SYNNE GRAN ØSTERN

120922

OBLIGATORY EXERCISE 1

2014-09-02

Trusted path in Linux

Operating system uses the trusted path to ensure no unauthorized program access and reads data form a terminal. Trusted path is also used when the system require a secure communication with the user. Typical example of this kind of communication is when a user wants to change the password.[1]

I have chosen to write about the trusted path solutions on Linux.

Secure Attention Key (SAK)

The Secure Attention Key on Linux OS can be regarded as the Linux version of the Windows "Ctrl+Alt+Del" keys. A special key combination must be pressed to enter the login screen. When the OS detects this sequence, is will start the trusted login process. According to the *Linux 2.4.2 Secure Attention Key (SAK) handling documentation* it is alleged that it is "an undefeatable way to kill all running programs which could be masquerading as login application."[2] To engage the SAK on Linux the user has to press one of two key sequences; ALT+SYSRQ+K sequence and the other way is to define the key sequence by using the loadkeys. The last way is what Andrew Morton recommends because the ALT+SYSRQ+K sequence only works if the kernel ware compiled with the "SysRq" support.[3] "The Magic SysRq key" makes it possible to give directly commands to the kernel by using a sequence of keys. However, this feature is also a security risk, by letting some unauthenticated key sequences to be pressed and the computer can shut itself down. Therefore, it is crucial to limit the console in an area of trusted people. It is also possible to disable the SysRq key.[4]

When the SAK is working correctly, it will kill all application on the x server, and if the run mode is at level 5, it will shut down. This does include the one that uses the /dev/console directory. This may include several applications that a user not necessarily want to kill. [5]

Trusted Path Execution (TPE)

The Trusted Path Execution Linux Security Module (LSM) were created to prevent "innocent" or ignorant users to run malicious or bad code and cause major damages to the system. TPE protects the system from the executions files. The kernel makes a quick check on the user credentials and

[1] Trusted path, trusted shell and secure attention key, IBM Knowledge Center. http://www-01.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.security/trusted_path_shell_sak.htm

[2] LINUX 2.4.2 Secure Attention Key(SAK) handling, Andrew Morton. 18. March 2001 https://www.kernel.org/doc/Documentation/SAK.txt

[3] Same as above

[4] The Magic SysRq key, Remote Serial Console HOWTO, http://tldp.org/HOWTO/Remote-Serial-Console-HOWTO/security-sysrq.html

[5] LINUX 2.4.2 Secure Attention Key(SAK) handling, Andrew Morton. 18. March 2001 https://www.kernel.org/doc/Documentation/SAK.txt

verifies that the exe-file is not being run on a vulnerable path on the system. If the TPE finds a case where is not acceptable it will return an error message. If the check verifies that the path the exe-file is running on is trusted and the user who runs the file is also trusted, no error message is returned. However, if both checks returns a untrusted value, then the exe-file will be failed. Both values has to be untrusted for the TPE to return an error message. The figure under show this relation.[6]

Path **User**	**Trusted**	**Untrusted**
Trusted	Execution allowed	Execution allowed
Untrusted	Execution allowed	Execution not allowed

The Linux Security Module (LSM) is a framework with several basic security modules for the kernel. Alone this framework does not make the OS more secure, it rather provides the infrastructure for the modules to function.[7]

Other solutions

I have briefly explained two trusted path solutions in Linux. I have not however gone specific into certain Linux distro like Red Hat and Ubuntu. These distros uses for instance SELinux and AppArmor for mandatory access control and other security solutions. These solutions also use the Linux Security Modules framework and are widely used today.[8]

[6] Trusted Path Execution for the Linux 2.6f Kernel as a Linux Security Module, Niki A. Rahimi. 10. June 04. https://www.usenix.org/legacy/events/usenix04/tech/freenix/full_papers/rahimi/rahimi_html/

[7] LSM Overview, Implementing SELinux as a Linux Security Module. https://www.nsa.gov/research/_files/selinux/papers/module/x45.shtml

[8] SELinux FAQ, fedoraproject. http://fedoraproject.org/wiki/SELinux_FAQ