Open Source Technology Improvement Fund (OSTIF)

Linux Kernel Security Vulnerability Reporting and Remediation Practices Review

Prepared for Open Source Technology Improvement Fund
October 17, 2020

Table of Contents

Engagement Overview

Introduction

The Open Source Technology Improvement Fund ("OSTIF")[1], as requested by the Linux Foundation ("LF")[2], sought funding to engage Atredis Partners ("Atredis")[3] to evaluate the Linux kernel development community's practices for vulnerability reporting and remediation and to seek potential areas of improvement. The engagement was designed to include how security vulnerabilities are reported, prioritized, mitigated, and ultimately incorporated into the Linux kernel.

OSTIF asked Atredis to review the following key aspects of the Linux kernel development community's vulnerability reporting and remediation practices while keeping in mind the unique challenges of free and open source software[4] development:

- Determine how the Linux kernel development community's practices differ from industry best practices and whether the differences are appropriate
- Evaluate existing communication channels and determine how communication channels could be improved
- Analyze the impacts of public security remediation activities, including patching and reporting

Engagement Approach and Methodology

The project began with a kickoff meeting with the OSTIF and the LF's designated key stakeholders on July 2, 2020. The information gathering and analysis activities concluded on August 21, 2020. During this time, Atredis Risk and Advisory consultants reviewed all provided resources of information, interviewed key stakeholders, and conducted independent research and analysis prior to developing this report. The draft report was issued to this project's key stakeholders for review and feedback prior to publishing the report publicly.

To accomplish the objectives of the engagement and to support clear observations and recommendations, Atredis segmented the activities for review into three primary categories representing the processes related to Linux kernel security vulnerability reporting and remediation practices.

These categories are referenced throughout the report, and are as follows:

[1] https://ostif.org/
[2] https://www.linuxfoundation.org/
[3] https://www.atredis.com/
[4] https://en.wikipedia.org/wiki/Free_and_open-source_software

1. **Security Vulnerability Disclosure** – The process for reporting security vulnerabilities to the Linux kernel development community for analysis and other related disclosures.
2. **Security Vulnerability Evaluation and Remediation** – The process of evaluating and remediating reported security vulnerabilities within the development and update process.
3. **Security Vulnerability Tracking** – The process for tracking Linux kernel security vulnerabilities.

Project Participants

The tables below outline the individuals who participated in the assessment process. Only the Atredis Risk and Advisory Practice conducted research and analysis for the engagement, while the other participants were interviewed based on their relevant experience, either with the Linux kernel development processes, or related disciplines.

In addition to researching publicly available information related to the security vulnerability reporting and remediation practices of the Linux kernel development community, OSTIF also directed Atredis to interview specific individuals, who have experience with the Linux kernel development community and security vulnerability reporting and remediation processes. Interviews were conducted via video conference and were approximately 30 – 60 minutes in duration.

NAME	ORGANIZATION	ROLE
Kiston Finney	Atredis Partners Risk and Advisory	Director of Risk and Advisory / Practice Oversight and Report Contributor
Bill Carver	Atredis Partners Risk and Advisory	Managing Principal Consultant Risk and Advisory Lead Assessor and Report Author
Michael Robbins	Atredis Partners Risk and Advisory	Principal Risk and Advisory Consultant Assessor and Report Contributor
Greg Nance	Atredis Partners Risk and Advisory	Principal Risk and Advisory Consultant Assessor and Report Contributor
Shawn Moyer	Atredis Partners	CEO and Cofounder
Nathan Keltner	Atredis Partners	CTO and Cofounder
Josh Thomas	Atredis Partners	COO and Cofounder
HD Moore	Atredis Partners	VP Research and Development
Zach Lanier	Atredis Partners	Managing Principal, Embedded:IoT Practice Lead
Charles Holmes	Atredis Partners	Research Consulting Director

NAME	ORGANIZATION	ROLE
Chris Bellows	Atredis Partners	Research Consulting Director
Darren Kemp	Atredis Partners	Managing Principal Research Consultant
Mike Kershaw	Atredis Partners	Principal Research Consultant

NAME	ORGANIZATION ROLE	TOPICS DISCUSSED	DATE(S)
Greg Kroah-Hartman	The Linux Foundation Lead Maintainer and Fellow	• Experience with the Linux kernel development community • Linux kernel security vulnerability reporting and remediation practices • What works well and areas for improvement	7/24/2020 8/11/2020
Kees Cook	Google Linux Kernel Security Engineer	• Experience with the Linux kernel development community • Linux kernel security vulnerability reporting and remediation practices • What works well and areas for improvement	8/19/2020 8/20/2020
Roy Yang	Google Technical Lead Container Optimization Services	• Experience with the Linux kernel development community • Linux kernel security vulnerability reporting and remediation practices • What works well and areas for improvement	8/3/2020
Garth Mollett	Red Hat Product Security Technical Lead	• Experience with the Linux kernel development community • Linux kernel security vulnerability reporting and remediation practices • What works well and areas for improvement	8/4/2020 8/11/2020
Wade Mealing	Red Hat Product Security PSIRT - Kernel	• Experience with the Linux kernel development community • Linux kernel security vulnerability reporting and remediation practices • What works well and areas for improvement	8/4/2020 8/11/2020

Engagement Context

Open Source Technology

Open source technology has seen a dramatic increase in use in recent years primarily due to its availability and relatively low cost to implement. The rapid adoption by corporations of open source software to support a wide variety of products has led to more contributions to the source code by individuals employed by these corporations. The opportunities for developers to gain a sense of empowerment and ownership within an open source project continue to lead to an increase in the overall number of contributors to open source code. Similar to what other software development environments face, challenges exist within the open source technology space.

