`Free' as in Freedom to Protest?

Fabio Massacci, Antonino Sabetta, Jelena Mirkovic, Toby Murray, Hamed Okhravi, Mohammad Mannan, Anderson Rocha, Eric Bodden, Dan Geer

Fabio Massacci, Antonino Sabetta

Open Source Software development is a technical job, which should make it kind of immune to bias. *We trust developers to do the right (technical) thing* and this trust has become even more important as the software supply-chain has become critical for software vendors who bundle or rely on open-source components.

The recent days proved us wrong. We discuss in this piece several different perspectives from the editorial board.

To kick-start the discussion, let's first review some of the recent attacks. In the ***node-ipc*** case[1] a developer pushed an update which ***deliberately but stealthily*** included code that sabotaged the computer of the users who installed the updated component. Such an attack was selective: a DarkSide on the reverse. If the computer IP was geo-located in Russia the attack would be launched. Several days and a few million downloads later, the `spurious code' was actually noticed and investigated. Linus's law on the many eyes *eventually* made the bug shallow[2] and the developer pulled back the changes.

In the node-ipc 'impulse hacktivism attack', the service invocation looked like this:

aHR0cHM6Ly9hcGkuaXBnZW9sb2NhdGlvbi5pby9pcGdlbz9hcGlLZXk9YWU1MTFlMTYyNzgyNGE5NjhhYWFhNzU4YTUzMDkxNTQ=

which is the base-64 encoded form of this:

https://api.ipgeolocation.io/ipgeo?apiKey=ae511e1627824a968aaaa758a5309154

[1] https://arstechnica.com/information-technology/2022/03/sabotage-code-added-to-popular-npm-package-wiped-files-in-russia-and-belarus/

[2] Meneely, A., & Williams, L. (2009, November). Secure open source collaboration: an empirical study of Linus' law. In *Proceedings of the 16th ACM conference on Computer and communications security* (pp. 453-462).

This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

The case of **styled-components**[3] is an interesting combination of a **malicious but ultimately innocuous** code change, whose intended effect was to print a message on the console of users whose system locale was set to "ru_RU", with an **accidental but disastrous mistake**: the developer forgot to include a required file (*postinstall.js*) without which the package installation would fail, causing a chain of failures in the build of its dependent projects[4]. Only after the missing file was added, the (malicious) code could eventually "work as designed".

Pushed by a different motivation, the developer of npm packages ***colors and faker***[5] ***deliberately and openly*** introduced errors in their code thus making them de facto unusable. In this case, the developer even created an issue on the software repository explaining he could not continue working on the project for free while not actually getting any revenues out of it. So he challenged other developers to actually come to the rescue. None arrived. Apparently the `free as in freedom'[6] had become `free as in free beer'[7], or this is a sign of deeper issues in the model through which commercial vendors have interacted with the open-source community so far[8].

Jelena Mirkovic

In the past years, we have seen a tendency in public opinion to sway widely into light-and-shadow, polarizing depictions of various social issues ranging from immigration, vaccines, climate change to more lightweight quarrels at the Oscars. It is not surprising that wars are another polarizing issue where sides are firmly taken and opponents are vilified, and damages are bestowed upon them because `they deserve it'.

Anderson Rocha

The more society lives in so-called social bubbles and opinions are in different extremes the more we will see different forms of protest, sometimes harming public confidence even in pieces of software that historically were considered clean. It is paramount to devise forms of protecting the development of high-quality software from malicious pieces of "demonstrators". Perhaps adopting some sort of WIKI-like development with more trustful developers acting as gate-keepers to guarantee good software development practices would be a starting point.

