Netradyne Open-Source Security Policy

v1.0

Internal and Confidential

netradyne

TABLE OF CONTENTS

Document Control

Document ID	NDOSS20231
Document Name	Netradyne Open-Source Security Policy
Document Status	Released
Document Released Date	16/FEB/2023
Document Author	Rajeev Ghosh<Rajeev.ghosh@netradyne.com>
Document Content Contributors	Sudhansu Kumar
Document Signatory	Saravanan Sankaran<saravanan.sankaran@netradyne.com>
Document Owner	Rajeev Ghosh<Rajeev.ghosh@netradyne.com>
Document Version	1.0
Information Classification	Internal

Document Edit History

Version	Date	Additions/Modifications	Prepared/Revised By
1.0	16/FEB/2023	Additions/First Version	Rajeev

Document Review/Approval

Date	Signatory Name	Organization/Signatory Title	Comments
17/FEB/2023	Saravanan Sankaran	Senior Director	

Distribution of Final Document

Name	Organization/Title
Netradyne Open-Source Security Policy	Infosec

1 Purpose

1.1 Netradyne's Free and Open-Source Security Policy

1.1.1 Introduction

The objective of this guideline is to provide security guidance to our stakeholders while choosing open-source software solutions. It will help organizations to understand and evaluate the security risk associated with using Open-Source Software and how to mitigate it.

"Source code" within the context of computer applications are the list of commands / scripts put together to create an application. It is the part of software that most computer users don't ever see. Any changes to the logic, feature sets or look and feel of the application is achieved by modifying the "program's source code". Usually this code is a guarded secret by application developers and is the developers / organization's Intellectual Property, often secured through Intellectual property Rights, Copyrights etc.

Open-source software is a software, where its developers (individuals and/or organizations) make the source code available to public so that anyone can inspect, modify, enhance it and redistribute it. It is generally distributed under special licensing schemes like Common Creatives (CC), GNU General Public License (GNU GPL) etc. Open-Source Software present several benefits as well as challenges, these are covered in additional details below.

2 Scope

Any Employee or Netradyne as an organization that is using or is considering using Open-Source software within their operations.

3 Roles and Responsibilities

Roles and responsibilities specific to this document are included below:

Role	***Responsibilities***
Owner	• Team or SME responsible for the process area needs to ensure this document is up to date and compliant with governing requirements. • Is the point of contact for the document. • Responsible for initiating and managing document review and the approval process from start to finish including gathering or delegating the collection of content including diagrams, formatting etc. as well as identifying stakeholders to participate in the peer review process.
Reviewers/Stakeholders	Representations from teams that can affect or be affected by the document under review (e.g., Operation, Security, Compliance, Quality)
Approvers	The Person(s) of authority to validate the document and sign-off on the latest version. Such Person include Document owner, Functional Team Lead, Security Lead, Product Delivery Lead.
Document Release	Document Owner/team to work with repository administrator to make release version available.

4 Procedure

4.1 Context

Open-Source is at the heart of the Netradyne business and development model. We believe that Open-Source is not a zero-sum game but that collaboration in the open, benefits all participants and creates a broader base for everybody to build upon.

Netradyne is proud to be a good citizen and leader of the global open-source community. Our approach to open source is "Open Source first, upstream first". This document provides transparency on how Netradyne operates and outlines our employee's guidelines on how to deal with open source specifically on the security aspect of OSS. It might also serve as inspiration for others implementing open-source policies. Netradyne uses the term "open source" for brevity and because it is the clearest descriptor in the context that we operate in.

We do recognize that open source is more than a development model and that there are ideals behind it; the idea is to respect communities and user's freedoms, and specifically, the freedoms to use, study, share, and improve the software. Netradyne has many engineers who passionately stand behind "free and open-source software" and supports these ideals as a company. When we discuss "open-source software", we refer to the definition of open-source licenses by the Open-Source Initiative ("OSI"). NETRADYNE considers "open source" to be software, including its documentation, released under a license that is compatible with OSI's open-source definition to be open source.

4.2 Selection Criterion

The decision of whether to choose between open source and proprietary software involve significant complexity and goes beyond mere price factors. Organizations should take an informed decision and consider amongst others, the following factors prior choosing an Open-Source Software.

4.2.1 Total Cost of Ownership (TCO):

Open-Source does not necessarily mean free. Although the license costs are zero, there may be charges for additional support that organizations may need to have, fixing bugs, trainings to develop skills within the team, premium that organizations may need to offer to hire employees with such skills etc.

4.2.2 Usability:

OSS often is not user friendly, as the developers focus on features rather than usability.

4.2.3 Skilled Resources:

Organizations may face challenges in hiring or retraining their existing staff on skill sets needed to support and develop the OSS in-house.