Security Vulnerability Disclosure and Tracking

Mature and well-established software companies and projects implement processes to facilitate security vulnerability disclosure and tracking. These processes provide an important mechanism for researchers to identify and responsibly disclose potential security vulnerabilities as they are discovered. These software providers have publicly available vulnerability disclosure policies to help facilitate the process, thereby allowing impacted parties time to develop remediation strategies before the vulnerability information is publicly shared and then potentially used by attackers who had not already discovered the flaws on their own.

Within the Information Security community there is a general respect and appreciation for standardized vulnerability disclosure practices. Researchers identify vulnerabilities and report them directly to the relevant vendor, giving them the opportunity to remediate the vulnerability in a timely manner. In return, vendors remediate the issue and, in some cases, give credit or financial bounties to the reporter. This incentivizes vendors and researchers to work together to ultimately provide better security for products without needless delay. This creates the best possible outcomes of providing better security for products without needless delay, while also ensuring the appropriate individuals are given credit for their work.

Closely aligned with vulnerability disclosure is the tracking and reporting of Common Vulnerabilities and Exposures (CVE)[5]. The original CVE concept was launched in 1999 by the MITRE Corporation, primarily as a mechanism to achieve greater interoperability for the management of vulnerability data across different sources[6]. Over time, the CVE concept has been widely adopted by the Information Security community and major software vendors and is used within many security tools.

The Linux Kernel[7]

The Linux kernel has become one of the most significant free and open source software projects in existence. What began as an idea to create a free operating system has continued to grow exponentially over the years and is now relied upon by:

- Mobile phones
- Consumer electronics
- Financial markets
- Supercomputers
- Public web servers
- Satellites
- The International Space Station
- IoT devices

Standards for development have consistently matured and the diversity of the contributors has facilitated innovation. Downstream distributions are able to customize the functionality of the kernel and the multitude of contributors work to identify software bugs, implement fixes, and perform testing to keep the software functioning smoothly and securely.

[5] https://cve.mitre.org/
[6] https://cve.mitre.org/about/history.html
[7] https://www.linuxfoundation.org/wp-content/uploads/2020/08/2020_kernel_history_report_082720.pdf

The Linux kernel, as with all other software development environments, must be updated regularly in order to provide new features and implement bug fixes. The potential risks that security vulnerabilities pose to the Linux kernel are largely addressed by the quick turnaround of patching software bugs, usually within one to two weeks. However, due to the large number of downstream distributions and the multiple versions of the kernel in use, how quickly the fixes are implemented (downstream) varies. There are cases where downstream distributions implement the latest kernel releases within hours or days. In other cases, the latest kernel updates are implemented within weeks, months, or years. There are scenarios where downstream distributions choose not to implement the latest kernel releases at all. Ultimately, the decision to implement current kernel versions and in what timeframe is made by the downstream distributions.

The Linux kernel development community consists of thousands of developers who are not directly employed by the LF. Primarily, kernel development contributors are employed by invested downstream distributions and maintainers of products and devices leveraging the Linux kernel. Contributors may also be paid consultants, or people contributing to the project on their own time. Within the development community there are assigned subsystem maintainers who are responsible for specific areas of functionality within the Linux kernel. The subsystem maintainers help to ensure that valid, tested, and functional code is used to update their areas of responsibility within the Linux kernel, which will ultimately be merged into the mainline kernel releases.

A management reporting structure directing the developers and subsystem maintainers to prioritize their development activities does not exist within the Linux kernel development community. While appropriate for this specific community, this lack of managed reporting structure creates a unique set of challenges that Atredis considered when contemplating process recommendations.

Linux Kernel Security Vulnerability Practices

Security Vulnerability Disclosure

Almost all bugs are submitted to the Linux kernel project through one of the numerous (approximately 200) mailing lists found in the source tree[8] and on the kernel.org website[9]. The `security@kernel.org` email alias was created in January of 2005. The catalyst for the email alias creation was a discussion thread asking to clear up confusion on the "…proper procedure for reporting possible kernel security issues" due to conflicting instructions and a lack of a formalized procedure[10]. The thread included discussions regarding a centralized handling of security vulnerability reporting similar to that of traditional software companies. The intent of the alias was to facilitate submissions of potential security vulnerabilities identified within the Linux kernel.

The email alias and supporting procedure was formalized with a commit on March 3, 2005[11]. This commit stated that the Linux kernel developers take security very seriously, and addressed the purpose of the alias, how and when to contact the alias, disclosure processes and timelines, and a statement on non-disclosure agreement expectations. It is up to the submitter to follow the prescriptive guidelines posted on the website to identify how and where to submit the request.

