Applying Cybersecurity Regulations and Industry Standards to Open Source Projects

Luci Stanescu

8 April 2025

Any fans of regulations?

Two perspectives

Regulated environments using OSS projects

Regulated enterprises need to assess their software supply chain

Available standards are hard to fit the complex OSS environments

OSS projects improving their security posture

There are great existing initiatives looking to improve OSS security posture (e.g. OpenSSF Best Practices, Scorecard, LF report on EU CRA)

Security regulations and standards are based on general best practices

What we’re evaluating

NCSC Vendor Security Assessment
CISA Vendor Supply Chain Risk Management Template

OpenSSF Best Practices
OpenSSF Scorecard

Do regulatory tools apply to OSS?

	NCSC VSA	OpenSSF Best Practices
Version control	V.A.3: Each product has a version-controlled code repository	Passing: The project MUST have a version-controlled source repository that is publicly readable and has a URL.
Issue tracking & remediation	V.J.1: The vendor has a process for issuing remediation. (refers to vulnerabilities)	Silver: The project MUST have a documented process for responding to vulnerability reports.

Regulatory tools

NCSC VSA

Created in support of TSA and ECR

“advice on how to assess the security of network equipment”

10 categories, 58 criteria items

Specific / prescriptive: V.D.1 ”The vendor makes use of modern heap protection mitigations”

CISA Vendor SCRM Template

“a standardized template of questions as a means to communicate ICT supply chain risk posture in a consistent way“

8 categories, we only focus on one: Secure Design and Engineering (category 3)

More generic: 3.13 “Does your organization configure the compilation and build processes to improve executable security?“

Open source tools

OpenSSF Best Practices

A program to promote security best practices in FLOSS projects

143 criteria over three tiers (badge levels)

Self-certification

OpenSSF Scorecard

20 automated checks that assess projects

A stated goal is to enable informed decisions about dependencies

Also based on security best practices

Regulated environments using OSS projects

Pain points

The problem

Can't ask OSS projects to fill in NCSC VSA or CISA Vendor SCRM Template.

Inconsistently addressing supply chain risk is problematic.

The solution

Step 1

Map regulatory standard to OSS tools

Step 2

Gain blessing for the mapping

Step 3

Record compliance data for used projects

Is a mapping possible?

32 / 58 NCSC VSA criteria items can be mapped to OpenSSF Best Practices criteria (8 automatable to OpenSSF Scorecard tests)

13 / 16 CISA Vendor SCRM items can be mapped to OpenSSF Best Practices criteria (4 automatable to OpenSSF Scorecard tests)

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.A.2: Software maintenance – This maintenance, as a minimum, covers security fixes for the product.	Passing: There MUST be no unpatched vulnerabilities of medium or higher severity that have been publicly known for more than 60 days.	Vulnerabilties
V.A.3: Each product has a version-controlled code repository	Passing: The project MUST have a version-controlled source repository that is publicly readable and has a URL.	Implied

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.A.4: Software releases – Each product goes through a rigorous software release cycle including internal testing [...].	Passing: The project MUST use at least one automated test suite [...]	CI-Tests
V.A.7: Use of tools, software and libraries – Third party tools (e.g. code compilers) software components and software libraries [...] are inventoried.	Silver: The project MUST list external dependencies in a computer-processable way.	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.A.8: Software documentation – The vendor provides up-to-date and technically accurate documentation alongside new releases of the product.	The project MUST provide, in each release, release notes that are a human-readable summary of major changes in that release [...].	N/A
V.B.1: Security culture – The vendor has a security culture which ensures that security principles are followed. / V.B.2: Secure Development Lifecycle	Silver: The project MUST implement secure design principles (from "know_secure_design"), where applicable.	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.B.3: Internal component management – Any shared internal components or libraries are kept up to date and only the latest stable, supported version is used. / V.B.4: External component management – Only supported external components are used within a product.	Silver: Projects MUST monitor or periodically check their external dependencies [...]. The project MUST [...] make it easy to identify and update reused externally-maintained components.	Dependency-Update -Tool