Fabio Massacci, Antonino Sabetta

We expect black-hat hackers to have mischievous tactics to avoid detection and possibly collude with corrupt or inefficient law enforcement. Darkside, the group behind the colonial pipeline attack, has been famous for not installing its ransomware on machines having a Ukrainian or Russian keyboard. The final list of safe haven keyboards has been published in

---

[3] https://www.npmjs.com/package/styled-components
[4] https://checkmarx.com/blog/new-protestware-found-lurking-in-highly-popular-npm-package/
[5] https://snyk.io/blog/open-source-npm-packages-colors-faker/
[6] Stallman, R. (2009). Viewpoint Why" open source" misses the point of free software. Communications of the ACM, 52(6), 31-33.
[7] Greenstein, S. (2018). Free Software without a Free Lunch or Free Beer. *IEEE Micro*, *38*(5), 94-96.
[8] https://christine.website/blog/open-source-broken-2021-12-11

Brian Krebs' blog[9]. Some of us might even accept state or police sponsored hacking as a technically appropriate solution. Stuxnet is an example[10] (good or bad depending on which side you are on), the Encrochat hacks for surveillance of drug cartels[11] is another one, and we might argue that the ACM Code of Ethics is good in theory[12] but pretty bad on the concretely chosen example `The Rogue' where sending worms on another country's ISP received a bill of health[13].

What we have underestimated is that `good' developers upon which the Open Source Community is made are actually humans and they can themselves do the equivalent of hurling bottles at the despised members of the `other party'.

Jelena Mirkovic
This is a slippery slope. Nothing in life is black-and-white and wars especially are messy, complicated, serious and leave long-living scars in generations of both winners and losers.

Hamed Okhravi
Even further, how can one ensure that the implanted version of the codebase does not spread to other regions? Attackers do not have a successful track record of ensuring that a piece of malware can only impact the intended target, even when it comes to highly-targeted malware; think Stuxnet[10]. How can one ensure that a piece of software that is meant to be shared does not spread beyond a certain region? IP geolocation is way too inaccurate. With widespread usage of consumer-grade VPNs these days, IP geolocation can be even more inaccurate. Consider this scenario: a user connects to a VPN service to bypass regional lock for a movie and while they are watching the movie, a service on their machine automatically updates, but grabs the infected version of a library by mistake because the machine looks like it is in a different region

Toby Murray
We might decide to consider these events exceptions: history to date has demonstrated that a developer is likely to have a vulnerable product because either they didn't fix their own code or they did not keep a third-party dependency up-to-date, the former case being more frequent than the latter once the code under control of developers has been properly counted.[14] Having a vulnerability because they updated to a version which included unwanted or malicious updates is way less likely. After all, developers with the best of intentions who unwittingly introduce vulnerabilities are far more common than those who intentionally insert vulnerable or unwanted functionality.

---

[9] https://krebsonsecurity.com/2021/05/try-this-one-weird-trick-russian-hackers-hate/
[10] Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, *9*(3), 49-51.
[11] O'Rourke, C. (2020). Is this the end for 'encro'phones?. *Computer Fraud & Security*, *2020*(11), 8-10.
[12] Gotterbarn, D., Bruckman, A., Flick, C., Miller, K., & Wolf, M. J. (2017). ACM code of ethics: a guide for positive action. *Communications of the ACM*, *61*(1), 121-128.
[13] Kirkpatrick M.S. Case Studies. in *ACM Code of Ethics and Professional Conduct*. DOI 10.1145/3274591
[14] Pashchenko, I., Plate, H., Ponta, S. E., Sabetta, A., & Massacci, F. (2020). Vuln4real: A methodology for counting actually vulnerable dependencies. *IEEE Transactions on Software Engineering*.

Fabio Massacci, Antonino Sabetta
If past is prologue, these events might also indicate a new factor of fragility of the overall software infrastructure. With protestware, there is a new reason for open-source software components to become the single point of failure of several companies and it has to do with political stances, personal beliefs and motivations, and even psychological and sociological factors. A risk that doesn't fare well on a company's 10-K form.

The coming of age of the open source world also brings the full realization of its key role in the economics of today's software industry. *Many cornerstone packages used in thousands of commercial products rely on one-dev-shows that receive no financial support from the corporations that benefit from them.* The frustration that ensues risks pushing more open-source developers to pull the plug on their projects.