As the Linux kernel development community matured and formalized more processes and development methods, it is of interest to note that the first stable kernel release also happened in 2005, as well as the 2.6.12 kernel release, which was the first release to use the newly created Git[12].

There is another email alias, `hardware-security@kernel.org`[13], specifically designed for purposes of handling hardware security issues. The members of this alias manage the hardware issues and related embargo processes which ultimately may have an impact on multiple operating systems, not just the Linux kernel and Linux distributions. Evaluation of this list and its associated processes was not in scope for this assessment. More information about the use of this alias and subsequent processes can be found on the `https://www.kernel.org` website.

---

[8] https://www.kernel.org/doc/html/latest/process/maintainers.html
[9] http://vger.kernel.org/vger-lists.html
[10] https://lore.kernel.org/lkml/41E2B181.3060009@rueb.com/
[11] https://git.kernel.org/pub/scm/linux/kernel/git/history/history.git/commit/?id=38d6c5c5a1cf08d5948726071ef7c7781ffe4f7c
[12] https://www.linuxfoundation.org/wp-content/uploads/2020/08/2020_kernel_history_report_082720.pdf
[13] https://www.kernel.org/doc/html/latest/process/embargoed-hardware-issues.html

Also of note are automated kernel fuzzing tools (e.g. syzkaller, Trinity) used to identify large numbers of potential bugs. The potential bugs identified by these tools are identified without real-world kernel usage context, are often difficult to replicate, and need more qualitative analysis before determining impacts and any mitigation strategies. The bugs identified by the tools should be addressed differently given the volume and analysis required. While not directly in scope for this project, Atredis sees value in reviewing potential security vulnerabilities reported by these automated tools regularly for relevance as they may have value in providing a more secure Linux kernel.

Security Vulnerability Evaluation and Remediation

During the assessment, Atredis learned from developers and other key stakeholders that the distinction between a generalized software bug and a security relevant software bug is virtually non-existent within the Linux kernel development community. A "bug is a bug"[14] is an approach that has been observed and emphasized to Atredis throughout this engagement. Very few submitters self-identify a bug as a security vulnerability upfront, resulting in the `security@kernel.org` alias rarely being utilized. Additionally, there are some instances where the bug fix submitted to the `security@kernel.org` alias is evaluated and ultimately considered to not have a security impact to the Linux kernel.

There is currently no documented prioritization methodology within the Linux kernel development community for reported security vulnerabilities. Rather, the focus is on addressing all bug fixes, security and otherwise, in an expedient manner. Once identified, fixes are usually applied to the mainline Linux kernel within a few weeks, with some outliers potentially requiring more time based on the potential impact a patch might have. To better understand how frequently changes are applied, the latest Linux kernel development report cites that for the 5.8 version of the kernel, there were 10.7 commits per hour[15].

Security Vulnerability Tracking

As it relates to the tracking of security vulnerabilities, there is currently no formal process within the Linux kernel development community to specifically tag and/or track changes to the kernel that are directly related to an identified security vulnerability. The security impact(s) of a kernel fix may not be known for months or even years after being addressed due to the time and effort needed to research exploitability. Additionally, downstream use of the kernel in distributions could play a part in determining potential security impacts of a kernel patch.

---

[14] http://kroah.com/log/blog/2018/02/05/linux-kernel-release-model/
[15] https://www.linuxfoundation.org/wp-content/uploads/2020/08/2020_kernel_history_report_082720.pdf

Currently there are multiple Linux kernel distributions which are CVE Numbering Authorities (CNA) that subsequently manage CVE entries which affect their products. Neither the Linux kernel project nor the Linux Foundation are currently registered as a CNA[16].

CVEs are designed to allow vulnerability databases and other capabilities to be linked together and to facilitate the comparison of security tools and services. A CNA is an organization that distributes CVE IDs to researchers and information technology vendors for inclusion in first-time public announcements of new vulnerabilities, without directly involving the CVE Team in the details of the specific vulnerabilities[17].

