COMP/ELEC513 Class Project

Complexity Analysis of Linux Security Modules

He Jiang

5 pages

Abstract

My project is focused on Linux Security Modules (LSM), specifically its relationship with the Linux kernel, the structure of its implementation modules, and its temporal evolution.

1. Introduction and Background

Linux is the one of the most notable and complex open-source contribution software to date. The vanilla Linux Operating System inherits Discretionary Access Control (DAC) from Unix. Linux Security Modules (LSM) was introduced in 2002 as a general purpose framework to unify functional needs of concrete implementations.(4) It effectively serves as a hook to the rest of the kernel, ranging from memory management to booting control.

SELinux, Security-Enhanced Linux, is a popular implementation of LSM, designed to be minimally impacting while extending access control abilities.

2. Motivation

I am currently in a computer security class, so I am interested in investigating the complexity related to security. I found Linux Security Modules unique because it is a lightweight interface between the main kernel, a gigantic codebase having evolved for almost 30 years, and multiple implementation modules by a variety of maintainers, from the NSA (SELinux) to business-to-business software companies (AppArmor).(6)

3. Key Findings

3.1 "Security" before and after LSM

Linux security module was introduced into mainline Linux kernel version 2.6.0 in December 2003. As a simplified way to highlight the impact of this new module, I performed a word count of "security" and "secure" over the .c and .h files in Linux kernel version 2.5.9, 2.6.0 and 2.6.0 without the /security subdirectory. The main difference between using a word count from a full-fledged parser is that we mostly examine the words from comments, since "security" embedded in a variable or function name is not considered to be a "word" separated by whitespaces.

Linux version	Count
2.5.9	136
2.6.0	507
2.6.0 w/o "security"	330

From the word count, "security" increased significantly from 2.5.9 to 2.6.0. I think this can be attributed to that developers were aware that Linux would be introducing LSM and documented their modifications on the code.

3.2 LSM and the rest of kernel

To demonstrate how LSM has changed from the original 2.6.0 to the most current 4.10.13, I created an #include graph for each version. The images in the report may be too small, please refer to the directory /graph for Postscript graphs. In both graphs, the nodes in the top row are non-LSM files, and those in the bottom row are LSM files.

In the first version, LSM only had five .c files, and all of them together included 17 .h files from the rest of the kernel. It still covered a wide range of functionalities: "swap.h", "pagemap.h", "mm.h" in memory management, "net/sock.h" and "netlink.h" in networking, "usb.h" in device model, and "file.h" in file systems. At the time, SELinux was the only implementation accepted into the mainline kernel.

In the current version, LSM has six .c files, but they include 59 .h files together. The scope of security it provides

