دانشگاه صنعتی شریف
دانشکدهٔ مهندسی کامپیوتر

قفل‌کردن هسته (Kernel Lockdown)

درس سیستم عامل

امیرمهدی نامجو

نیم‌سال دوم تحصیلی ۰۰ـ۹۹

معرفی و انگیزه ایجاد این ویژگی

قفل‌کردن هسته [1] یکی از ویژگی‌های جدید هسته سیستم‌عامل لینوکس [2] است که در نسخه 5.4 به آن اضافه شده است. هدف از ایجاد این ویژگی، قراردادن مرز مجکم‌تری بین بین فضای کاربر [3] و فضای هسته [4] از طریق محدودسازی کاربر ریشه [5] است. [۱] کاربر ریشه که با UID 0 مشخص می‌شود، به طور پیش‌فرض سطح دسترسی بالایی به تمام سیستم داشته و حتی امکان ویرایش هسته را هم دارد. [۲]

فناوری‌هایی نظیر UEFI Secure Boot هم به همین منظور ایجاد شده‌اند تا اطمینان حاصل کنند که یک سیستم قفل شده، تنها برنامه‌هایی را اجرا می‌کند که توسط منبعی معتبر امضا شده باشد. با این وجود، از آن جایی که کاربر ریشه امکان ویرایش کد هسته را دارد، عملا نمی‌توان چنین عملکرد درستی را تضمین کرد و اگر کنترل کاربر ریشه از دست صاحب اصلی سیستم خارج شده باشد یا کد مخربی را اجرا کند، امنیت سیستم به خطر می‌افتد. [۳]

مکانیزم قفل‌کردن هسته از همین رو ایجاد شده است تا در صورت فعال‌سازی در سیستم‌هایی که امنیت آن‌ها اهمیت بالایی دارد، بخشی از دسترسی‌های کاربر ریشه را هم غیرفعال کرده و اجازه تغییرات غیرمجاز در کدهای هسته را ندهد. بدین ترتیب مرز بین فضای کاربر و فضای هسته مستحکم‌تر شده و در سیستم‌هایی که این مکانیزم در آن‌ها فعال باشد، نمی‌توان برخی از تغییرات را حتی به کمک کاربر ریشه انجام داد. [۳]

جزئیات و تاریخچه پیاده‌سازی

تاریخچه مختصر

اولین نکته‌ای که در مورد این ویژگی حائز اهمیت است، این است که از حدود سال ۲۰۱۲ زمزمه‌های پیاده‌سازی آن وجود داشته ولی به دلایل حواشی فراوان پیرامون محدود کردن سطح دسترسی کاربران و همچنین بحث پیرامون تاثیرگذار یا تاثیرگذار نبودن آن، این موضوع به نسخه نهایی هسته راه‌نیافته بود؛ اما ویژگی‌های مشابهی در توزیع‌های مختلف لینوکس برای عملکردی مشابه پیاده‌سازی شده بودند. [۴]

یکی از دلایل این حواشی مربوط به این می‌شود که در هسته لینوکس، مکانیزمی برای اعمال سیاست‌های امنیتی در قالب Linux Security Module قرار گرفته بود. این مکانیزم دست کاربر را برای اعمال محدودیت‌های امنیتی مختلف باز می‌گذارد. با این وجود مشکلی که وجود دارد، این بخش از فضای کاربر قابل اعمال است اما چنین سازوکاری که قرار است جلوی ویرایش نابه‌جای هسته را بگیرد، باید پیش از این که بتوان سیاست امنیتی خاصی را اعمال کرد، فعال بشود.

جزئیات پیاده‌سازی

در نتیجه مواردی که در قسمت قبل گفته شد، نوعی ماژول امنیتی مقدماتی در هنگام بالاآمدن [6] سیستم ایجاد می‌شود تا قلاب‌های امنیتی [7] اولیه لازم برای مکانیزم قفل کردن هسته ثبت بشوند. این قلاب امنیتی در پرونده

[1] Kernel Lockdown
[2] Linux Kernel
[3] User space
[4] Kernel Space
[5] root
[6] Boot
[7] Security Hook

lsm_hooks.h تعریف شده و در پرونده security.c در تابع security_locked_down فراخوانی می‌شود.

```
//lsm_hooks.h
1820: int (*locked_down)(enum lockdown_reason what);

//security.c
2402: int security_locked_down(enum lockdown_reason what)
{
    return call_int_hook(locked_down, 0, what);
}
```

عملا ایجاد این قلاب امنیتی به هسته اجازه می‌دهد حالت‌های مختلف نامعتبر بودن را که در توابع مختلف امنیتی مشخص شده‌اند، به درستی چک کند.

فعال‌سازی حالت قفل‌سازی هسته به شکلی است که علاوه بر این که بعد از اجرا شدن می‌توان آن را فعال کرد [۵] ، در هنگام بالا آمدن هم می‌توان آن را به عنوان پارامتر مشخص کرد. همچنین در هنگام ساختن [۱] کد هسته هم می‌توان آن را در پرونده‌های KConfig مشخص کرد.