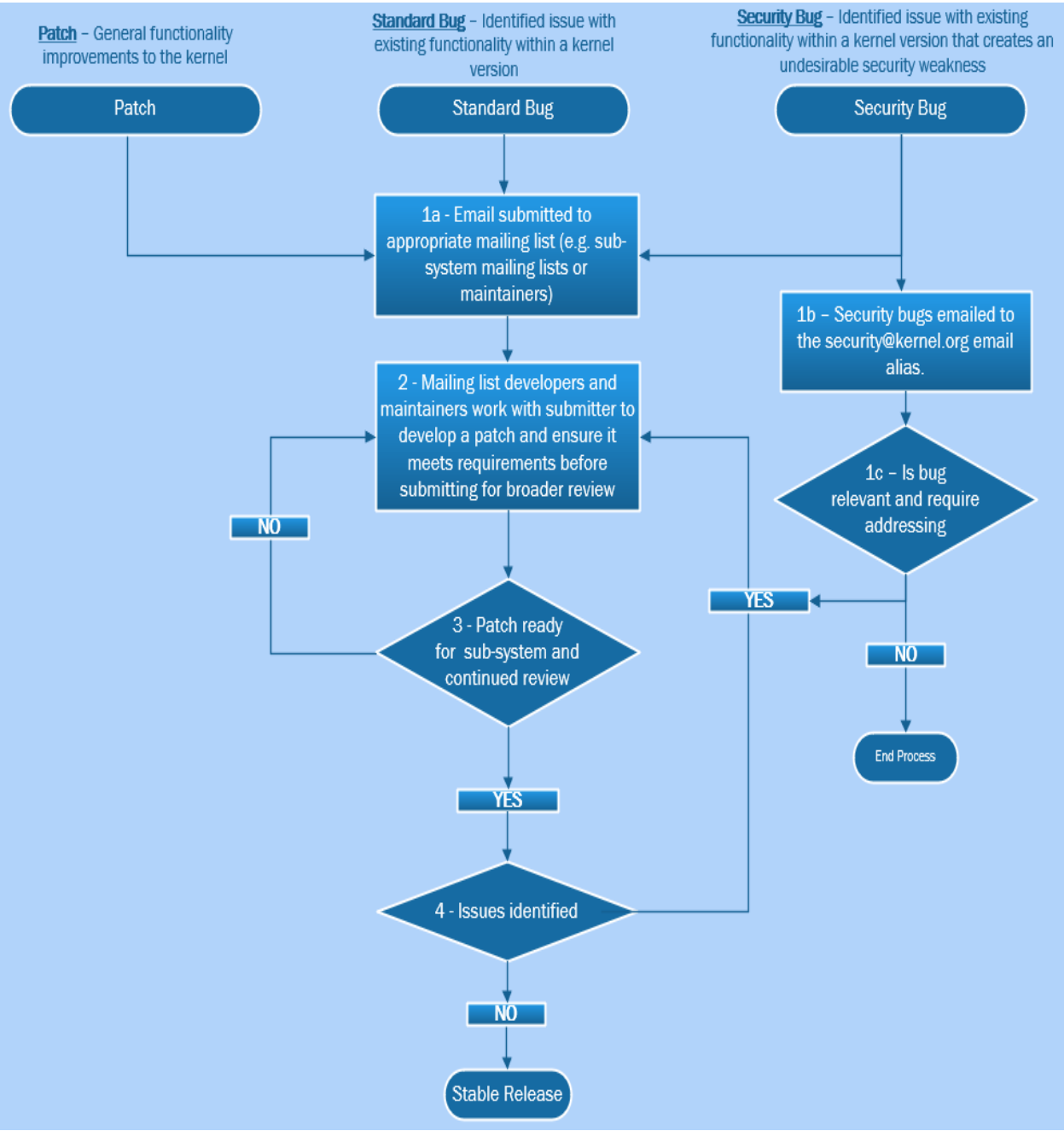
Although two separate programs, CVEs feed into the National Vulnerability Database (NVD). The CVE List feeds NVD, which then builds upon the information included in CVE Entries to provide enhanced information for each entry such as fix information, severity scores, and impact ratings[18].

[16] https://cve.mitre.org/cve/request_id.html#cna_participants
[17] https://cve.mitre.org/about/faqs.html
[18] https://cve.mitre.org/about/cve_and_nvd_relationship.html

Disclosure and Remediation Process Workflow Overview

Figure 1 developed by Atredis below provides a visual representation of the bug and bug fix submission process for the Linux kernel.

Figure 1 - Disclosure and Remediation Process Workflow



The representation in Figure 1 is inclusive of all bug fixes (security related or not) and outlines the following high-level process steps:

- **1a** – After identifying a bug fix, a submitter sends an email to the appropriate mailing list based on the scope of what is affected. The mailing lists and maintainers can be found in multiple locations including: `http://vger.kernel.org/`
- **1b** – After identifying a security bug, a submitter sends an email to the `security@kernel.org` email alias. This alias consists of approximately twelve individuals, including some of the primary kernel maintainers. Sending an email to this private alias keeps the information about the bug fix out of public mailing lists while the bug is analyzed. This allows for some time and discussion between those maintainers on the `security@kernel.org` list before a fix is implemented and communicated.
- **1c** – In some cases, the submissions to the `security@kernel.org` email alias are not immediately identified as being security vulnerabilities and are instead identified as general functionality bugs. In either case, submissions to this list are reviewed and then, if applicable, follow the standard process for a bug fix.
- **2** – Once a bug fix has been submitted, the subsystem maintainers, developers, and submitters work together in the open mailing list forum(s) to finalize a fix for the associated bug and or functionality. Within the mailing list, the bug fix is reviewed and evaluated prior to submission to the subsystem maintainer and testers for review.
- **3** – Once evaluated, tested, and approved, the patch is incorporated into the development kernel.
- **4** – If there are no issues identified, the patch is then incorporated into the applicable stable release.

