CENTER FOR CYBER DEFENCE AND INFORMATION SECURITY

KTH VETENSKAP OCH KONST

SWEDISH ARMED FORCES

Swedish Defence University

Swedish Defence Research Agency

National Defence Radio Establishment

MSB Swedish Civil Contingencies Agency

checkm8


checkm8
checkm8
checkm8
checkm8
checkm8
checkm8
checkm8
checkm8
checkm8
checkm8
checkm8

Use-after-free

A UAF has three conditions,
1. a free of a pointer (without nullification),
2. a use of the pointer, and
3. a specific temporal order—the use is after the free.

```
char *c = (char *)malloc(32);          c =
strcpy(c, "foo");                      c = "foo"
free(c);                               c = ???
char *e = (char *)malloc(32);          c =
strcpy(e, "bar");                      c = "bar"
printf("%s\n", c);
return 0;
```

The Linux kernel is critical infrastructure

96.3% of the world's top one million servers run on Linux
95% of all cloud infrastructure operates on it
70% of all mobile devices are Android, running on top of the Linux kernel
71.8% of IoT developers choose Linux as their preferred operating system

The Linux Kernel Project is large

Linus Torvalds, 1991

Up until 2021

1,060,172 commits
24,300 different authors
32.2 million lines of code
74.3k files

April 17, 2022 – May 17, 2022

Excluding merges, **375 authors** have pushed **706 commits** to master and **706 commits** to all branches. On master, **1,680 files** have changed and there have been **16,155 additions** and **10,582 deletions**.

Kernel.org Bugzilla – Bug List

Home | New | Browse | Search | Search [?] | Reports | Help | New Account | Log In | Forgot Password

Fri May 20 2022 08:49:55 UTC

This list is too long for Bugzilla's little mind; the Next/Prev/First/Last buttons won't appear on individual bugs.

Hide Search Description

Tree: Mainline **Status:** NEW, ASSIGNED, REOPENED

9564 bugs found.

9564 bugs found.

ID	Product	Comp	Assignee ▲	Status ▲	Resolution	Summary	Changed ▼
215938	Power Ma	cpufreq	linux-pm	NEW	---	amd-pstate ignoring scaling_max_freq after waking from suspend	08:37:19
216005	Drivers	Input De	drivers_input-devices	NEW	---	"psmouse serio1: TouchPad at isa0060/serio1/input0 lost sync at byte 1" spam with kernel 5.17.7	07:53:58
216004	Drivers	Video(Ot	drivers_video-other	NEW	---	X server restarts when detaching eGPU, even when not used	07:34:57
216000	Drivers	PCI	drivers_pci	NEW	---	TBT storage hotplug fail when connect via thunderbolt dock	06:59:59
215867	Platform	x86-64	platform_x86_64	NEW	---	tboot suspend broken	06:10:14
215975	File Sys	NFS	trondmy	NEW	---	NFSD stops serving clients	05:26:46
216002	Virtuali	kvm	virtualization_kvm	NEW	---	When a break point is set, nested virtualization sees "kvm_queue_exception: Assertion `!env->exception_has_payload' failed."	02:41:27
216003	Virtuali	kvm	virtualization_kvm	NEW	---	Single stepping Windows 7 bootloader results in Assertion `ret < cpu->num_ases && ret >= 0' failed.	00:55:53
215949	Drivers	Network	drivers_network	NEW	---	Resume from suspend regression with aquantia atlantic driver	22:39:50
215958	Drivers	PCI	drivers_pci	NEW	---	thunderbolt3 egpu cannot disconnect cleanly	21:03:54
215988	Drivers	Sound(AL	perex	NEW	---	0414:a00d No input for Mic/Line-In	20:46:46
215934	Drivers	Sound(AL	perex	NEW	---	Behringer UMC 404 HD : clock source 41 is not valid and audio stuttering (confirmed for bcdDevice = 1.35)	18:18:13
216001	IO/Stora	Other	io_other	NEW	---	Samsung BAR Plus 256 GB flash drive significantly slower than expected	16:48:47
215079	Drivers	Sound(AL	perex	NEW	---	No more sound after kernel update	16:07:41
215886	Drivers	Network	drivers_network	NEW	---	dpaa2: TSO offload on lx2160a causes fatal exception in interrupt	16:01:19
208455	Drivers	Input De	drivers_input-devices	NEW	---	Dell XPS 15 9500 - psmouse serio1: elantech: elantech_send_cmd query 0x02 failed.	15:33:43