So, the protests on too many free riders or the enemies of the country next door is just an example of the future likely failure of the governance of the process. The threat from an individual developer making changes to their product to limit or degrade its functionality for users in certain geographic regions, or to include outright hostile functionality, is magnified when that product is widely relied upon.

Hamed Okhravi
What would be the impact of silently implanted, intentional bugs in a codebase that may be used or deployed in various systems with various missions? Given the risk of spreading beyond the targeted region, how can one even begin to understand the secondary and tertiary impact of such bugs? Could they result in system crashes and possibly damages to safety-critical systems or even lives outside the targeted region? These questions are hard to answer because they get to the core of why cybersecurity is hard: reasoning about complex systems with many interdependencies is exceedingly difficult even in specific cases and intractable in the general case.

Fabio Massacci, Antonino Sabetta
While presumably the node-ipc modification was meant to go unnoticed (it was encoded in such a way not to be immediately readable) it was not "professionally stealthy" given that it used an obfuscation method that is rather primitive, to say the least. Also the geo-location service used is rather accurate but not infallible[15], so some end-users might have been impacted also outside of Russia and Belarus. The characteristics of the changes suggest that the technical execution of this action as well its actual (as opposed to the intended) consequences were not really thought through, but they were the result of some sort of "impulse hacking".

Developers who rely on third-party components may wish to keep their dependencies up-to-date, to ensure that they are taking advantage of recent security patches; yet when third-party developers introduce unwanted changes, those undermine trust in the supply chain and the

[15] The provider of the geo-location API used in the node-ipc case claims that, at the granularity of countries, it has a 99% accuracy (https://ipgeolocation.io/faq.html).

virtues of regularly patching third-party components. We are back to the conundrum discussed in the previous issue on Solarwinds.[16]

How can we distinguish the “normal” (unintentional) security bugs introduced in a software update from the “intentional” security bugs?

Anderson Rocha
As new forms of protest embedded into software takes place so does the need of developing checking software and protocols that could employ intelligent techniques for detecting security flaws in addition to a wiki-like development pipeline as mentioned previously. Perhaps an area of research that could be supported in this regard would be AI-based security flaw and protesware detection both in terms of checking written code and real-time software testing.

Hamed Okhravi
To understand how much intentional bugs can complicate the operation of the existing bug finding tools, consider why bug finding is hard. The difficulty of finding normal (unintentional) bugs is traditionally attributed to the corner cases in programming languages that hinder an accurate analysis[17]. Now consider that most research has focused on unintentional bugs that are introduced as a result of programmer error, which may occasionally face such corner cases. Intentional bugs introduced by protestware can make this analysis vastly more difficult and inaccurate by explicitly targeting the corner cases to further impede detection. Furthermore, with unintentional bugs, analysis has to be done once on a given codebase; with intentional bugs, the analysis needs to be repeated N times, where N is the number of different regional versions of the code to ensure that the specific versions of the codebase are not implanted with vulnerabilities.

[16] Massacci, F., Jaeger, T., & Peisert, S. (2021). Solarwinds and the challenges of patching: Can we ever stop dancing with the devil?. *IEEE security & privacy*, *19*(2), 14-19.
[17] M. Miller, “Trends, challenges, and strategic shifts in the software vulnerability mitigation landscape,” Blue Hat, Israel, 2019.

Dan Geer
Poisoning a common resource, be it an open source codebase or a municipal water plant, is a consequence of interdependence. In any such case of interdependence, any solution must be presumed to have have important side effects, e.g., faultlessly geocoding the Internet by global treaty might make extending the Westphalian principle whereby a sovereign is culpable for attacks emanating from its territory into a matter of settled international law yet the side effects might well favor authoritarians. Similarly, making a service provider culpable for its use of code regardless of whether it wrote it itself or incorporated it from an open source pool might well accelerate the balkanization of the Internet while boosting cyber-industrial consolidation. As others have implied, keeping honest people honest is a high goal but one that is feasible. Keeping dishonest people honest is far harder and may not be feasible. As the line between data and code blurs with the proliferation of machine learning, provenance likely becomes undecidable.