Conclusions

Linux is the world's largest and most pervasive open source software project in the history of computing [19], with a passionate and engaged development community that has been supporting the Linux kernel for almost thirty years. Atredis acknowledges the near impossibility of understanding all perspectives related to security vulnerability reporting practices within the limited scope and timeframe allotted for this engagement. Throughout the discovery and data gathering sessions, Atredis considered several possible recommendations. The two key recommendations included in this report are the culmination of objective evaluation of current practices and are intended to be a catalyst for additional discussions regarding security vulnerability reporting within the Linux kernel development community.

Vulnerabilities in the Linux kernel are often not known to be a security bug at the time a bug is reviewed and the fix is developed. In some cases, security vulnerabilities are not known for months or even years after the fact. Additionally, a security vulnerability may be directly related to how the Linux kernel or a specific patched component is being implemented and compiled by a downstream distribution. The Linux kernel by itself is just a kernel. It essentially does not become fully functional or usable in a practical sense until additional software and services build upon the kernel to create an operating system or runtime environment.

The consensus from the designated key stakeholders and the reviewed references indicate that the processes for evaluating and patching bugs within the Linux kernel development community may not be perfect, but they are effective. The stakeholders interviewed felt that the processes for resolving bugs in supported versions of the Linux kernel, security-relevant or otherwise, address these defects in a reasonable and satisfactory timeframe. While there is no formal prioritization of fixes within the Linux kernel development process, this approach has proven reasonable and appropriate given the general speed at which fixes have been implemented since the Linux kernel's inception in 1991 [20], relative to other open source software projects.

With that stated, a key remaining issue is that kernel patches may take months, years, or in some cases never make it into downstream distributions of the Linux kernel. An area of some contention within the Linux kernel security vulnerability reporting and remediation processes is related to the tracking and reporting of CVEs. Research conducted on this topic, specifically focused on evaluating how long it takes for downstream distributions of the Linux kernel to

[19] https://www.linuxfoundation.org/projects/linux/
[20] https://www.linux-magazine.com/Online/News/Linus-Torvalds-Confirms-the-Date-of-the-First-Linux-Release

be patched against known security vulnerabilities with a CVE designation, is extensive and ongoing[21].

Specifically, there is some debate as to whether the responsibility for reporting and tracking of CVEs should reside with the Linux kernel development community or the downstream distribution maintainers. Downstream distributions, which leverage the kernel for their products, in many cases have deviated from the Linux kernel with their own development to allow functionality that is needed to support their specific products and their end users' needs.

As a result of this downstream manipulation of the Linux kernel, distribution development teams are put in a challenging position of maintaining security within their products given the deviation from the known-good stable releases. If modifications have been made to stable releases of the Linux kernel, then the benefits of swift patching (including security vulnerabilities) conducted by the Linux kernel development community are minimized. The result is that distributions must essentially maintain their own versions of the modified Linux kernel on an on-going basis. Downstream distributions are then left to monitor Linux kernel releases and review the new code base for potential security implications for their specific products. Often, the downstream distributions select patches they have deemed to have a potential security impact, and then test and deploy what they conclude is necessary.

Recommendations

Keep Security Vulnerability Discussions Public Instead of Private

The reporting and analysis of potential security vulnerabilities should be conducted in public, like standard bug reporting and analysis. The concept of private security vulnerability disclosure is fundamentally counter to the concept of free and open source software development and use. Apart from embargo restrictions, the private nature of these activities is most likely unnecessary.

If private analysis of reported security vulnerabilities prevented exploits from being in the wild for long periods of time, Atredis would see some value in maintaining this approach. However, the combination of the speed at which most bugs are fixed within supported versions of the Linux kernel and the lengthy duration of time required for downstream distributions to remediate issues within their Linux kernel versions minimizes the perceived value of private discussions. Keeping potential security bug and bug fix discussions private instead of in public view provides an additional avenue for scrutiny.

[21] https://events19.linuxfoundation.org/wp-content/uploads/2017/11/Sub-system-Update-Kernel-Self-Protection-Project-Kees-Cook-Google.pdf

It is unwise to assume that bad actors have not already discovered a security vulnerability prior to its identification within the public Linux kernel development community. The assumption is that nearly all bugs within the Linux kernel have some potential for scenarios that could lead to exploitable security vulnerabilities.

CVE Reporting Should Reside with Downstream Distributions

The responsibility for managing CVE reporting and tracking should reside with the downstream distributions building their projects based on the Linux kernel. CVE reporting and tracking is widely adopted as a strategy for mitigating risk within downstream Linux distributions. Organizations or individuals who utilize these downstream Linux distributions may be heavily reliant upon CVE tracking and reporting to track and manage vulnerabilities and keep computing environments as secure as possible.

CNAs are required to provide critical details for CVE entries such as product specifications, vulnerable versions, potential exploit impacts, mitigation instructions, and other key information. This information is most effectively provided by downstream distributions because they have the most knowledge about their products. It is not feasible nor appropriate for the Linux kernel development community to have responsibility for managing CVEs or being a CNA. In recent discussions between one of the lead maintainers for the Linux kernel and representatives from MITRE it was also agreed that CVEs were not designed for and do not work for the Linux kernel[22].