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.C.3: Build environments and automation – Build environments are simple, and the build process is automated.	Passing: […] the project MUST provide a working build system that can automatically rebuild the software from source code.	Vulnerabilties
V.C.4: Role-based access – Only individuals with a need have access to the internal code base.	Silver: The project MUST clearly define and publicly document the key roles in the project and their responsibilities […].	Branch-Protection

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.C.5: Code review – All code is independently reviewed prior to acceptance.	Gold: (not quite) The project MUST have at least 50% of all proposed modifications reviewed before release by a person other than the author.	Branch-Protection, Code-Review
V.C.6: Repeatable builds – All builds of released software can be replicated at a future date.	Gold: The project MUST have a reproducible build.	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.D.1: Heap protections / V.D.2: Stack protections / V.D.3: Data execution prevention / V.D.4: ASLR / V.D.5: Memory mapping protections	Gold: Hardening mechanisms MUST be used in the software produced by the project so that software defects are less likely to result in security vulnerabilities.	N/A
V.E.1: Software and firmware signing – Vendor's software and firmware is digitally signed.	Silver: The project MUST cryptographically sign releases of the project results intended for widespread use [...].	Signed-Releases

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.E.3: Secure update – Updates are delivered via a secure channel.	Passing: The project MUST use a delivery mechanism that counters MITM attacks.	N/A
V.G.1: Automated testing – Once developed, extensive security tests are automatically run.	Passing: The project MUST use at least one automated test suite [...]. It is SUGGESTED that the test suite cover most (or ideally all) the code branches, input fields, and functionality.	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.G.5: Fuzzing	Silver: (memory-unsafe languages) [...] at least one dynamic tool (e.g., a fuzzer or web application scanner) MUST be routinely used.	Fuzzing
V.G.6: External testing	Gold: The project MUST have performed a security review within the last 5 years.	N/A
V.G.7: Dynamic application security testing	Passing: It is SUGGESTED that at least one dynamic analysis tool be applied [...].	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.H.2: Protocol Standardisation – The product can be configured to only use standardised protocols.	Passing: The software produced by the project MUST use, by default, only cryptographic protocols and algorithms that are publicly published and reviewed by experts [...] + others.	N/A
V.H.3: Management plane security – By default, the product is configured to only use up-to-date, secure protocols on the management plane. / V.H.5: No unencrypted protocols	Gold: The software produced by the project MUST support secure protocols for all of its network communications [...].	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.H.9: Good Practice Guidance – The vendor is explicit about the threats to the equipment that they have sought to mitigate, and those they have not.	Silver: The project MUST provide an assurance case [...]. The assurance case MUST include: a description of the threat model [...].	N/A
V.J.1: The vendor has a process for issuing remediation. (refers to vulnerabilities)	Passing: The project MUST have a documented process for responding to vulnerability reports.	N/A

Proposed mapping

NCSC VSA	OpenSSF Best Practices	OpenSSF Scorecard
V.J.3: Vulnerability reporting – publicly advertised route for disclosure of security issues.	Passing: The project MUST publish the process for reporting vulnerabilities on the project site.	Security-Policy

