Open-Source vs. Proprietary Software:

A Case Study in Trust and Security

2020 International Conference on Nuclear Security (ICONS), 10-14 February 2020, Vienna, Austria

Keith S. Morgan (morgank@lanl.gov) & Paul S. Graham (grahamp@lanl.gov)

Los Alamos National Laboratory, Los Alamos, New Mexico, United States of America

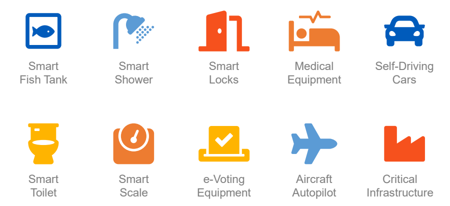
Introduction

Software plays an ever-increasing trusted role in all aspects of life. We[1] trust the software in self-driving cars to get us to our destination safely. We[1] trust the software in our smart locks to keep intruders out of our houses. We[1] even trust the software in our smart tank to feed our goldfish. The software running nuclear facilities is no exception; it requires trust in the functionality, safety, security etc. As the use of digital systems in nuclear facility systems continues to rise, trust in the software powering those systems plays an increasingly important role in nuclear security.

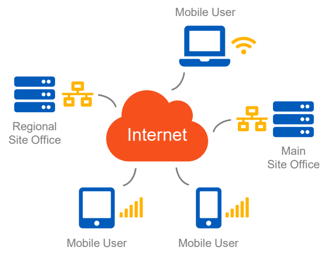
In this work we examine the trustworthiness and security of open-source vs. closed-source (i.e. proprietary) software. We define each in terms of licensing models, source release models, business models and so forth. We attempt to enumerate the advantages and disadvantages of each, particularly with respect to trust and security. Finally, we present a case study in the software powering virtual private network (VPN) devices, important network security components specifically recommended for nuclear security applications by the International Atomic Energy Agency (IAEA) [1] and the United States Nuclear Regulatory Commission (NRC) [2].

[1]Perhaps, more accurately, "some of us".

Smart Stuff



Virtual Private Network



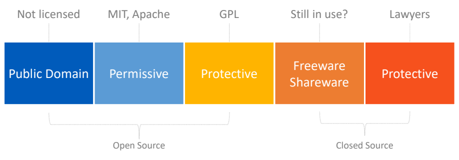
Background

A distinguishing factor between open- and closed-source software is the licensing model on the licensing continuum [3]. On the left are open-source models which allow source distribution with certain restrictions. The "permissive" licenses require attribution. The "protective" licenses require the same license in derivative works. On the right are closed source (i.e. proprietary) models.

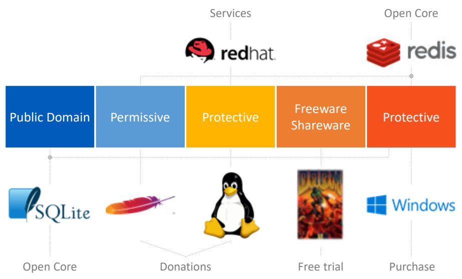
Another factor is the business model. Proprietary software survives on sales of the software. Open-source projects, on the other hand, rely on donations, paid memberships, in-kind donations, paid services, or paid premium features (a.k.a. open-core).

Another aspect is the development model. Proprietary software development is typically a black box and therefore no trust information can be derived. On the open source-side, however, there are many facets that impact trust. One facet is project stewardship. The steward might be a non-profit foundation or one individual (a.k.a. "Benevolent Dictator for Life"). A second facet is the development process, specifically with respect to community involvement. For example, some projects accept community contributions while other have very limited community involvement.

Software Licensing Continuum



Software Development Business Models

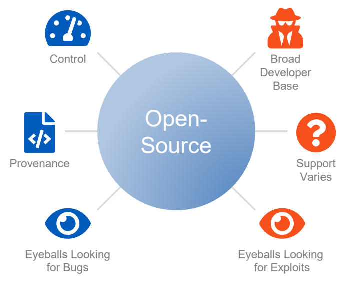


Pros and Cons

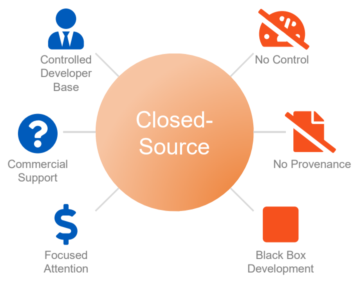
One of the major advantages of open-source software is freedom for end users – freedom to control their own destiny (fix bugs, add features, etc.), freedom to audit the source code and freedom to trace the provenance, or origin, of the software. Another major advantage, with respect to trust, is crowd sourcing. Linus Torvalds famously said, "given enough eyeballs, all bugs are shallow." Crowd sourcing, however, can also be a disadvantage. Eyeballs may be looking for exploits. Open-source developers may also have varying standards of quality, motives, etc. Finally, support can vary wildly from project to project.