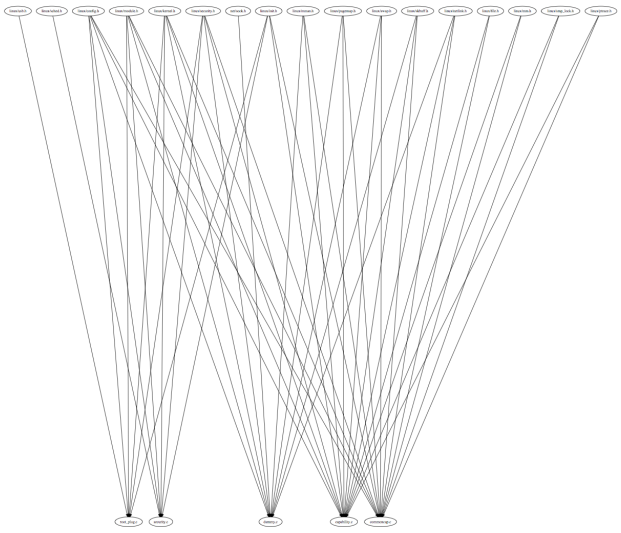

Figure 1. #include graph of directory LSM in Linux 2.6.0

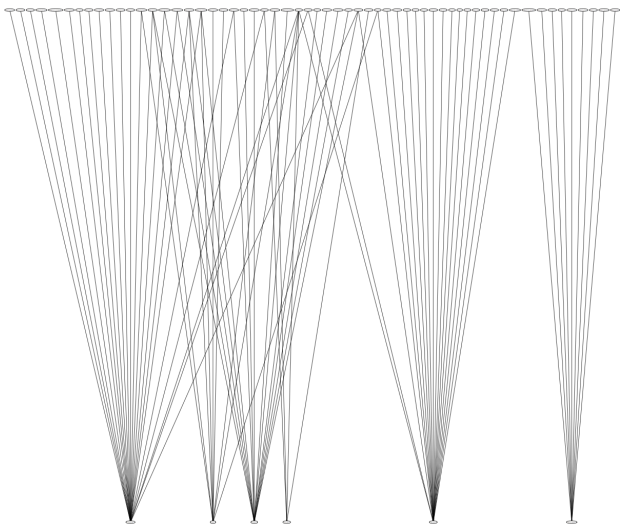

Figure 2. #include graph of directory LSM in Linux 4.10.13

has also expanded. For example, LSM now includes networking protocols such as "ipv6.h" and thread management files such as "mutex.h".

3.3 Structure of different LSM implementations

As of Linux 4.10.13, there are 5 accepted implementations of LSM. I constructed an #include graph for each of them and found them to have dramatically different structures. SELinux and AppArmor are similar. They were both initially released in 1998 and integrated very early in the kernel, and they were both originally developed by organizations (NSA) and/or software companies (RedHat, SUSE). (6) (7)

The TOMOYO graph has a funnel structure. The central node is "common.h", which takes in the many .h files from other parts of the kernel and then delegate to .c files in the bottom layer, which are parts of the TOMOYO module. In SELinux and AppArmor modules, there are no such clear division of work between .c files. In AppArmor, "lsm_hooks.c", which contains all the function hooks into the Linux kernel, is included by lsm.c along with many other "include"s. In contrast, the same file is only included by "tomoyo.c" along with "common.h". TOMOYO was launched in 2003 and merged into mainline Linux kernel 2.6.30 in 2009. If I were a system administrator, I would find this software design easier to comprehend. (8)

SMACK and yama were initially developed by the same author, Casey Schaufler, and initially released in 2008. (9) SMACK is "committed to the premise that security does not have to be complicated". (10) From the #include graph comparing to the dense graphs of other systems, it does seem they have fulfilled their promise. Similar to TOMOYO, they also implemented division of labor, between filesystem and networking security, for example. Besides desktop Linux systems, it has also been used in mobile OS (MeeGo) and its continuation Tizen, embedded devices, and TV. MeeGo was released in 2010. (11) Android, a major player in the mobile OS market, didn't adopt SELinux until Android 4.3 in 2012. This may be attributed to SMACK's simplicity comparing to the gigantic SELinux. (12)

yama is a tiny module designed to "collect system-wide DAC security protections that are not handled by the core kernel itself". (13) It was introduced the latest into the kernel. From a software design perspective, I think these 5 implementations are very diversified for different needs.