Eric Bodden
Data models that have been pre-trained by third parties are a new type of common resource. Their increasingly widespread use poses challenges that have not been discussed in enough depth in the security community so far. Like software components, they too build an increasingly important supply chain, yet at the same time the trust that we can put in these models is quite unclear. Who trains these models and to what extent? How may these people be influenced, how may the resulting models be biased? What may be the impact of these biases? These are questions that we need to discuss, also from a geo-strategic point of view.

Jelena Mirkovic
One opportunity is to rethink open source and supply chain verification and governance systems. We are in need of “three laws of robotics” but for computer code.

Fabio Massacci, Antonino Sabetta
Discussion of changes to the source code driven by considerations that are political rather than technical have taken place in other projects as well. A proposal was made on March 2nd 2022 in the OpenBLAS project repository[18] to drop the support for Russian-produced Elbrus E2000 processor family, based on the fact that "*the Elbrus processor is a so-called homegrown processor, with the primary use case of circumventing sanctions*"[19] and that it is used by the Russian military and intelligence organizations. Along pour ingrained expectation that Internet standards should be based on technical merit[20], members of the OpenBLAS community have pointed out that: (1) the Elbrus processor has a compatibility mode that allows it to run even code built for other architectures; (2) dropping Elbrus support would have no effect on any forks that might exist internally to the Russian military and intelligence orgs; (3) there are indeed legitimate civilian uses of that architecture. Also, they stated their belief that the decision to

[18] https://github.com/xianyi/OpenBLAS
[19] https://github.com/xianyi/OpenBLAS/issues/3551
[20] Russell, A. L. (2006). 'Rough consensus and running code' and the Internet-OSI standards war. *IEEE Annals of the History of Computing*, *28*(3), 48-61.

include or drop support for a certain architecture should be driven purely by technical considerations and not political ones.

Jelena Mirkovic

A solution might also be on the technical side, can we develop methods that detect and interpret functional changes in code between its versions, especially when focusing on data exfiltration or data modification? Can we develop better code integration processes to detect and weed out unwanted additions? How can we develop code development systems that uphold some code functionality principles by themselves, rejecting insider attacks?

Mohammad Mannan

Protestware should be treated as a new security problem by the security community that needs to be addressed. There are decent tools now for detecting (exploitable) software bugs due to developers' mistakes, both at the source/machine-code levels; and some tools also exist to detect backdoors and covert communication channels (with varying levels of success). There isn't anything on protestware, and the challenges introduced by protestware are also somewhat unique. For example, we need to consider unusual resource consumption, software/hardware corruption, and even the possibility of malicious or intentionally incorrect computation/logic bugs. Checking for context-based (e.g., ISP, geo-location, or keyboard settings) exceptional execution paths could be a starting point for detection. However, developing an effective solution would take time and effort (albeit exciting for researchers!), but spending resources on this may be essential as open-source projects are used by many government and business operations.

GLOBAL CONCLUSION

We may debate whether it is appropriate or not -- some developers will take an activist role and make malicious changes to their code, in situations like the current war in Ukraine. Even if there are serious consequences to people outside the developers' target, this may continue to happen in the future -- as developers are humans too, and humans aren't always rational. Working on these complex issues can help us prevent future supply chain attacks, which can bite friend and foe alike.

Acknowledgements

Part of this material is based upon work supported under Air Force Contract No. FA8702-15-D-0001 and the European Commission H2020 Programme under grant 952647(AssureMOSS). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of their employers, or the U.S. Air Force, or the European Commission. This work is also supported by the São Paulo Research Foundation - FAPESP - under grant DéjàVu No. 2017/12646-3.