Additionally, it should be noted that some downstream distributions are already moving toward focusing on how to quickly update their releases with long-term stable and stable kernel versions - without selectively choosing updates - in order to help reduce the risks and challenges posed by potential kernel security vulnerabilities. Moving toward this approach will help downstream distributions take greater advantage of the speed at which bugs are fixed within Linux kernel versions and help to improve the overall security of their products.

[22] https://www.youtube.com/watch?v=HeeoTE9jLjM

Appendix I: Research Resources

The Linux Kernel Documentation:
https://www.kernel.org/doc/html/latest/

The Linux Kernel Documentation – The Linux kernel user's and administrator's guide:
https://www.kernel.org/doc/html/latest/admin-guide/index.html

The Linux Kernel Documentation – The Linux kernel user's and administrator's guide – Hardware vulnerabilities:
https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/index.html

The Linux Kernel Documentation – The Linux kernel user's and administrator's guide – Reporting Bugs:
https://www.kernel.org/doc/html/latest/admin-guide/reporting-bugs.html

The Linux Kernel Documentation – The Linux kernel user's and administrator's guide – Security bugs:
https://www.kernel.org/doc/html/latest/admin-guide/security-bugs.html

The Linux Kernel Documentation – The Linux kernel user's and administrator's guide – Bug Hunting:
https://www.kernel.org/doc/html/latest/admin-guide/bug-hunting.html

The Linux Kernel Documentation – The Linux kernel user's and administrator's guide – Bisecting a bug:
https://www.kernel.org/doc/html/latest/admin-guide/bug-bisect.html

The Linux Kernel Documentation – Working with the kernel development community:
https://www.kernel.org/doc/html/latest/process/index.html

The Linux Kernel Documentation – Working with the kernel development community – Linux kernel licensing rules:
https://www.kernel.org/doc/html/latest/process/license-rules.html

The Linux Kernel Documentation – Working with the kernel development community – HOWTO do Linux kernel development:
https://www.kernel.org/doc/html/latest/process/howto.html

The Linux Kernel Documentation – Working with the kernel development community – Linux Kernel Contributor Covenant Code of Conduct Interpretation:
https://www.kernel.org/doc/html/latest/process/code-of-conduct-interpretation.html

The Linux Kernel Documentation – Working with the kernel development community – Linux Kernel Enforcement Statement:
https://www.kernel.org/doc/html/latest/process/kernel-enforcement-statement.html

The Linux Kernel Documentation – Working with the kernel development community – Submitting Drivers For The Linux Kernel:
https://www.kernel.org/doc/html/latest/process/submitting-drivers.html

The Linux Kernel Documentation – Working with the kernel development community – Everything you ever wanted to know about Linux -stable releases:
https://www.kernel.org/doc/html/latest/process/stable-kernel-rules.html

The Linux Kernel Documentation – Working with the kernel development community – Linux Kernel patch submission checklist:
https://www.kernel.org/doc/html/latest/process/submit-checklist.html

The Linux Kernel Documentation – Working with the kernel development community – Embargoed hardware issues:
https://www.kernel.org/doc/html/latest/process/embargoed-hardware-issues.html

The Linux Kernel Documentation – Working with the kernel development community – List of maintainers and how to submit kernel changes:
https://www.kernel.org/doc/html/latest/process/maintainers.html

The Linux Kernel Documentation – Working with the kernel development community – Applying Patches To The Linux Kernel:
https://www.kernel.org/doc/html/latest/process/applying-patches.html

Linux kernel mailing list:
http://vger.kernel.org/vger-lists.html

LKML Archive on lore.kernel.org – Date: Mon, 10 Jan 2005 10:46:57 -0600:
https://lore.kernel.org/lkml/41E2B181.3060009@rueb.com/

index : kernel/git/history/history.git – 2005-03-09 16:42:18 -0800:
https://git.kernel.org/pub/scm/linux/kernel/git/history/history.git/commit/?id=38d6c5c5a1cf08d5948726071ef7c7781ffe4f7c

Wikipedia – Linux kernel:
https://en.wikipedia.org/wiki/Linux_kernel

Wikipedia – Open source software:

https://en.wikipedia.org/wiki/Free_and_open-source_software

The Linux Foundation:
https://www.linuxfoundation.org/

The Linux Foundation – About:
https://www.linuxfoundation.org/about/

The Linux Foundation – Projects:
https://www.linuxfoundation.org/projects/linux/

The Linux Foundation – Training:
https://training.linuxfoundation.org/about/

The Linux Foundation – Board Members:
https://www.linuxfoundation.org/about/board-members/

The Linux Foundation – Leadership:
https://www.linuxfoundation.org/about/leadership/

The Linux Foundation – Linux Foundation Fellows:
https://www.linuxfoundation.org/about/linux-foundation-fellows/

The Linux Foundation – Technical Advisory Board:
https://www.linuxfoundation.org/about/technical-advisory-board/

The Linux Foundation – Newsroom – Press:
https://www.linuxfoundation.org/newsroom/press/