On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits

Qiushi Wu and Kangjie Lu
University of Minnesota
{wu000273, kjlu}@umn.edu

***Abstract*—Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption, while its openness and flexibility enable quicker innovation. More importantly, the OSS development approach is believed to produce more reliable and higher-quality software since it typically has thousands of independent programmers testing and fixing bugs of the software collaboratively.**

In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of stealthily introducing vulnerabilities in OSS via hypocrite commits (i.e., seemingly beneficial commits that in fact introduce other critical issues). The introduced vulnerabilities are critical because they may be stealthily exploited to impact massive devices. We first identify three fundamental reasons that allow hypocrite commits. (1) OSS is open by nature, so anyone from anywhere, including malicious ones, can submit patches. (2) Due to the overwhelming

Its openness also encourages contributors; OSS typically has thousands of independent programmers testing and fixing bugs of the software. Such an open and collaborative development not only allows higher flexibility, transparency, and quicker evolution, but is also believed to provide higher reliability and security [21].

A prominent example of OSS is the Linux kernel, which is one of the largest open-source projects—more than 28 million lines of code used by billions of devices. The Linux kernel involves more than 22K contributors. Any person or company can contribute to its development, e.g., submitting a patch through git commits. To make a change of the Linux kernel, one can email the patch file (containing `git diff` information) to the Linux community. Each module is assigned with a few maintainers (the list can be obtained through the script `get_maintainer.pl`). The maintainers then manually or employ

Minor memory leak needs fixing

```
1        pointerA = pointerC = malloc(...);
2        ...
3        pointerB = malloc(...);
4  ★     if (!pointerB) {
5  ★         return -ENOMEM;
6  ★     }
```

On the Feasibility of Stealthily Introduci Vulnerabilities in Open-Source Software Hypocrite Commits

Qiushi Wu and Kangjie Lu
University of Minnesota
{wu000273, kjlu}@umn.edu

***Abstract*—Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption, while its openness and flexibility enable quicker innovation. More importantly, the OSS development approach is believed to produce more reliable and higher-quality software since it typically has thousands of independent programmers testing and fixing bugs of the software collaboratively.**

In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of stealthily introducing vulnerabilities in OSS via hypocrite commits (i.e., seemingly beneficial commits that in fact introduce other critical issues). The introduced vulnerabilities are critical because they may be stealthily exploited to impact massive devices. We first identify three fundamental reasons that allow hypocrite commits. (1) OSS is open by nature, so anyone from anywhere, including malicious ones, can submit patches. (2) Due to the overwhelming patches and performance issues, it is impractical for maintainers to accept preventive patches for "immature vulnerabilities". (3) OSS like the Linux kernel is extremely complex, so the patch-review process often misses introduced vulnerabilities that involve complicated semantics and contexts. We then systematically study hypocrite commits, including identifying immature vulnerabilities and potential vulnerability-introducing minor patches. We also identify multiple factors that can increase the stealthiness of hypocrite commits and render the patch-review process less effective. As proof of concept, we take the Linux kernel as target OSS and safely demonstrate that it is practical for a malicious committer to introduce use-after-free bugs. Furthermore, we systematically measure and characterize the capabilities and opportunities of a malicious committer. At last, to improve the security of OSS, we propose mitigations against hypocrite commits, such as updating the code of conduct for OSS and developing tools for patch testing and verification.