به طور کلی سه حالت برای قفل‌سازی هسته وجود دارد. LOCKDOWN_NONE که عملا این مورد را فعال نمی‌کند. حالت LOCKDOWN_INTEGRITY_MAX که جلوی عملیات‌هایی که منجر به تغییر غیرمجاز در هسته (عموما از سمت کاربر ریشه) می‌شوند را گرفته و حالت LOCKDOWN_CONFIDENTIALITY_MAX که حتی امکان افشای داده‌های سطح هسته را هم می‌گیرد. با این وجود باید توجه کرد که در اصل حالت‌های خیلی بیش‌تری وجود دارند که همه آن‌ها در قالب enum lockdown_reason در پرونده Security.h قرار گرفته‌اند. به دلیل ترتیبی بودن enumها، عملا فعلاسازی LOCKDOWN_INTEGRITY_MAX باعث فعال شدن همه حالت‌های امنیتی پیشین آن نظیر LOCKDOWN_KEXEC هم می‌شود. [۶]

به عنوان مثال یکی از تغییراتی که در این نسخه داده شده است، مربوط به پرونده kexec_file.c است:

```
    static int kimage_validate_signature(struct kimage *image){
    ...
        if (!ima_appraise_signature(READING_KEXEC_IMAGE) &&
            security_locked_down(LOCKDOWN_KEXEC))
            return -EPERM;
    ...
 }
```

این تابع در صورتی که امضای پرونده توسط واحد ima [۲] تایید نشود و حالت LOCKDOWN_KEXEC هم فعال باشد، خطای مجاز نبودن عملیات (EPERM) می‌دهد. در این جا لازم است به تغییراتی که در پرونده ima_policy.c داده شده است هم اشاره کنیم. در تابع ima_appraise_signature تابعی که شناسه مربوط به آن داده شده است را با لیست قوانین موجود در ima مقایسه می‌کند و در صورتی که یکی از این قواعد، تابعی مشابه تابع داده شده

[1] Build
[2] Integrity Measurement Architecture

داشته و امضای دیجیتال آن معتبر بود، نتیجه را به صورت true باز می‌گرداند. [۷]

```
bool ima_appraise_signature(enum kernel_read_file_id id)
{ ...
    func = read_idmap[id] ?: FILE_CHECK;
    list_for_each_entry_rcu(entry, ima_rules, list) {
        if (entry->action != APPRAISE) continue;
        if (entry->func && entry->func != func) continue;
        if (entry->flags & IMA_DIGSIG_REQUIRED) found = true;
        break;
    }
return found; }
```

بخش عمده تغییرات مربوط به این قابلیت مربوط به پرونده lockdown.c می‌شود. مثلا پیاده‌سازی اصلی تابع نظیر شده به قلاب امنیتی locked_down با نام lockdown_is_locked_down در این پرونده قرار دارد. در این پرونده توابعی برای خواندن و نوشتن هم پیاده‌سازی شده است که در اصل یکسری بررسی‌ها روی وضعیت و حالات قفل‌سازی هسته انجام داده و در صورت نیاز پیام‌ها یا خطاهایی را که در هنگام خروجی‌دادن این توابع باید تولید بشوند، مشخص کرده و در نهایت عملکرد اصلی خواندن و نوشتن را انجام می‌دهند. [۷]

همچنین تابعی برای فعال‌سازی وضعیت قفل‌کردن هسته هم به نام lock_kernel_down وجود دارد که در آن کنترل می‌شود که سطح خواسته شده برای قفل کردن کمتر از سطح فعلی نباشد و در صورتی که مشکلی نبود، سطح قفل‌سازی هسته به سطح گفته شده ارتقا می‌یابد. قسمت‌های جالی از کد پیرامون نگاشت قلاب‌های امنیتی و چک صورت گرفته در هنگام قفل‌کردن کرنل برای سطح دسترسی مشخص که در این پرونده قرار دارند، در زیر آورده شده است:

```
static int lock_kernel_down(const char *where, enum lockdown_reason level)
 { ...
   if (kernel_locked_down >= level)
       return -EPERM;
 ... }

   static struct security_hook_list lockdown_hooks[] __lsm_ro_after_init={
       LSM_HOOK_INIT(locked_down, lockdown_is_locked_down),};
```

بدین ترتیب به شکلی نسبتا سطح بالا، نحوه پیاده‌سازی قابلیت قفل‌کردن هسته را در لینوکس بررسی کردیم. این قابلیت ریزه‌کاری‌های دیگری هم در قسمت‌های دیگر کد هسته دارد. تغییراتی جزئی در کدهای مربوط به ACPI [۱] که در هنگام بوت‌شدن سیستم برای مدیریت توان و شناسایی قطعات سخت‌افزاری کاربرد داده شده است. به علاوه تغییراتی جزئی در file.c و inode.c داده شده تا وضعیت قفل‌بودن هسته در حالاتی خاص گزارش بشود. از ذکر ریز این جزئیات به دلیل این که در اصل عملکرد نقش مهمی نداشتند، چشم‌پوشی شده است. [۷]

[1] Advanced Configuration and Power Interface

منابع

[1] ZDNet - Linux to get kernel ’lockdown’ feature
https://www.zdnet.com

[2] Kernel Newbies - Kernel v5.4
https://kernelnewbies.org/

[3] LWN: Linux Weekly News - v5.4 Kernel Lockdown
https://lwn.net/

[4] LWN: Linux Weekly News - 2012 Article about kernel lockdown
https://lwn.net/

[5] kernel_lockdown(7) — Linux manual page
https://man7.org/

[6] Kernel Commit Message
https://git.kernel.org/

[7] Linux Source Code - v5.4
https://github.com/torvalds/linux