liblzma - OSS and Backdoors

Exploring the xz-utils Backdoor, its Emergence and its Impact on FOSS and OSS

9525469
xnacly.me
Applied Computer Science
DHBW Mosbach

June 11, 2024

Abstract

In recent years, several vulnerabilities in the open-source software supply chain were discovered. The most recent being the intentionally placed backdoor in the compression library named *liblzma*. This paper aims to explore the implementation of said backdoor while highlighting the insertion of the backdoor and the inserters use of social engineering enabling their placement in the leadership of the project. Furthermore ways of preventing similar attacks are presented and evaluated on the example of the *liblzma* situation.

1 Introduction

FOSS[1] is generally defined as software the user can "[...] run, copy, distribute, study, change and improve [...]" [2]. This requires the source to be available and enables the dependence of other software on subsets or the entirety of the code. On the other hand, source available or OSS[2] are distinct from FOSS software. Some licenses do not require the resulting product to be licensed under the same license as its dependencies, such as the MIT license[3]. It therefore differs from the GPL[4] and software licensed with the MIT-Licence can therefore not be referred to as free open-source software, but rather as open-source software.

Most OSS-projects accept contributions from individuals and enterprises. This is wanted and required to support the actuality of said software. Most OSS projects accept changes matching their pre-defined contribution guidelines and credit the contributor for their addition. These contributors often use the software they are contributing to and therefore make changes they care for, such as adding drivers for new devices to the linux kernel [5]. However, other independent OSS contributors are abusing the contribution system by exploiting the trust the unpaid maintainers have in the quality of the submitted changes. As was the case with `liblzma` or the `xz-utils` OSS library.

[1] Free and Open-Source Software [1]

[2] Open-Source Software

[3] Requires the license to be present in "all copies or substantial portions of the Software" [3]

[4] Requires all copies of the software to be licensed as GPL [4]

1.1 Dependence on FOSS and OSS

Open source software is often divided into reusable components, such as libraries or toolkits implementing a specific feature, and built upon by other software. The goal is to use tried and tested components in the creation of new OSS, thus building on field tested and established software found in the OSS community.

Not only does OSS depend on other libraries from the OSS ecosystem. Proprietary software also makes use of said OSS components, while being forced to adhere to their terms, as declared in their respective licenses [6].

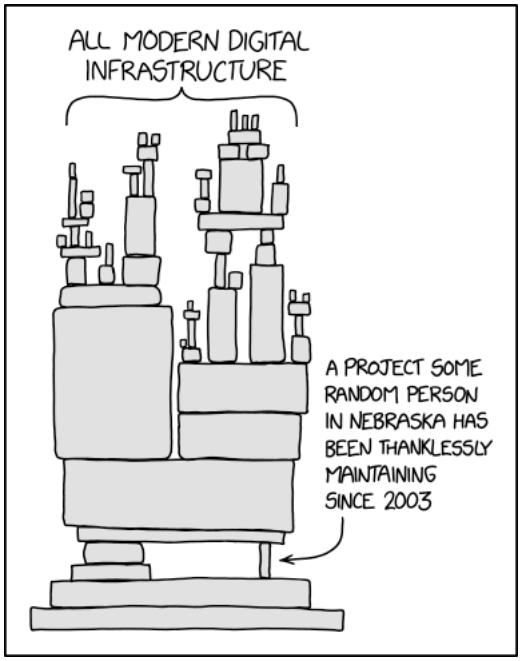

Figure 1: Dependency [7]

Commonly used examples for said libraries are `libcurl`, which provides multiprotocol file transfers [8], `raylib` which is a libary for videogames programming [9] and the `sqlite` library, that implements a in process database [10]. These library examples are so widely used, a vulnerability in them would impact the security of the whole software space.

1.2 Supply Chain Security

[11]

1.3 xz-utils and liblzma

[12]

2 Backdoor Exploration

2.1 Implementation

2.2 Social Engineering

2.3 Pressure on OSS Maintainer

2.4 Affected Systems

3 Response

3.1 Patches

3.2 Releases on Hold

3.3 Vetting Source Code

4 Prevention

4.1 Funding FOSS and OSS

4.2 Vetting Dependency

4.3 Appreciation for FOSS Maintainers

References

[1] R. M. S. (RMS). (2021), [Online]. Available: `https://www.gnu.org/philosophy/pragmatic.html` (visited on 06/04/2024).

[2] F. S. Foundation. (2024), [Online]. Available: `https://www.gnu.org/philosophy/free-sw.html` (visited on 06/04/2024).

[3] Opensource.org. (2024), [Online]. Available: https://opensource.org/license/MIT (visited on 06/04/2024).

[4] Opensource.org. (2024), [Online]. Available: https://opensource.org/license/gpl (visited on 06/04/2024).

[5] T. kernel development community. “Linux - introduction - device drivers.” (), [Online]. Available: https://linux-kernel-labs.github.io/refs/heads/master/lectures/intro.html#device-drivers (visited on 06/10/2024).

[6] D. Stenberg. “Companies using curl in commercial environments.” (2024), [Online]. Available: https://curl.se/docs/companies.html (visited on 06/09/2024).

[7] xkcd. “Dependency.” (), [Online]. Available: https://xkcd.com/2347/ (visited on 06/09/2024).

[8] D. Stenberg. “Libcurl - the multi-protocol file transfer library.” (2024), [Online]. Available: https://curl.se/libcurl/ (visited on 06/09/2024).

[9] raysan5. “Raylib is a simple and easy-to-use library to enjoy videogames programming.” (2024), [Online]. Available: https://www.raylib.com/ (visited on 06/09/2024).

[10] D. R. Hipp. “What is sqlite?” (2024), [Online]. Available: https://www.sqlite.org/index.html (visited on 06/09/2024).

[11] M. Stapelberg. “Supply chain security with go.” (2024), [Online]. Available: https://media.ccc.de/v/gpn22-438-supply-chain-security-with-go (visited on 06/07/2024).

[12] cy. “Common code <> different backdoors.” (2024), [Online]. Available: https://media.ccc.de/v/gpn22-304-common-code-different-backdoors (visited on 06/05/2024).

Appendix