I. Introduction

Open source software (OSS) shares its source code publicly, and allows users to use, modify, and even distribute under an open-sourcing licence. Since the forming of the Open Source Initiative in 1998, OSS has thrived and become quite popular. For example, as of August 2020, GitHub was reported to have over 40 million users and more than 37.6 million public repositories [19] (increased by 10 million from June 2018 [18]). It was also reported that everyone uses OSS [50] while 78%

Its openness also encourages contributors; O thousands of independent programmers testing of the software. Such an open and collaborat not only allows higher flexibility, transparen evolution, but is also believed to provide high security [21].

A prominent example of OSS is the Linux one of the largest open-source projects—more lines of code used by billions of devices. T involves more than 22K contributors. Any pe can contribute to its development, e.g., sub through git commits. To make a change of t one can email the patch file (containing `git d` to the Linux community. Each module is few maintainers (the list can be obtained th `get_maintainer.pl`). The maintainers then ma tools to check the patch and apply it if it is dee popular OSS, such as FreeBSD, Firefox, and adopts a similar patching process.

Because of the wide adoption, OSS like t and OpenSSL has become attractive targets attacks [9, 15]. While adversaries are incent always easy to find an exploitable vulnerabili is often extensively tested by developers an static and dynamic ways [63]. Even a bug wa not manifest the exploitability and impacts as wish. Thus, finding ideal exploitable vulnera not only advanced analyses and significant e bit of luck.

In this paper, we instead investigate the in from a critical perspective—the feasibility committer stealthily introducing vulnerabilit after-free (UAF) in OSS through hypocrite com beneficial minor commits that actually introdu issues). Such introduced vulnerabilities can be can exist in the OSS for a long period and be malicious committer to impact a massive numb users. Specifically, we conduct a set of studies understand and characterize hypocrite comm our suggestions for mitigation.

Fix leak, introduce Use-After-Free

```
1        pointerA = pointerC = malloc(...);
2        ...
3        pointerB = malloc(...);
4  ★     if (!pointerB) {
5  +         kfree(pointerA);
6  ★         return -ENOMEM;
7  ★     }
```

On the Feasibility of Stealthily Introduci Vulnerabilities in Open-Source Software Hypocrite Commits

Qiushi Wu and Kangjie Lu
University of Minnesota
{wu000273, kjlu}@umn.edu

***Abstract*—Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption, while its openness and flexibility enable quicker innovation. More importantly, the OSS development approach is believed to produce more reliable and higher-quality software since it typically has thousands of independent programmers testing and fixing bugs of the software collaboratively.**

In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of stealthily introducing vulnerabilities in OSS via hypocrite commits (i.e., seemingly beneficial commits that in fact introduce other critical issues). The introduced vulnerabilities are critical because they may be stealthily exploited to impact massive devices. We first identify three fundamental reasons that allow hypocrite commits. (1) OSS is open by nature, so anyone from anywhere, including malicious ones, can submit patches. (2) Due to the overwhelming patches and performance issues, it is impractical for maintainers to accept preventive patches for "immature vulnerabilities". (3) OSS like the Linux kernel is extremely complex, so the patch-review process often misses introduced vulnerabilities that involve complicated semantics and contexts. We then systematically study hypocrite commits, including identifying immature vulnerabilities and potential vulnerability-introducing minor patches. We also identify multiple factors that can increase the stealthiness of hypocrite commits and render the patch-review process less effective. As proof of concept, we take the Linux kernel as target OSS and safely demonstrate that it is practical for a malicious committer to introduce use-after-free bugs. Furthermore, we systematically measure and characterize the capabilities and opportunities of a malicious committer. At last, to improve the security of OSS, we propose mitigations against hypocrite commits, such as updating the code of conduct for OSS and developing tools for patch testing and verification.