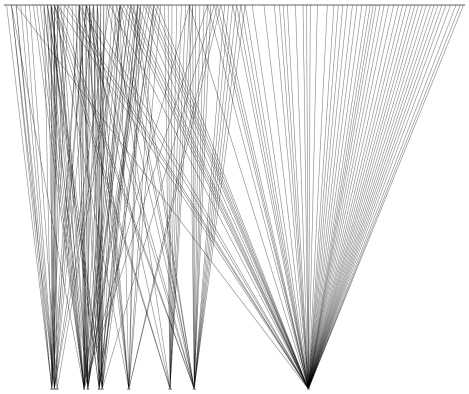

Figure 3. #include graph of SELinux in Linux 4.10.13

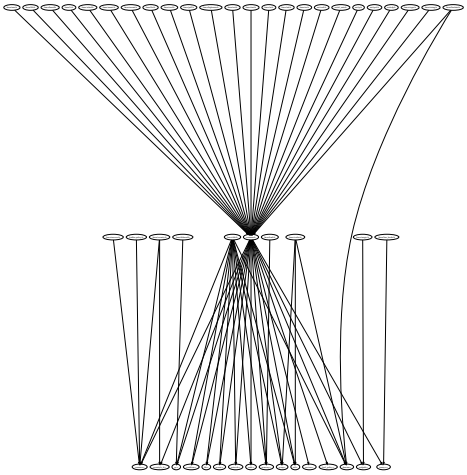

Figure 5. #include graph of TOMOYO in Linux 4.10.13

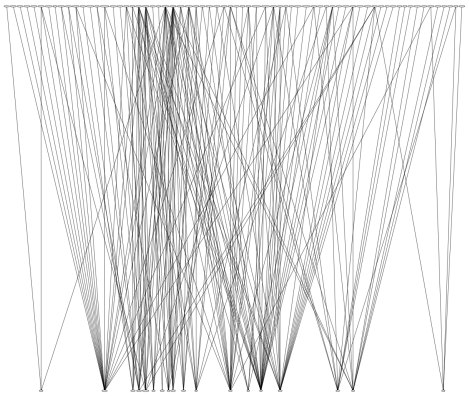

Figure 4. #include graph of AppArmor in Linux 4.10.13

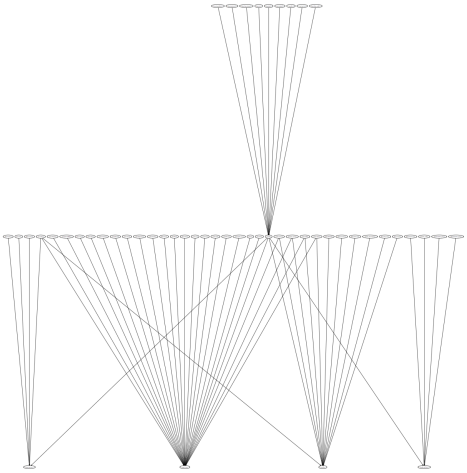

Figure 6. #include graph of SMACK in Linux 4.10.13

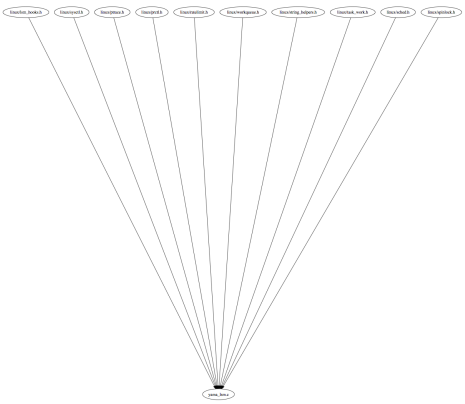

Figure 7. #include graph of yama in Linux 4.10.13

3.4 Temporal evolution of LSM

Over the years, the number of contributors to LSM has increased and the structure has become slightly simpler.

3.5 Methods

3.5.1 Word Count

First, I extracted all the .h and .c files from Linux source package using shell commands.(3) I wrote a Java program based on a class assignment I did last year. (15) It utilizes the Java Apache Spark framework for parallel MapReduce computing. (16). I found this to be more straightforward than cscope. (22)

3.5.2 #include graph

Initially I used cinclude2dot, an open source perl script to create an #include graph, but it does not search .h files outside of the given directory. (17). I wrote a script in Python to generate a DOT representation of the graph. (18) Then I used graphviz to visualize the DOT graph. (19).

3.5.3 Temporal evolution of LSM

I used Gource, an open-source software version control visualization tool. (20)

3.6 Future Work

I spent a few hours on pfff, a multi-functional code analysis tool open sourced by Facebook. (21). I was trying to use it to produce a treemap which illustrates the spatial relationships and dependencies between modules in constrained space, so it is very suitable for Linux kernel. However, I failed to get it to build the linux kernel because of parse errors. The last

