Clarifications on the “hypocrite commit” work (FAQ)

Qiushi Wu and Kangjie Lu, *University of Minnesota*

December 15, 2020

We recently finished a work that studies the patching process of OSS. Its goal is to improve the security of the patching process. The corresponding paper has been accepted by IEEE S&P 2021. I shared the abstract of the paper on Twitter, which then resulted in heated discussion and pushback. I apologize for the misleading abstract which did not show the details and caused many confusions and misunderstandings. Therefore, we would like to make a few clarifications.

We would like to first mention that we are a young research group with improving the kernel security as the first priority. In the past several years, we devote most of our time to improving the Linux kernel, and we have found and fixed more than one thousand kernel bugs; the extensive bug finding and fixing experience also allowed us to observe issues with the patching process and motivated us to improve it. Thus, we consider ourselves security researchers as well as OSS contributors. We respect OSS volunteers and honor their efforts. We have never intended to hurt any OSS or OSS users. We did not introduce or intend to introduce any bug or vulnerability in OSS. The following are the clarifications to the common concerns we received.

*** The purpose and research value of the work**

The project aims to improve the security of the patching process in OSS. As part of the project, we study potential issues with the patching process of OSS, including causes of the issues and suggestions for addressing them. This study indeed reveals some issues, but its goal is to call for efforts to improve the patching process---to motivate more work that develops techniques to test and verify patches, and finally to make OSS safer.

In this work, we collect 138 previous bug-introducing patches (not introduced by us). Based on these patches, we summarize their patterns, study specific reasons why bug-introducing patches are hard to catch (with both a qualitative and a quantitative analysis), and more importantly, provide suggestions to addressing the problem. In this work, we introduce the concept of “immature vulnerability” where a vulnerability condition of it is missing, but it can be turned into a real one when the condition is implicitly introduced by a patch for another bug. We also develop tools that help us find code places that may suffer from bug-introducing patches, and suggest what may make these bug-introducing patches hard to catch.

*** Did the authors introduce or intend to introduce a bug or vulnerability?**

No. As a part of the work, we had an experiment to demonstrate the practicality of bug-introducing patches. This is actually the major source of the raised concerns. In fact, this experiment was done safely. **We did not introduce or intend to introduce any bug or vulnerability in the Linux kernel**. All the bug-introducing patches stayed only in the email exchanges, without being adopted or merged into any Linux branch, which was explicitly confirmed by maintainers. Therefore, the bug-introducing patches in the email did not even become a Git commit in any Linux branch. None of the Linux users would be affected. The following shows the specific procedure of the experiment.

*** The procedure of the experiment**

We carefully designed the experiment to ensure safety and to minimize the effort of maintainers.

(1). We employ a static-analysis tool to identify three "immature vulnerabilities" in Linux, and correspondingly detect three real minor bugs that are supposed to be fixed. The "immature vulnerabilities" are not real vulnerabilities because one condition (such as a use of a freed object) is still missing. The "immature vulnerabilities" and the three minor bugs are independent but can be related by patches to the bugs.

(2). We construct three incorrect or incomplete minor patches to fix the three bugs. These minor patches however introduce the missing conditions of the "immature vulnerabilities", so at the same time, we prepare three other patches that correct or complete the minor patches.

(3). We send the incorrect minor patches to the Linux community through email to seek their feedback.

(4). Once any maintainer of the community responds to the email, indicating "looks good", we immediately point out the introduced bug and request them to not go ahead to apply the patch. At the same time, we point out the correct fixing of the bug and provide our proper patch. In all the three cases, maintainers explicitly acknowledged and confirmed to not move forward with the incorrect patches. This way, we ensure that the incorrect patches will not be adopted or committed into the Git tree of Linux.

We did check different versions of Linux and further confirmed that none of the incorrect patches was adopted. To conclude, we ensured that the experiment was done safely, and all the incorrect patches stayed only in the email exchanges and will not be merged into or even become a Git commit in any Linux branch.

