NERC
NORTH AMERICAN ELECTRIC RELIABILITY CORPORATION

Security Guideline for the Electricity Sector - Supply Chain

Risk Considerations for Open Source Software

The objective of the reliability guidelines is to distribute key practices and information on specific issues critical to promote and maintain a highly reliable and secure bulk power system (BPS). Reliability guidelines are not binding norms or parameters to the level that compliance to NERC's Reliability Standards is monitored or enforced. Rather, their incorporation into industry practices is strictly voluntary.

Open Source Software

"Open source software is software with source code that anyone can inspect, modify, and enhance".[1] It may or may not also be available for download in binary form, ready to use. Common examples of open-source software are The Linux Operating System, Mozilla Firefox, and Google Chrome.

Open-source software is built periodically, incorporating submissions from a community of contributing developers to implement new features, address security vulnerabilities, and fix bugs. Users obtain the software, as well as updates when new versions are released, from a distribution point maintained by the community.

User trust in open-source software involves a combination of trust in the distribution point for the software, "faith" in the supporting community of developers and supporters, and observations from inspecting the code, each in varying degrees depending upon the user's technical capability. Most users are "downloaders", entirely dependent on the development and user communities for security support.

Open-source projects differ widely in their assumptions regarding the technical capabilities of their users, the thoroughness of documentation, and the level of support from the developer community. Because there is no buyer-seller relationship or contract, the user is responsible for evaluating his ability to support his use of the product himself, which includes proper installation, configuration, and ongoing monitoring for security vulnerabilities and updates.

Analyzing Operational Risk

A useful way to analyze risk and set priorities for reducing risk is to create a *threat model* (operational environment for the software) for the system being considered. A threat model describes the things that affect the system, the things the system can affect, and the criticality of those things. It also describes what can go wrong, either accidental or deliberately induced by an attacker, and how likely those are, as

[1] https://opensource.com/resources/what-open-source

well as ways those threats can be reduced. Shown below is a list of some common things to consider when analyzing operational risks for software.

- Does the software require an Internet connection?
- Can the system be reached by an internal network user in a different department?
- How directly can the system be reached by an external attacker?
- What would be the impact (criticality) if the system failed (went offline or was disabled)?
- What would be the impact (criticality) if the system mis-operated (had inappropriate outputs)?
- What systems can be connected to or are reachable from the system and how critical are they?
- Where do the system's inputs come from and how trustworthy are those inputs?
- Does the software require or run with administrative privilege?
- How *trustworthy* is the software?

A threat model helps you to determine the type and extent of compensating security controls that might be needed to reduce risk to an acceptable level.[2]

Evaluating Trustworthiness of Software

Risks of software can be roughly divided into risk that the software might not be authentic (a malicious look-alike), and risk that the software has defects than can be exploited as security vulnerabilities or that might cause misoperation.

The risk of software being inauthentic or having been tampered with can be reduced by always obtaining the software from its original point of distribution, normally a web site maintained by its creators. The original distribution point can usually be found using an Internet search engine, or perhaps in blog posting by its users. The distribution site should be protected by TLS (an HTTPS URL), which allows you to check the web site certificate to ensure that it is valid and that the certificate was issued to a "subject" or "subject alternate name" that is consistent with expectations for the software.[3]

The risk from defects comes from a combination of the probability that defects will exist balanced with the existence of an active community of maintainers and their responsiveness in addressing defects.

Intelligence Sources

The Internet provides many resources that will provide evidence bearing on the activity, reputation, and trustworthiness both of a software product and of its development community.

- Use an Internet search engine to find the distribution point, and search for references or reputation

[2] https://en.wikipedia.org/wiki/Threat_model
[3] https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/

- https://www.virustotal.com/en - use to perform virus scan of files and URLs
- https://nvd.nist.gov/vuln/search?execution=e2s1 – search for reported security vulnerabilities
- https://scan.coverity.com/projects - code analysis statistics for some open source projects
- https://www.openhub.net – activity statistics for some open source projects

Case Studies

Studying some past cyber security issues in the software supply chain helps to understand these risks and illustrate how defensive principles can be applied.

- Havex[4]
  - An attacker compromised legitimate distribution points for proprietary industrial control system software, leading to compromise of the suppliers' customers when these versions were installed as updates.
  - Risk: An attacker can compromise a distribution point for software.
  - *Countermeasure*: Verify authenticity of software by verifying its digital signature or verifying the correct hash value.
- Heartbleed[5]
  - Heartbleed (CVE-2014-0160) was a serious implementation flaw in OpenSSL of the SSL "heartbeat" feature. It allowed a malformed "heartbeat" request to obtain replies containing confidential data like usernames and passwords that had been sent protected by TLS. It was originally discovered on April 3, 2014, and a patch was announced by the OpenSSL project just four days later, on April 7, 2014.
  - Heartbleed is notable because it illustrates that even active and well-supported projects can have serious issues.
  - Risk: Serious defects can and do arise, even in projects with an attentive developer community.
  - *Countermeasure:* Ensure that you understand where security announcements and advisories will be published and have a process to ensure that you look for and receive updates when they are released.

Unsupported Software

Both open-source and proprietary software can become *unsupported*, meaning that the software is no longer being actively developed or updated. Most seriously, this means that the application poses an *unknown* risk - any security issues discovered in the software will not be fixed, and even if vulnerabilities become known, you may not be made aware of them.

[4] https://ics-cert.us-cert.gov/advisories/ICSA-14-178-01
[5] https://www.us-cert.gov/ncas/alerts/TA14-098A http://xkcd.com/1354/

Organizations with their own development staff might be able to take over (“fork”) the open-source code base and continue development, or in the case of proprietary software, obtain the source code from its developers. Doing this requires bringing the code into a secure development process, including managing security vulnerabilities in its components. Few organizations are prepared to take ownership at this level.

Unsupported software should be avoided, especially in critical applications, but if you need to use unsupported software, the following measures can be used to minimize risk.

- Reduce attack surface[6]
  - remove or disable non-essential features
  - reduce privilege level (software should not run with administrative privilege)
- Control access
  - Restrict access to users cognizant of the risks and trained in secure operation procedures
- Isolate
  - prevent access to or from the Internet and tightly control access to the internal network using restrictive firewall rules
  - Consider operating on a hardened system reserved for that application
- Monitor
  - Use and monitor a host-based firewall
  - Consider application whitelisting

Additional topics and guidance for Supply Chain Security can be found at the Supply Chain Risk Mitigation Program page.[7]

[6] https://en.wikipedia.org/wiki/Attack_surface
[7] https://www.nerc.com/pa/comp/Pages/Supply-Chain-Risk-Mitigation-Program.aspx