The advantages and disadvantages of closed-source (i.e. proprietary) software are primarily the inverse of the advantages and disadvantages of open-source software. Development behind closed doors means there are no prying eyeballs looking for exploits. The software owner can tightly control the code and developer base. Support is more common and financial interests may lead to more focused attention (i.e. people time) on the code. On the other hand, closed source software takes freedom and control from the end users. It is impossible to trace the provenance of the code and auditing is typically infeasible and may even be actively discouraged.

Open-Source Pros & Cons



Closed-Source Pros & Cons



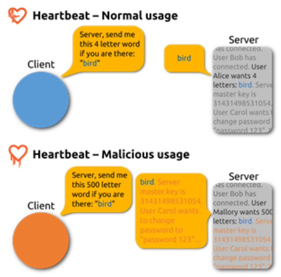
Case Study

To highlight some of the differences between open-source and closed-source software, we present a case study of two public vulnerabilities discovered in VPN technology.

Juniper Networks sells a line of VPN devices powered by its closed-source ScreenOS software. In 2015 Juniper Networks announced that during an internal code review they had discovered "unauthorized code in ScreenOS that could allow a knowledgeable attacker to gain administrative access to NetScreen devices and to decrypt VPN connections" [4]. An unidentified actor had surreptitiously planted a back door with a master password into the source code.

OpenSSL is an open-source cryptography library. In 2014 security researchers identified a bug in the OpenSSL implementation of the TLS heartbeat extension [5]. TLS is the protocol used to secure web communications – it is also used in some VPN configurations. A PhD student in Germany, who co-authored the extension, wrote an implementation for OpenSSL. One of the core OpenSSL developers reviewed the code but missed a bug that allowed an attacker to read memory of vulnerable systems, compromising private information (e.g. keys, passwords, etc.).

OpenSSL CVE-2014-0160 a.k.a. Heartbleed


Heartbeat – Normal usage
Client
Server, send me this 4 letter word if you are there: "bird"
bird
Server
User Bob has connected. User Alice wants 4 letters: bird. Server master key is 31431498531054. User Carol wants to change password "password 123".
Heartbeat – Malicious usage
Client
Server, send me this 500 letter word if you are there: "bird"
bird. Server master key is 31431498531054. User Carol wants to change password to "password 123"...
Server
User Bob has connected. User Mallory wants 500 letters: bird. Server master key is 31431498531054. User Carol wants to change password "password 123".

Juniper ScreenOS CVE-2015-7755

Juniper

Important Announcement about ScreenOS®

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT

Summary

Two good reasons one might trust open-source software are: 1) the freedom to audit source code, and 2) the many benefits of crowd sourcing. Two good reasons one might trust proprietary software are: 1) no prying eyeballs, and 2) a controlled development. However, the case study presented in this work showed counter examples of each approach. A major vulnerability was discovered in the open-source OpenSSL project due to community-contributed code that was not properly vetted. Likewise, two major vulnerabilities were discovered in the proprietary Juniper ScreenOS software despite a supposedly controlled development environment.

There is not a one-size-fits-all solution. The best solution for a particular use case will depend on requirements and the specifics of the available products. By carefully weighing the factors presented in this work, one can reasonably evaluate the options and pick the best option to maximize trust and security.

References

[1] International Atomic Energy Agency, 2011. IAEA Nuclear Security Series No. 17. Computer Security at Nuclear Facilities.

[2] J. T. Michalski, F. J. Wyant, and D. Duggan, Secure Network Design Techniques for Safety System Applications at Nuclear Power Plants. Sandia National Laboratories, 2010.

[3] Webbink, Mark. Open source from a Proprietary Perspective. Available online: https://web.archive.org/web/*/https://www.redhat.com/f/summitfiles/presentation/May31/Open%20Source%20Dynamics/Troan_OpenSourceProprietyPersp.pdf

[4] https://forums.juniper.net/t5/Security-Incident-Response/Important-Announcement-about-ScreenOS/ba-p/285554

[5] http://heartbleed.com/

Image Credits