://www.usenix.org/legacy/event/usenix04/tech/freenix/full_papers/rahimi/rahimi_html/index.html