I. Introduction

Open source software (OSS) shares its source code publicly, and allows users to use, modify, and even distribute under an open-sourcing licence. Since the forming of the Open Source Initiative in 1998, OSS has thrived and become quite popular. For example, as of August 2020, GitHub was reported to have over 40 million users and more than 37.6 million public repositories [19] (increased by 10 million from June 2018 [18]). It was also reported that everyone uses OSS [50] while 78%

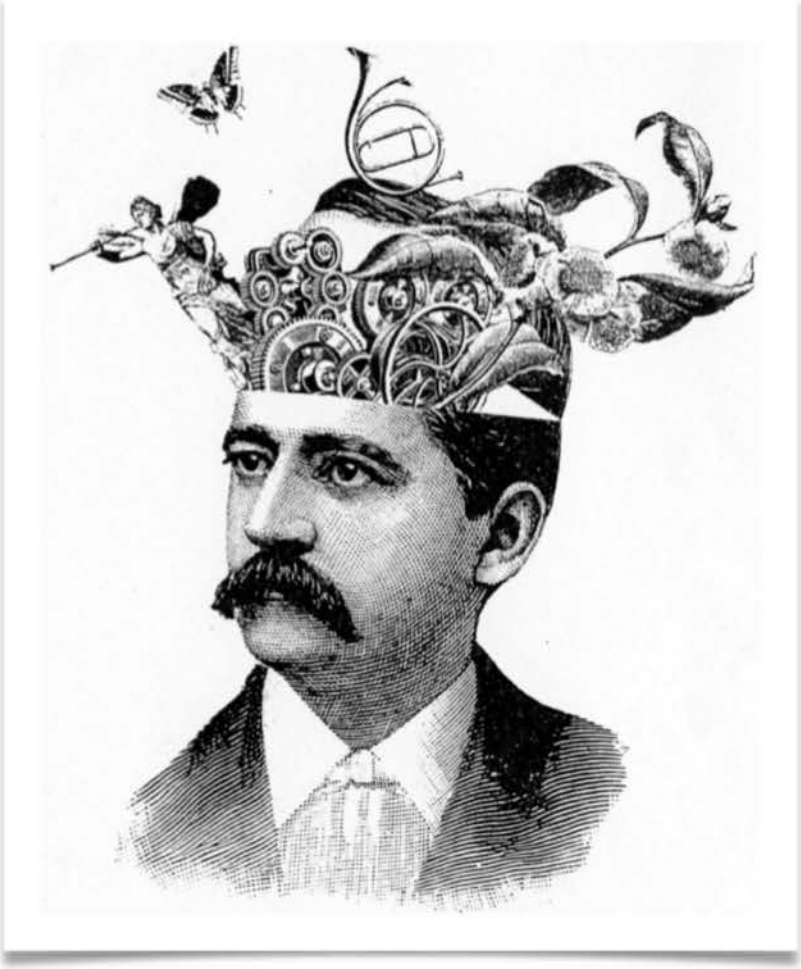
Its openness also encourages contributors; O thousands of independent programmers testing of the software. Such an open and collaborat not only allows higher flexibility, transparen evolution, but is also believed to provide high security [21].

A prominent example of OSS is the Linux one of the largest open-source projects—more lines of code used by billions of devices. T involves more than 22K contributors. Any pe can contribute to its development, e.g., sub through git commits. To make a change of t one can email the patch file (containing `git d` to the Linux community. Each module is few maintainers (the list can be obtained th `get_maintainer.pl`). The maintainers then ma tools to check the patch and apply it if it is dee popular OSS, such as FreeBSD, Firefox, and adopts a similar patching process.

Because of the wide adoption, OSS like t and OpenSSL has become attractive targets attacks [9, 15]. While adversaries are incent always easy to find an exploitable vulnerabili is often extensively tested by developers an static and dynamic ways [63]. Even a bug wa not manifest the exploitability and impacts as wish. Thus, finding ideal exploitable vulnera not only advanced analyses and significant e bit of luck.

In this paper, we instead investigate the in from a critical perspective—the feasibility committer stealthily introducing vulnerabilit after-free (UAF) in OSS through hypocrite com beneficial minor commits that actually introdu issues). Such introduced vulnerabilities can be can exist in the OSS for a long period and be malicious committer to impact a massive numb users. Specifically, we conduct a set of studies understand and characterize hypocrite comm our suggestions for mitigation.

