Delta Security Certification for Software Supply Chains

Ákos Milánkovich | Security Evaluation Analysis and Research Laboratory
Katja Tuma | Vrije Universiteit

We developed and evaluated an industry-level tool to support practitioners in analyzing and visualizing delta-certification to aid small companies specializing in security evaluation of the software supply chain.

The continuous growth of open source software using a combination of cloud and local storage has resulted in complex software supply chains that, if left unchecked, can lead to unprecedented security breaches. Indeed, the contagions publicly exposed vulnerabilities, such as Colonial Pipeline,[1] Codecov,[2] and Solarwinds,[3] which equate to a storm of potential threats expanding in scope, complexity, and impact. Gartner predicts that by 2025, 45% of organizations worldwide will have experienced attacks on their software supply chains, a threefold increase from 2021.[4] On the bright side, organizations with software bill of materials (SBOMs) (i.e., record containing the details and supply chain relationships of various components) could identify the vulnerable component faster, compared to spending weeks to find vulnerabilities.[5,6]

Introduction

In domains where security certification is required (such as safety-critical systems[7]), keeping track of security issues for software updates, including internal and external dependencies, is not trivial.[8] Certifying software requires frequent recertification of components, their propagating effects on legacy components, and their compositions into systems that are part of the software supply chain. In addition, rapid collaboration, continuous integration (CI), and continuous deployment (CD) are dramatically changing the speed. Still, propriety software distributors are liable on the grounds of partner agreements to deliver a (re)certified product.

The situation is even worse for small and medium-sized enterprises, which often consume open source software. In principle, the liability rests with the organization that placed the open source software into a product and placed that product on the market. When an organization uses open source software, it takes on responsibility to (re)certify that component and verify that no security vulnerability is propagated up the software supply chain. To this aim, organizations could adopt static application security testing (SAST) products. But, SAST solutions produce high numbers of false positives, which means further resources must be spent on sieving through security alerts.

Due to the limited use of certification in the information and communications sector, the European regulation (Cybersecurity Act) on cybersecurity mandated consistent software certification where the certification is required. To this aim, the European regulation tasked the European Union Agency for Cybersecurity

Digital Object Identifier 10.1109/MSEC.2023.3311464
Date of current version: 20 September 2023

(ENISA) to create holistic certification schemes to increase trust in digital solutions. Traditionally, software certification requires analysts to pore over the software as a whole, which can be a time-consuming process. This costs of generating the evidence might still be acceptable as one-off expense if the software is frozen after production or is slowly evolving over time. But in the era of CI/CD, such certification processes could cripple the software development industry. In fact, the Cybersecurity Act emphasizes that "(72) flexible cybersecurity solutions are necessary for the industry to stay ahead of cyber threats, and therefore any certification scheme should be designed in a way that avoids the risk of being outdated quickly" (https://eur-lex.europa.eu/eli/reg/2019/881/oj). Focusing the recertification on the software changes and their effects could help shorten the recertification process. Similar to SBOMs, the evidence collected for security *delta-certification* (i.e., analyzing the introduced changes in a new project version) could help analysts to quickly check whether currently used software packages are vulnerable to known vulnerabilities. But, there is no existing work investigating recertification and how to integrate it into the development process of open source software.

To address this gap, this work investigates how security certification scales when software is continuously updated by new commits. From the perspective of the practitioner, we focus our investigation on two specific research questions (RQ):

- RQ1: What are the key performance indicators for measuring industrial relevance of automating delta-assessment?
- RQ2: To what extent can the effort of the human analyst be reduced by automating part of delta-assessment?

The Security Evaluation and Research Laboratory (SEARCH-Lab or SLAB) developed an industry-level tool, based on lightweight static analysis, to explore the feasibility of delta-certification for multiparty open software and services (MOSS) of the supply chain. This approach can help shorten the review time by at least 30% (as examined in the "Evaluation" section) and continuously identify vulnerabilities while enforcing policies to improve precision.

Beyond the State-of-the-Art

Table 1 shows existing certification schemes and their suitability for certifying rapidly changing open source software. Industry-standard security certification schemes, like Common Criteria (CC),[9] Common Criteria based European Candidate Cybersecurity Certification (EUCC),[10] and European Cybersecurity Certification Scheme on Cloud Services (EUCS)[11] heavily rely on development process auditing and documentation-based evaluation, which are challenging to automate. Santos et al.[12] proposed a metrics framework suitable for supply chains and in the industrial context to promote the level of trust between the nodes of a supply chain by using the same metrics and goals related to certification. However, to the best of our knowledge, no existing approach provides a possible solution for integrating certification into the CI/CD process that is also compatible with automation.

Delta-certification is a novel certification approach that focuses on the changes between the certified and new versions of the target.[13] By emphasizing the deltas, analysts can save time and effort by delving deeper into the changes rather than performing a complete evaluation again. This approach requires appropriate automation tools to detect and present changes to the human analyst.

Table 1. Existing certification schemes and their support for automated evaluation.

Scheme	Document Based	Effort	Static Analysis	Delta Evaluation	Full Name for Reference
CC	Yes	High	Partly	No	Common criteria (ISO/IEC 15408)
EUCC/EUCS	Yes	High	Yes	No	EUCC scheme, EUCS, cloud services scheme (ENISA)
CSPN	Partly	Medium	Partly	No	Certification de Sécurité de Premier Niveau (Agence Nationale de la Sécurité des Systèmes d'Information)
MASVS	No	Medium	Optional	No	Mobile application security verification standard (Open Worldwide Application Security Project)
CLS	Partly	Medium	Yes	No	Cybersecurity labeling scheme (Cyber Security Agency of Singapore)
SCL	No	Medium	Yes	No	VESSEDIA approach for security evaluation (H2020)
This work	No	Low	Yes	Yes	Incremental and continuous certification scheme (SLAB)

Technical Challenges

We outline three major technical challenges of automating the certification for MOSS of the supply chain.