The Linux Foundation – Blog:
https://www.linuxfoundation.org/resources/blog/

The Linux Foundation – 2020 Linux Kernel History Report:
https://www.linuxfoundation.org/wp-content/uploads/2020/08/2020_kernel_history_report_082720.pdf

The Linux Foundation – Improving Trust and Security in Open Source Projects:
https://www.linuxfoundation.org/wp-content/uploads/2020/02/improving_trust_security_in_oss_projects.pdf

Wikipedia – Linux Foundation – Linux Foundation Projects:
https://en.wikipedia.org/wiki/Linux_Foundation#Linux_Foundation_Projects

Open Source Technology Improvement Fund:
https://ostif.org/

Open Source Technology Improvement Fund – About Staff:
https://ostif.org/about-the-ostif-staff/

Open Source Technology Improvement Fund – OSTIF Supported Projects:
https://ostif.org/ostif-supported-projects/

MITRE – Common Vulnerabilities and Exposures:
https://cve.mitre.org/

MITRE – Common Vulnerabilities and Exposures List Home:
https://cve.mitre.org/cve/

MITRE – Common Vulnerabilities and Exposures Submit a CVE Request Form:
https://cveform.mitre.org/

MITRE – Common Vulnerabilities and Exposures History:
https://cve.mitre.org/about/history.html

MITRE – Common Vulnerabilities and Exposures – CVE List Documents and Guidance:
https://cve.mitre.org/about/documents.html#cve_list

MITRE – Common Vulnerabilities and Exposures FAQs:
https://cve.mitre.org/about/faqs.html

MITRE – Common Vulnerabilities and Exposures About CVE Entries:
https://cve.mitre.org/cve/identifiers/index.html

MITRE – Common Vulnerabilities and Exposures CVE Reference Key / Maps:
https://cve.mitre.org/data/refs/index.html

MITRE – Common Vulnerabilities and Exposures CVE Numbering Authority (CNA) Rules:
https://cve.mitre.org/cve/cna/rules.html

MITRE – Common Vulnerabilities and Exposures CVE Numbering Authority (CNA) Rules – Section 7:
https://cve.mitre.org/cve/cna/rules.html#section_7_assignment_rules

MITRE – Common Vulnerabilities and Exposures CVE Numbering Authority (CNA) Rules – Section 8.1:
https://cve.mitre.org/cve/cna/rules.html#section_8-1_cve_entry_information_requirements

MITRE – Common Vulnerabilities and Exposures Researcher Reservation Guidelines:
https://cve.mitre.org/cve/researcher_reservation_guidelines

MITRE – Common Vulnerabilities and Exposures CVE Numbering Authority (CNA) Rules – Appendix C: Process to Correct Assignment Issues or Update CVE Entries:
https://cve.mitre.org/cve/cna/rules.html#appendix_c_process_to_correct_assignment_issues_update_cve_entries

MITRE – Common Vulnerabilities and Exposures CVE Numbering Authority (CNA) Rules – Submitting CVE Entry Information to CVE Team:
https://cve.mitre.org/cve/cna.html#submitting_cve_entry_info

MITRE – Common Vulnerabilities and Exposures – Submitting CVE Entries to Root CNAs:
https://cveproject.github.io/docs/cna/submitting_cve_entries_to_root_cnas/index.html

MITRE – Common Vulnerabilities and Exposures – CVE Program Overview:
https://cve.mitre.org/cve/cna/CVE_Program_Overview.pptx

MITRE – Common Vulnerabilities and Exposures – CVE Request Web Form Training:
https://cve.mitre.org/docs/docs-2016/CVE_Request_Web_Form_Overview.pdf

MITRE – Common Vulnerabilities and Exposures – CVE Board Charter:
https://cve.mitre.org/community/board/charter.html

MITRE – Common Vulnerabilities and Exposures – CVE Numbering Authorities:
https://cve.mitre.org/cve/cna.html

MITRE – Common Vulnerabilities and Exposures – CVE IDs and How to Get Them:
https://cve.mitre.org/CVEIDsAndHowToGetThem.pdf

MITRE – Common Vulnerabilities and Exposures – Towards a Common Enumeration of Vulnerabilities:
https://cve.mitre.org/docs/docs-2000/cerias.html

MITRE – Common Vulnerabilities and Exposures – Participating CNAs:
https://cve.mitre.org/cve/request_id.html#cna_participants

MITRE – Common Vulnerabilities and Exposures – Terminology:
https://cve.mitre.org/about/terminology.html

Youtube – CVE Program – CVE Program Overview:
https://youtu.be/lkWUJXNu0kQ

Youtube – CVE Program – Becoming a CNA:
https://youtu.be/OqRZg02AP-g

Youtube – CVE Program – CNA Process:
https://youtu.be/yLqUMKD2Y9k

Youtube – CVE Program – Assigning CVE IDs:
https://youtu.be/He3uUzk0Fzs

Youtube – CVE Program – CVE Entry Creation:
https://youtu.be/qfbh4YGaHjw

Youtube – CVE Program – CVE Entry Submission Process:
https://youtu.be/uiKNWnBbeFg