4.2.4 Security:

It is dangerous to assume that OSS is implicitly secure and safe to use. However, since the source code is in public domain, the code is up for review by anyone and security enthusiasts review such codes to identify vulnerabilities within the code. Multiple security researchers and organizations (Multi Eye Principle) review the code and look at it from their individual perspectives. Further, since the code is in public domain, the risks of it having Trojans is much less. Nevertheless, due to a lack of central oversight / authority that provides quality/ security assurances on the code that is distributed it is possible that malicious users may embed or corrupt the code with malwares / Trojans since the code is publicly available. Moreover, potential exploits (Proof of concepts and ready to use) may be readily available on the internet for known and published vulnerabilities in such OSS.

4.3 Some Security Considerations for using Open-Source Software

Once an organization has decided to deploy and use OSS, they should consider the following security controls to provide a layer of assurance to their business.

4.3.1 Governance:

Organizations should develop a comprehensive policy governing the usage of OSS. Amongst other things, the policy should cover an acceptable usage of OSS and the acceptable risk appetite for OSS. Risk Assessment should on a minimum cover risk related to license requirements, operations (support for the software and stability of the software) and security (vulnerabilities and known exploits).

4.3.2 Asset Inventory:

Organizations should maintain a detailed inventory of OSS used within the organization detailing instances (number / quantity) of use, version etc. Where OSS is used as a component of your in-house application, a data call should be performed to determine what components are OSS and what versions are currently is use. The inventory should include libraries, frameworks, middleware, and applications. Maintain a profile of each OSS to include the code's origin, where to get updates, and how often the community releases new versions.

4.3.3 Documentation:

Where OSS is used as a component of your in-house application, a comprehensive documentation of the components used should be maintained. This includes libraries, frameworks, middleware, and applications.

4.3.4 Source Code Repository:

Maintain a repository of the source code of OSS deployed within the organization.

4.3.5 Whitelisted OSS:

OSS should be qualified after conducting relevant usability, stability, and security tests. Only pre-approved and qualified OSS should be used and deployed within the organization.

4.3.6 Baseline Controls:

The OSS should be treated as any other information system deployed within an enterprise and should comply with the entire relevant baseline and recommended controls prescribed within the National Information Assurance Policy including its classification.

4.3.7 Risk Management:

Organizations should conduct a Risk Assessment on the OSS, based on the criticality of the services that it will deliver and implement necessary controls to manage any risks that may arise.

4.3.8 OSS Installation:

Any OSS installed / deployed should adhere to the organization's system installation procedure. This should ensure that:

- Only whitelisted OSS is deployed in the organization.
- All deployments are vetted and approved through a formal system deployment/change management process.
- All deployments are inventoried in the asset register.
- Only authorized individuals such as the system administrators should install/deploy OSS.
- Use OSS from reliable and trusted sites. Wherever possible prefer source code to binaries. It is always good to download the source code, verify against the MD5 checksums provided. Examples of trusted sites as recommended by Open-Source Initiative include freshmeat.net, sourceforge.net, osdir.com, developer.berlios.de and bioinformatics.org
- Ensure that the OSS is tested and updated with the latest patches.

4.3.9 Security Assessments:

Perform a security assessment to identify and patch any known vulnerabilities in the OSS. For critical applications, it is recommended to do a combination of an automated static analysis (source code scanning) and dynamic analysis to find vulnerabilities in individual applications. On identifying a vulnerability, the organization should:

- Check if a patch or an updated version is available to fix the identified vulnerability.
- With caution and responsible disclosure, ask the open-source community for help.
- Post the issues to the community and see if a member can help with the fix.
- Fix them yourself. Use your own or third-party development resources to resolve the issues.

4.3.10 Application Hardening:

As with any other software, the OSS should be configured in a secure manner:

- Disable unwanted services.
- In a development environment, unused and unwanted libraries, APIs, and DLL should be removed/disabled.
- User permissions should adhere to least privileges and a need to have model.
- Use a Defence in Depth strategy to include all the possible measures that need to be taken for all OSS, along with other products in the network.

4.3.11 Patch Management:

The organization's Patch Management process should monitor and update patches released for the OSS.

- Check the community associated with your open-source code. These communities typically have bug-tracking systems and mailing lists that give information on known security issues and provide the latest news and exploits.
- When a new vulnerability is identified, the organization should explore possible mitigation strategies that can be implemented until a patch is available.
- Once a patch is released, test the patch for stability and applicability within your test environment prior deploying on your production systems.
- Have a time bound approach to patch all vulnerable OSS in place.