The Linux Kernel Maintainers



Cognitive complexity

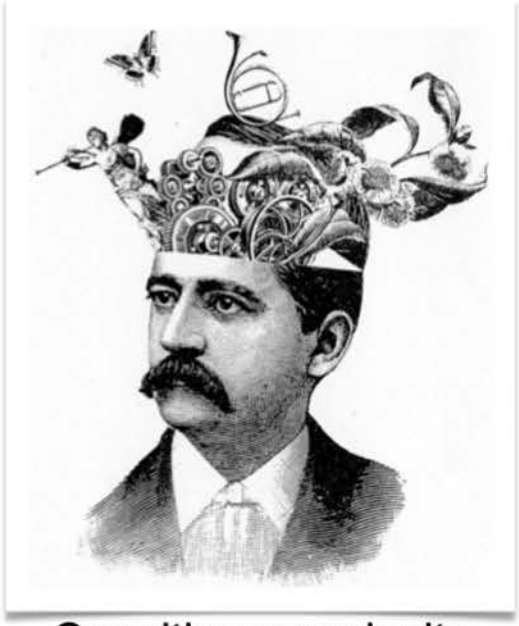
Conditions	Catch rate(%)
Concurrent issue	19.4%
Implicit release	36.3%
UAF in error-paths	42.0%
Alias	38.4 %
Indirect call	5/9
Baseline	56.6%

"In total, we collected 138 CVE-assigned vulnerabilities of different types, which are introduced by minor patches."

The Linux kernel
is critical infrastructure

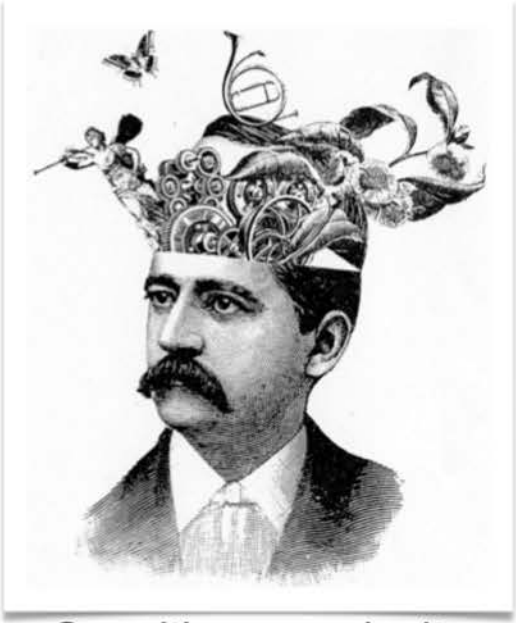
96.3% of the world's top one million servers run on Linux
95% of all cloud infrastructure operates on it
70% of all mobile devices are Android, running on top of the Linux kernel
71.8% of IoT developers choose Linux as their preferred operating system

Root cause: Cognitive complexity



Cognitive complexity

Developers need cognitive assistance

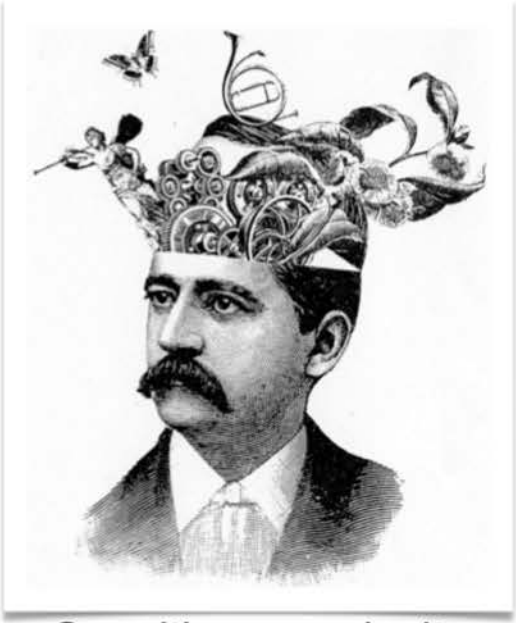


Cognitive complexity

Better tools for secure software development

- Advanced static-analysis techniques
- High-coverage, directed dynamic testing
- Memory-safe programming languages
- Formal verification
- …