Wikipedia – Common Vulnerabilities and Exposures:
https://en.wikipedia.org/wiki/Common_Vulnerabilities_and_Exposures

Linux Kernel Summit 2019 – Reflections on kernel development process, quality and testing – Dmitry Vyukov:
https://linuxplumbersconf.org/event/4/contributions/554/attachments/353/584/Reflections__Kernel_Summit_2019.pdf

Youtube – BlueHat IL 2020 - Dmitry Vyukov - syzkaller: Adventures in Continuous Coverage-guided Kernel Fuzzing:
https://youtu.be/YwX4UyXnhz0

Linux Security Summit – syzbot and the tale of thousand kernel bugs – Dmitry Vyukov:
https://events19.linuxfoundation.org/wp-content/uploads/2017/11/Syzbot-and-the-Tale-of-Thousand-Kernel-Bugs-Dmitry-Vyukov-Google.pdf

PCGamer – Linux founder tells Intel to stop inventing 'magic instructions' and 'start fixing real problems':
https://www.pcgamer.com/linux-founder-tells-intel-to-stop-inventing-magic-instructions-and-start-fixing-real-problems/

Youtube – hupstream – Kernel Recipes 2016 - From ‘git tag’ to the front page - Konstantin Ryabitsev:
https://www.youtube.com/watch?v=vohrz14S6JE

Linux Kernel Monkey Log – Linux Kernel Release Model:
http://kroah.com/log/blog/2018/02/05/linux-kernel-release-model/

Youtube – hupstream – Kernel Recipes 2019 - CVEs are dead, long live the CVE!:
https://www.youtube.com/watch?v=HeeoTE9jLjM

Linux Kernel Monkey Log – What Stable Kernel Should I Use?:
http://kroah.com/log/blog/2018/08/24/what-stable-kernel-should-i-use/

Operating system distribution security contact lists:
https://oss-security.openwall.org/wiki/mailing-lists/distros

ZDNet – Google to Samsung: Stop messing with Linux kernel code. It's hurting Android security:
https://www.zdnet.com/article/google-to-samsung-stop-messing-with-linux-kernel-code-its-hurting-android-security/

Linux Security Summit – The State of Kernel Self Protection – Kees Cook:
https://events19.linuxfoundation.org/wp-content/uploads/2017/11/Sub-system-Update-Kernel-Self-Protection-Project-Kees-Cook-Google.pdf

ITWire – Red Hat dev questions why older Linux kernels are patched quietly:
https://www.itwire.com/security/red-hat-dev-questions-why-older-linux-kernels-are-patched-quietly.html

Wikipedia – Open source software:
https://en.wikipedia.org/wiki/Open-source_software

Re: Linux kernel: CVE-2018-14619 kernel: crash (possible privesc) in kernel crypto subsystem:
https://seclists.org/oss-sec/2018/q3/209

The Register – Hidden Linux kernel security fixes spotted before release – by using developer chatter as a side channel:
https://www.theregister.com/2020/09/04/linux_kernel_flaw_detection/

Appendix III: About Atredis Partners

Atredis Partners was created in 2013 by a team of security industry veterans who wanted to prioritize offering quality and client needs over the pressure to grow rapidly at the expense of delivery and execution. We wanted to build something better, for the long haul.

In six years, Atredis Partners has doubled in size annually, and has been named three times to the Saint Louis Business Journal's "Fifty Fastest Growing Companies" and "Ten Fastest Growing Tech Companies". Consecutively for the past three years, Atredis Partners has been listed on the Inc. 5,000 list of fastest growing private companies in the United States.

The Atredis team is made up of some of the greatest minds in Information Security research and penetration testing, and we've built our business on a reputation for delivering deeper, more advanced assessments than any other firm in our industry.

Atredis Partners team members have presented research over forty times at the BlackHat Briefings conference in Europe, Japan, and the United States, as well as many other notable security conferences, including RSA, ShmooCon, DerbyCon, BSides, and PacSec/CanSec. Most of our team hold one or more advanced degrees in Computer Science or engineering, as well as many other industry certifications and designations. Atredis team members have authored several books, including *The Android Hacker's Handbook*, *The iOS Hacker's Handbook*, *Wicked Cool Shell Scripts*, *Gray Hat C#*, and *Black Hat Go*.

While the Atredis client base is strictly confidential, and engagements often operate under stringent nondisclosure agreements, Atredis has delivered notable public security research on improving the security of Google, Motorola, Microsoft, Samsung and HTC products, and were the first security research firm to be named in Qualcomm's Product Security Hall of Fame. Atredis has received four research grants from the Defense Advanced Research Project Agency and has identified entirely new classes of vulnerabilities in hardware, software, and the infrastructure of the World Wide Web.

In 2015, we expanded our services portfolio to include a wide range of advanced risk and security program management consulting, expanding our services reach to extend from the technical trenches into the boardroom. The Atredis Risk and Advisory team has extensive experience building mature security programs, performing risk and readiness assessments, and serving as trusted partners to our clients to ensure the right people are making informed decisions about risk and risk management.