4.3.12 Compliance:

From time to time, verify the inventory of OSS, to ensure compliance with the OSS Usage Policy. Software composition analysis (SCA) products may be used to analyse application composition to detect components known to have security and/or functionality vulnerabilities or that require proper licensing.

4.3.13 Business Continuity / Capacity Building:

Organizations should ensure that their employees have the relevant skills required to maintain an OSS. Trainings should be regularly imparted, and skills distributed to avoid single point of failures. Knowledge bases of the OSS should be developed and maintained within the organization.

5 Conduct

Compliance Checks to this process to be performed through various methods, including but not limited to reports, internal/external audits, Awareness training/assessments and feedback to the process owner. Non-compliance will be escalated to the Netradyne leadership team.

6 Exception

Exception to this procedure must be approved through the Netradyne Exception Management Process.

7 Terms/Acronyms

Term/Acronym	Definition
Commit message	A good explanation and rationale of changes to files under source code management.
Contributor	Could be anyone who comments on an issue or pull request, people who add value to the project (whether it's triaging issues, writing code or documentation, filing bugs, contributing use cases, or organizing events), or anybody with a merged pull request (perhaps the narrowest definition of a contributor).
Employee	For the purposes of this policy, the term employee includes all individuals working at all levels at all SUSE entities and affiliate businesses, whether permanent, fixed- term or temporary.
OpenSUSE Factory	openSUSE Factory is the source for openSUSE Tumbleweed and the intermediate upstream for select SUSE products.
Patch	A patch is a set of changes to a computer program or its supporting data designed to update, fix, or improve it.
Pull request	Contributions to a source code repository that uses a version control system are commonly made by means of a pull request, also known as a merge request. The contributor requests that the project maintainer pull the source code change, hence the name "pull request". The maintainer has to merge the pull request if the contribution should become part of the source base.
Upstream	Within information technology, the term upstream (and related term "downstream") refers to the flow of data. An upstream in open source is the source repository and project where contributions happen and releases are made. The contributions flow from upstream to downstream. When talking about an upstream, it's usually the precursor to other projects and products. One of the best-known examples is the Linux kernel, which is an upstream project for many Linux distributions.
Upstream First	Patches are contributed upstream and included downstream only if they become part of the upstream.
OSS	Open-Source Software
SCA	Software Composition Analysis
API	application programming interface
DLL	Dynamic Linked Libraries

References

7.1 Templates
<List of (or Links to) associated templates>
7.2 Policies
<List of (or Links to) associated corporate level policies>
7.3 Process/Procedures
Netradyne Vulnerability & Patch Management Procedure_v1.0.pdf

7.4 Standards
<List of (or Links to) related Netradyne Standards>
7.5 Miscellaneous
<List of (or Links to) any relevant documentation not covered in the list above>

8 Appendix: OSS Licensing

Open-source licenses are generally classified as permissive, reciprocal (restrictive), or falling somewhere in between, depending on the nature of the restrictions and obligations contained in the license.

8.1 Permissive open-source licenses:

These are generally characterized by more lenient obligations, which may include allowing secondary distribution so long as the distributor provides attribution or allowing secondary software alterations or improvements to remain proprietary. This flexibility facilitates the ability of end users or licensors to incorporate the licensed software into derivative products. Example: The MIT license, the Apache Software License, and the Berkeley Software Distribution license (BSD)

8.2 Reciprocal or restrictive open-source licenses:

These are generally characterized by more stringent restrictions and obligations for end users or licensors of open-source software. The distinguishing feature of a reciprocal open-source license is the so called "copyleft" requirement. The copyleft requirement obligates those who incorporate and later distribute code to do so under same terms and conditions as the original piece of open-source software, even if that software has been modified.

Additionally, open-source licenses containing copyleft obligations will frequently require licensors to reveal the underlying source code or make it available when requested by others. These copyleft obligations can create risky exposure for licensee's failing to monitor the terms and conditions governing the open-source components being incorporated into such licensee's software code. Example: The GPL

9 Appendix A: Document RACI Matrix

Role/Activity	Document Owner/Functional Area Lead	Document Contributor	ND Leadership	Functional Area Team	InfoSec	All ND Member(s)
Ensure document is kept current	A	R	I, C	R, C	C	I
Ensure stakeholders are kept informed	A	R	-	R	C	-
Ensure document contains all relevant information	A	R	I, C	R, C	C	I
Ensure document adheres to document governance policy	A, R	R	I	R, C	R, C	I
Provide SME advice	I, R	A, R	I	R, C	I, C	I
Gathering and adding document contents	I	A, R	I, C	R, C	C	I
Document Approval	A	R	I, R	I	I, R	I

Key	
R	Responsible
A	Accountable
C	Consulted
I	Informed