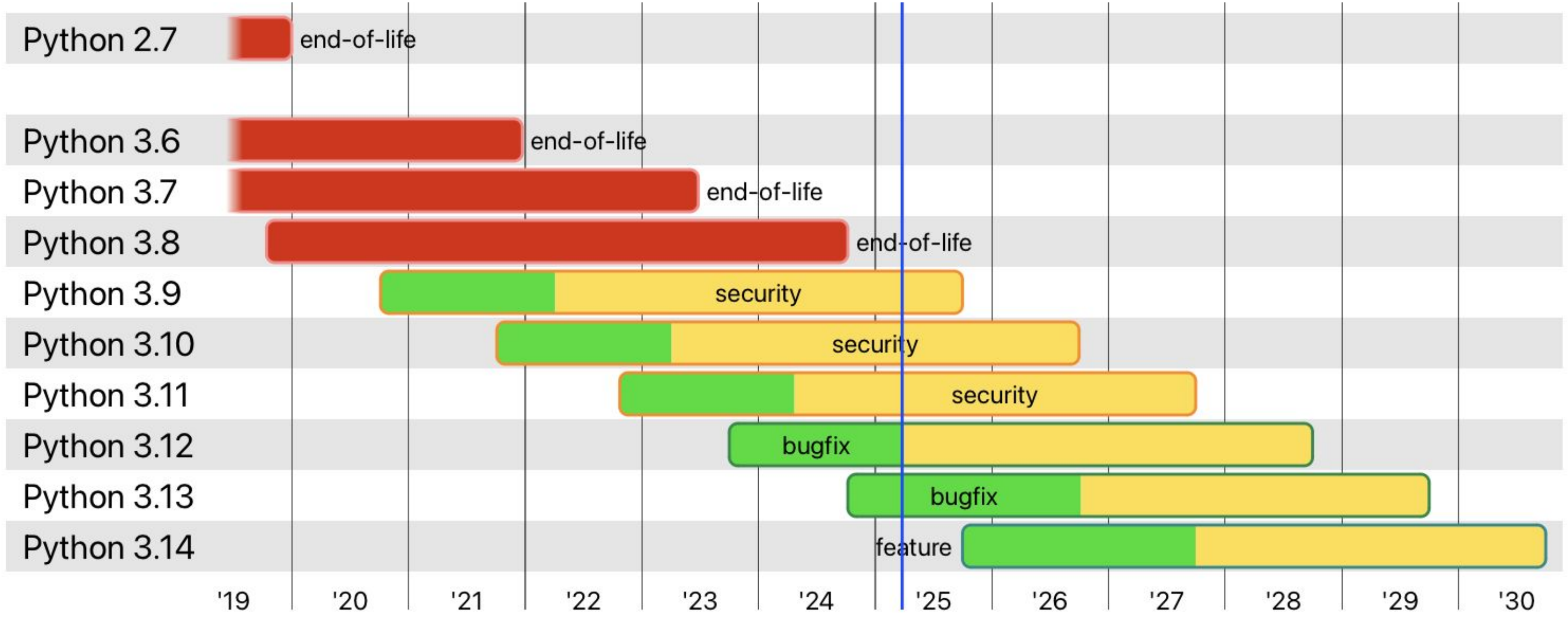
What can be further learned

Exemplified using NCSC VSA

Security declaration (VSA criteria)

- V.A.1: Product lifecycle process – The vendor clearly identifies the lifecycle for each product.
- V.A.5: Development processes and feature development – There is one primary release train of the product.
- V.A.6: International release and forking – The vendor maintains a single, global version line for each product. There are a minimal number of other versions (ideally none).
- V.D.7: Security improvement and secure execution environments – The vendor has plans to continue to improve its product's security.

Python lifecycle



https://devguide.python.org/versions/

Security declaration (the why)

- Communicates expectations to end users
- Emphasises importance of security to contributors
- OpenSSF BP Passing:
  - The project MUST have at least one primary developer who knows how to design secure software.
  - At least one of the project's primary developers MUST know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them.

Code quality (VSA criteria)

- V.B.5: Unsafe Functions – There are no unsafe functions used within the vendor’s released code.
- V.B.6: Redundant and duplicate code – The vendor’s source tree is maintained to a level that there is limited redundant or duplicate code.
- V.B.7: File structure – The vendor’s source tree is maintained to a level where code complexity is minimised, and functions perform single, clear actions.
- V.B.9: Comments – The source tree has suitable and understandable comments through it.