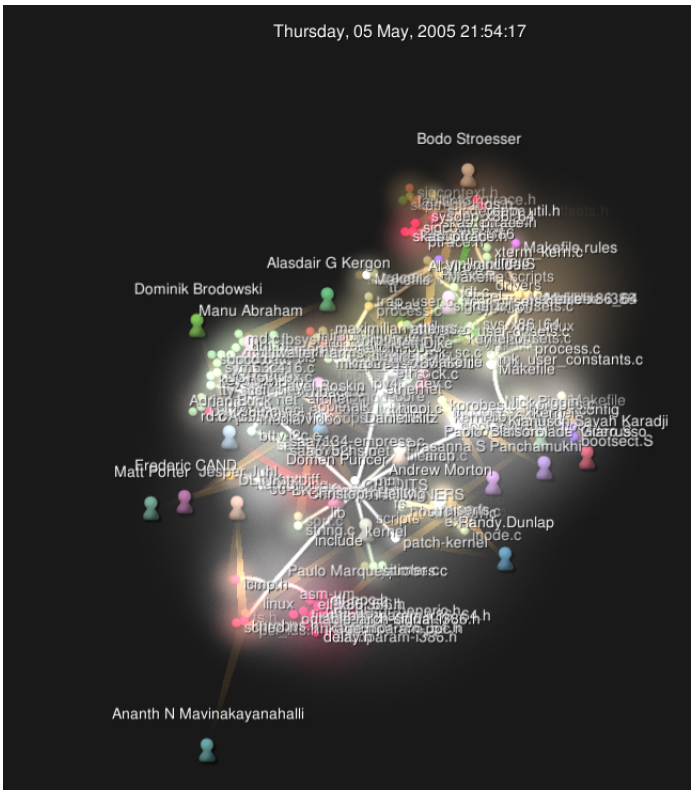

Figure 8. A Gource visualization of an early commit in the Linux Git repository

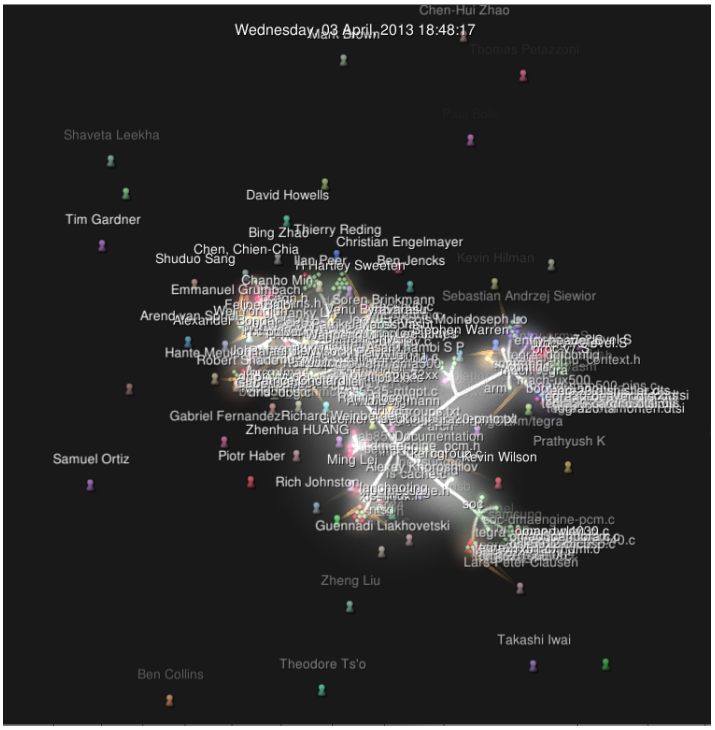

Figure 9. A Gource visualization of a more recent commit in the Linux Git repository

major commit for pfff was a few years ago so it may have not taken into account some changes in C, or the kernel should be built in a different way. I would like to dive more into the source code of pfff in ocaml and maybe modify it to make it work with kernel 4.10.13.

References

[1] Github: Linux, https://github.com/torvalds/linux

[2] Linux Cross References, http://lxr.free-electrons.com/

[3] The Linux Kernel Archives, https://www.kernel.org/

[4] Aleix M. MartõÂnez, and Avinash C. Kak, *Linux Security Modules: General Security Support for the Linux Kernel*. Proceedings of the 11th USENIX Security Symposium, August 2002

[5] Wikipedia: Linux Security Modules, https://en.wikipedia.org/wiki/Linux_Security_Modules

[6] Wikipedia: AppArmor, https://en.wikipedia.org/wiki/AppArmor

[7] Wikipedia: SELinux, https://en.wikipedia.org/wiki/Security-Enhanced_Linux

[8] Wikipedia: Tomoyo, https://en.wikipedia.org/wiki/Tomoyo_Linux

[9] Wikipedia: SMACK, https://en.wikipedia.org/wiki/Smack_(software)

[10] The SMACK Project, http://schaufler-ca.com/

[11] Wikipedia: MeeGo, https://en.wikipedia.org/wiki/MeeGo

[12] Security-Enhanced Linux in Android, https://source.android.com/security/selinux/

[13] Yama, https://www.kernel.org/doc/Documentation/security/Yama.txt

[14] Add secure bit to prevent set[ug]id executables from exec()ing, https://lwn.net/Articles/368600/

[15] COMP 322: Fundamentals of Parallel Programming (Spring 2017), https://wiki.rice.edu/confluence/display/PARPROG/COMP322

[16] Apache Spark, http://spark.apache.org/

[17] Github: cinclude2dot, https://github.com/frabcus/cinclude2dot

[18] Drawing graphs with dot, http://www.graphviz.org/Documentation/dotguide.pdf

[19] Graphviz, http://www.graphviz.org/

[20] Github: Gource, https://github.com/acaudwell/Gource

[21] Github: pfff, https://github.com/facebook/pfff

[22] Cscope, http://cscope.sourceforge.net/