*** Is this human research?**

This is not considered human research. This project studies some issues with the patching process instead of individual behaviors, and we did not collect any personal information. We send the emails to the Linux community and seek community feedback. The study does not blame any maintainers but reveals issues in the process. The IRB of UMN reviewed the study and determined that this is not human research (a formal IRB exempt letter was obtained).

Throughout the study, we honestly did not think this is human research, so we did not apply for an IRB approval in the beginning. We apologize for the raised concerns. This is an important lesson we learned---Do not trust ourselves on determining human research; always refer to IRB whenever a study might be involving any human subjects in any form. We would like to thank the people who suggested us to talk to IRB after seeing the paper abstract.

*** Would this work leak the personal information of the maintainers?**

No**.** (a). The procedure will not collect any personal data, individual behaviors, or personal opinions. It is limited to studying the patching process these communities follow, instead of individuals. All of these emails were sent to the communities instead of individuals. (b). We have tried our best to protect the email information of maintainers. In particular, to protect the maintainers from being searched through our emails, we used a random email account.

*** Is the Linux community informed?**

Bug-introducing patches is a known problem in the Linux community, as multiple prior papers showed (details are in the related work section of the paper). We also emailed the community to inform them that malicious committers can introduce bugs, and they acknowledged that they knew patches could unintentionally or intentionally introduce other bugs, and they will do their best to review them. Before the paper submission, we also reported our findings to the Linux community and obtained their feedback.

*** Does this project waste certain efforts of maintainers?**

Unfortunately, yes. We would like to sincerely apologize to the maintainers involved in the corresponding patch review process; this work indeed wasted their precious time. We had carefully considered this issue, but could not figure out a better solution in this study. However, to minimize the wasted time, (1) we made the minor patches as simple as possible (all of the three patches are less than 5 lines of code changes); (2) we tried hard to find three real bugs, and the patches ultimately contributed to fixing them.

*** The work taints the relationship between academia and industry**

We are very sorry to hear this concern. This is really not what we expected, and we strongly believe it is caused by misunderstandings. The work honors maintainer efforts and is purposed to improve the Linux kernel. When there is a security weakness, both academia and industry should work together to fix it, instead of overlooking it. More importantly, users of OSS have the right to know the potential risks; on the other hand, exposing the issue has clear benefits for the OSS community because it calls for efforts to fix the issue. It would motivate researchers and professionals to develop tools that automatically test and verify the patches, which would alleviate maintainer burden. One thing clear is that this is a hard and urgent issue that requires collaborations between academia and industry. We are eager to continuously work with the community and develop useful tools to help with the patch review and bug detection. We believe our goal is the same as maintainers---to make OSS safer and the OSS communities more prosperous.

In fact, there are many great examples in which the exposure of security weaknesses by researchers effectively improved the security. For example, in 2013, we demonstrated that Apple's app review process was flawed (attackers could upload malware that invokes private APIs)[1]; because of the exposure, Apple soon hardened the system and introduced process-based isolation for private APIs. Therefore, we also believe that this work would actually improve Linux-kernel security.

*** Suggestions to improving the patching process**

In the paper, we provide our suggestions to improve the patching process.

- OSS projects would be suggested to update the code of conduct, something like "By submitting the patch, I agree to not intend to introduce bugs".
- We need automated tools for testing and verifying patches. The relevant techniques include directed fuzzing, (under-constrained) symbolic execution, formal methods, etc. More details are in the paper.
- OSS maintaining is understaffed. We should very much appreciate and honor maintainer efforts, and increase potential incentives if possible to encourage more people to join the maintaining.
- We hope both reporters and maintainers are aware of the potential bug-introducing patches. Also, tools can be developed to check "immature vulnerabilities".

If you have further concerns, please email us at kjlu@umn.edu.

References

[1] “Jekyll on iOS: When Benign Apps Become Evil.” Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, and Wenke Lee. In Proceedings of t*he 22th USENIX Security Symposium*. Washington, DC, August 2013.