Code quality (the why)

- Code complexity seems to correlate with number of vulnerabilities [1]
- Code quality can encourage contributions
- Unsafe functions are considered unsafe for a reason

[1] I. Chowdhury and M. Zulkernine, “Can complexity, coupling, and cohesion metrics be used as early indicators of vulnerabilities?”, Proceedings of the 2010 ACM Symposium on Applied Computing, 2010

What are unsafe functions

```
luci@dev-noble:~/sandbox$ cat dangerous.py
import sys
eval(sys.argv[1])
luci@dev-noble:~/sandbox$ bandit dangerous.py
[...]
Run started:2025-04-03 11:47:14.521892
Test results:
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer
ast.literal_eval.
   Severity: Medium   Confidence: High
   Location: dangerous.py:2
   More Info:
https://bandit.readthedocs.io/en/latest/blacklists/blacklist_calls.html#b307-eval
1     import sys
2     eval(sys.argv[1])
--------------------------------------------------
```

Restricted environment (VSA criteria)

- V.C.1: Segregation of development environment – Development environment is segregated from corporate network and protected from the internet.
- V.G.2: Testing rigour – Developers cannot modify the build environment to hide or disregard build issues.

Restricted environment (the why and how)

- Least privilege principle is good practice
- Supply chain attacks are on the rise
- A project could be:
  - affected through its supply chain (e.g. GH actions)
  - targeted directly (as part of a supply chain attack)
- Either way, focus should be on protecting code integrity

Security best practices (VSA criteria)

- V.D.6: Least Privilege code – The vendor follows a 'least privilege' methodology when developing and executing code within their products.
- V.H.6: No undocumented administrative mechanisms – hard coded passwords, access key pairs.
- V.H.7: No undocumented administrative features.
- V.H.8: No default credentials – No default passwords are left on the device after the initial setup.
- V.H.1: Product hardening – The product can be easily hardened into a secure configuration.

Security best practices (the why)

- Secure-by-default is important because, unfortunately, configurations are rarely hardened.
- Secure-by-default also harmonizes configuration hardening.
- Important further hardening configuration steps should be emphasized in documentation.

Testing (VSA criteria)

- V.G.3: Security Testing – Security functionality is tested to demonstrate correct operation.
- V.G.4: Negative testing – Extensive negative testing is performed against every product release, including a wide range of potential failure cases, inappropriate message sequencing and malformed messages.

Testing (the why)

- Explicitly focusing on security controls/checks ensures these continue to function as intended.
- Testing success paths and regressions is common, but leads to blind spots (triggering error conditions is not that difficult and is what threat actors, not normal users, will do).

Vulnerability response (VSA criteria)

- V.J.2: Issue comprehension – For issues, the vendor identifies the root cause analysis.
- V.J.4: Issue transparency – The vendor is transparent about their patching of security issues.
- V.J.5 Product Security Incident Response Team (PSIRT) – The vendor has set up the PSIRT structures within its organisation.

Vulnerability response (the why)

- Vulnerabilities happen: being prepared ensures these get handled coherently and without panic.
- Ignoring the root cause would just result in exploitability soon after.
- Open source promotes transparency – allows users to make informed, risk-based decisions.

Does it sound familiar?

- 3.12. Does your organization reuse existing, well-secured software and hardware components, when feasible, instead of duplicating functionality?
- 3.16. Does your organization maintain and manage a Product Security Incident Reporting and Response program (PSIRT)?
- 3.17. Does your organization analyze vulnerabilities to identify root cause?

CISA Vendor SCRM Template Highlights

- 3.4. Does your organization document and communicate security control requirements for your hardware, software, or solution offering?
- 3.11. Does your organization verify that third-party software provides required security requirements/controls?
- 3.15. Does your organization configure offerings to implement secure settings by default?

Takeaways

- Regulations can be adapted to open source projects.
- Use of open source in strongly-regulated environments can be encouraged by the adoption of general-purpose security best practices.
- Open source projects can improve their security posture by looking at government and industry cybersecurity standards.

Thank you! Questions?