REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
06/02/2021	Conference Paper	

4. TITLE AND SUBTITLE
Enhancing Software Supply Chain Security Workshop Position Paper: On minimum requirements for testing software source code

5a. CONTRACT NUMBER
FA8702-19-C-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Andrew A Lilley Brinker; Andrew Buttner

5d. PROJECT NUMBER
03211CY0CA

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730

8. PERFORMING ORGANIZATION REPORT NUMBER
PRS-21-1560

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
United States Air Force (AFLCMC)

10. SPONSOR/MONITOR'S ACRONYM(S)
AFLCMC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
Submitted to NIST Enhancing Software Supply Chain Security Workshop, June 2-3, 2021.

14. ABSTRACT
Analyzing practices and risks, particularly by looking at the wealth of metadata associated with software repositories, is a scalable and powerful approach to assessing software supply chain security. This approach addresses two key questions for software supply chains:
1. Are there identifiable supply chain attacks present in this repository currently?”
2. Does this repository follow good software development practices which reduce the risk of vulnerabilities or supply chain attacks in the future?”

15. SUBJECT TERMS
Software Engineering (General); Information Security (General); Hipcheck; supply chain security; metadata

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE		3	Susan Carpenito
					19b. TELEPHONE NUMBER *(Include area code)* 781-271-7646

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

Andrew Lilley Brinker
Lead Cybersecurity Engineer
The MITRE Corporation
Phone: 571-425-5875
Email: alilleybrinker@mitre.org

Enhancing Software Supply Chain Security Workshop Position Paper:
On minimum requirements for testing software source code (Item 4)

Traditional approaches to testing software source code, whether through static or dynamic code analysis, suffer from limitations in scalability. Static code analyses are conservative and inherently produce false positives which must be assessed or require sampling approaches which may miss true vulnerabilities. Dynamic code analyses require individualized setup and tuning to produce usable results. In either case, these approaches require substantial labor to establish, operate, and triage results.

In software, there is no distinction between the code produced by a vendor and the code which they incorporate from third parties. Open-source dependencies, direct or transitive, all execute at the same level of privilege. All code must be treated with the same level of rigor in its analysis and with the same degree of suspicion as to the threat it may pose to the system or network on which it is hosted.

To complicate matters, there may exist many individual dependencies needing to be assessed. Given concerns about the security of both development and operational environments, the security of development dependencies not present in the final delivered source code must also be considered. Without this, the compromise of development environments through malicious or risky dependencies may lead to operational compromise as well.

Attempting to apply static or dynamic code analysis at scale to these dependencies is more than can be reasonably managed. While the scaling challenges may be reduced by greater agreement between analyzers (see the Software Assurance Results Interchange Format [SARIF] effort as an example) or by aggregation of reusable results in a central information clearinghouse, there is a better approach which more directly addresses risk and sidesteps the challenging requirements of human labor inherent to code analysis.

Analyzing practices and risks, particularly by looking at the wealth of metadata associated with software repositories, is a scalable and powerful approach to assessing software supply chain security. This approach addresses two key questions for software supply chains:

1. Are there identifiable supply chain attacks present in this repository currently?”
2. Does this repository follow good software development practices which reduce the risk of vulnerabilities or supply chain attacks in the future?”

MITRE has developed Hipcheck, a tool for analyzing software repositories to produce answers to these questions. Hipcheck looks at contribution history to understand who has contributed to a software project, what they have contributed, and to what degree those contributors can be trusted. It looks at information on other dependencies a project incorporates to identify possible attacks in those dependencies, and it looks at metadata around contribution acceptance to identify whether code is reviewed by someone other than the author prior to inclusion in a project. These analyses drive directly at questions of both identifiable attacks and poor coding practices.

Supply chain attacks against open source software include “typosquatting,” where an attacker publishes a malicious but similarly named version of a legitimate package; and “malicious contribution,” where changes which erode security pass through a project’s normal change management process. Hipcheck’s attack analyses currently include detection of potential typosquatting attacks and potential malicious contributions.

Software development risk can arise from a lack of code review as well as the presence of unquestionably poor coding practices, such as the persistent and unjustified use of unsafe functionality for which there exists safe alternatives (e.g., the `strcpy` function in C, or `eval` function in JavaScript). Hipcheck employs statistical methods to surface unusually large or textually unusual changes in a project’s history and incorporates knowledge on the level of trust in the contributor of those changes. It also checks for use of unsafe functionality which may indicate a lack of quality control on the software.

Scrutinizing a project’s metadata to understand risk can function as a gating mechanism for a vendor’s decision to include certain third-party open-source code. Evidence of this diligence being performed could be required as part of an acquisition process. These analyses can also be applied to first party software being acquired to help characterize their own practices in managing change and the incorporation of third-party code.

In Hipcheck, the individual analyses are filtered through a set of user-configurable thresholds with sensible, initial defaults based on real-world software repository analysis. Thresholds are weighted according to their importance to produce a final “risk score,” against which an overall risk tolerance can be compared to produce a “pass” or “investigate” recommendation. This workflow can be incorporated into automated DevSecOps processes to limit the need to manually assess incoming software and to help guide such a review by identifying potential risks.

It is our perspective that Hipcheck’s software metadata analysis is an effective way to answer questions of risk in first- and third-party software for federal acquisition in a scalable manner necessary to meet the challenges of modern software development and acquisition. Hipcheck provides greater visibility into software supply chain security and creates actionable paths to assess and manage it at scale.