Root cause: Cognitive complexity



Cognitive complexity

Better tools for secure software development

Research and innovation

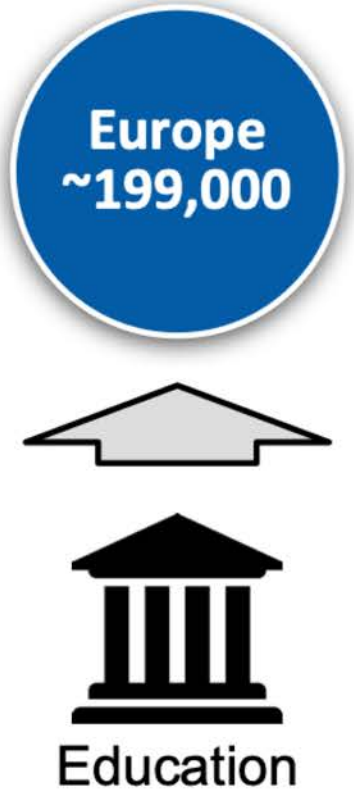
Insecure systems require many defenders

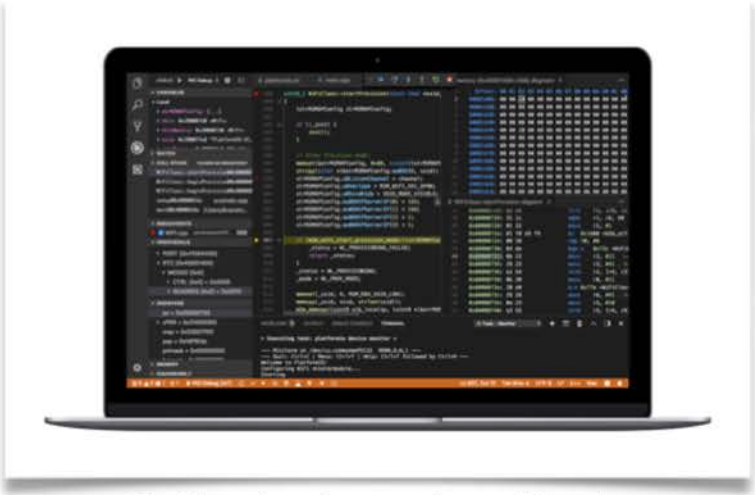
The Cybersecurity Workforce Gap 2021

Europe ~199,000

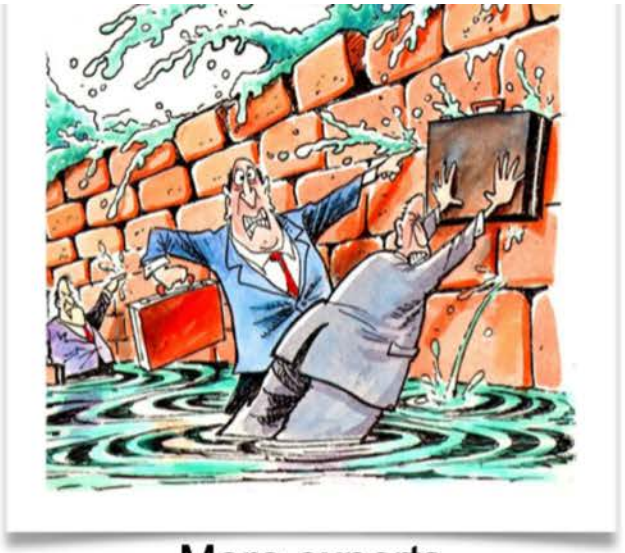
Insecure systems require many defenders

The Cybersecurity Workforce Gap 2021





Better tools and methods



More experts

Research
Innovation

Education

CENTER FOR CYBER DEFENCE
AND INFORMATION SECURITY