Finding Evidence Sources at Scale

The software supply chain is composed of large projects with internal and transient dependencies, which means that the security evaluation would require an investigation of thousands of files. The number of file changes at individual commits or builds is usually significantly lower. But precisely identifying only security-relevant files is not trivial. For example, in the course of conducting the case study (see the "Evaluation" section), we discovered that logging was implemented using a particular library. We defined a *filter rule* to list the instances the logger was called and the *assessment rule* can decide whether it contains sensitive data. Should a later commit include a change in the logging library used, the filter rule would need to be updated to detect the new implementation. Currently, the rule update needs to be performed by the user. While potentially this could be automated using source code tracking, it would not be trivial, even for simpler rule updates. Finding evidence sources is especially challenging in situations of absence of security control, since security controls can be implemented in different ways and at different locations in the code base. The first technical challenge we face is to quickly gather the "prime suspects" in the set of available files for a deeper analysis.

"There is no existing work investigating recertification and how to integrate it into the development process of open source software."

The Level of Human Involvement

Our approach is based on static code analysis, which has often been characterized by high false-positive rates. It is well-known that striking a balance between false positives (alerts of potentially vulnerable code that is actually safe) and false negatives (overlooked vulnerabilities) is challenging in practice. The level of human involvement and its role in striking the balance is not trivial to determine. The level of human involvement in code-review has been investigated at Facebook, where static analyzers have been used as bot participants in code review, making automatic comments whenever an engineer submits a code (so-called *diff time* deployment).[14] The alternative deployment model (dubbed as traditional *offline* deployment) was also used internally. In the case of offline deployment, the issues are presented to engineers outside of their workflow (e.g., a security expert opening an issues for the developer to solve). A striking observation is that the diff time deployment saw a 70% fix rate, where a more traditional offline deployment saw a 0% fix rate. Bringing the security analyst in the loop (without introducing too much manual work) might be the key to effectively automate the security evaluation of the software supply chain. But, it is not yet clear which part of the security evaluation of the supply chain should still be handled by the analyst and what analyses can be safely automated. Therefore, the second technical challenge we face is to determine to what extent (and where) the analyst needs to be involved so that the assessments are precise, quickly obtained, and reliable so they can be used in a real industrial context.

Finding Evidence for Concrete Security Requirements

Once identified, the list of potential evidence files has to be scrutinized to identify whether a particular change in the code base has resulted in failing or passing concrete security requirements. For some requirements (e.g., DATA.2, no sensitive data should be logged) finding the lines of code where the logging library is called is already a good starting point. But other requirements require gathering evidence not only from source code, but also configurations files or outputs of running monitoring tools. Minor modification in the code could cause a cascading effect, where several types of files have to be modified for the project to satisfy the requirement again. For instance, introducing rate limiting to prevent denial of service attacks using the Zuul library is useless without appropriately configuring the proxy in the configuration files. Many of these propagating changes are very dependent on particular language features and runtime environment frameworks, which makes generalizing the automated solutions difficult. The third technical challenge is to automate the assessment of the impact of the code modifications on fulfillment of concrete security requirements.

Certification With DeltAICert

In this section we outline the delta-certification scheme and the main components of the tool.

The Scheme

The delta-certification workflow is shown in Figure 1. The scheme consists of a mandatory baseline certification, which establishes the minimum requirements for the certification process. The baseline certification is conducted by security analysts who use tools to generate technical evidence for subsequent delta-certifications. The filtering rules ("Listings 1 to 5") can be used to this purpose. These can be further extended to the use of other SAST tools.

In the delta-certification, the human analyst leverages information gained from running third-party tools [e.g., SonarQube (www.sonarsource.com)] to define the rules to extract evidence. The extracted evidence is then assessed based on established requirements and recorded in the certification report. Certification is granted if all requirements are met. Delta-certifications are typically conducted when a new version of the target of evaluation (ToE) is released after a baseline certification. However, if the certification's validity period has expired or an incident occurs, a new baseline certification is required to earn the certificate. For this approach to be practicable, we suggest the use of methods and tools to provide continuous certification from commit to commit instead of releases. In this way, the automation can handle most of the cases.

DeltAICert Tool

The purpose of the DeltAICert tool is to help security analysts focus only on changes that may have an impact on previous certifications rather than starting the certification process from scratch. The tool was developed at SLAB, a company with experience in conducting security assessments of software and services in the software supply chain. The company has previously developed an internal methodology (MEFORMA[15]) to assess the ToE. First, the analysts gathers the information, iteratively scopes the analysis, and conducts threat modeling and risk analysis. DeltAICert automates the information gathering and for delta-certification, the comparison steps. The tool also supports the collection of evidence during security assessments and provides scripting abilities for the automatic extraction of useful information from evidence.

To certify parts of the software supply chain, the delta-certification scheme requires (read-only) access to the source code repository, to be triggered by new versions of the ToE and check out the project for analyzing the source code and configuration files. Next, DeltAICert will collect and analyze the code base for evidence. The tool analyzes various type of evidence files, including source code snippets, (parts of) runtime logs, network captures, results of static or dynamic analysis tools, etc. Some of the outcomes can be used directly as evidence [e.g., common vulnerabilities and exposures (CVE) scores for identified vulnerabilities], while source code snippets may still require the analyst to read and understand what has been modified.

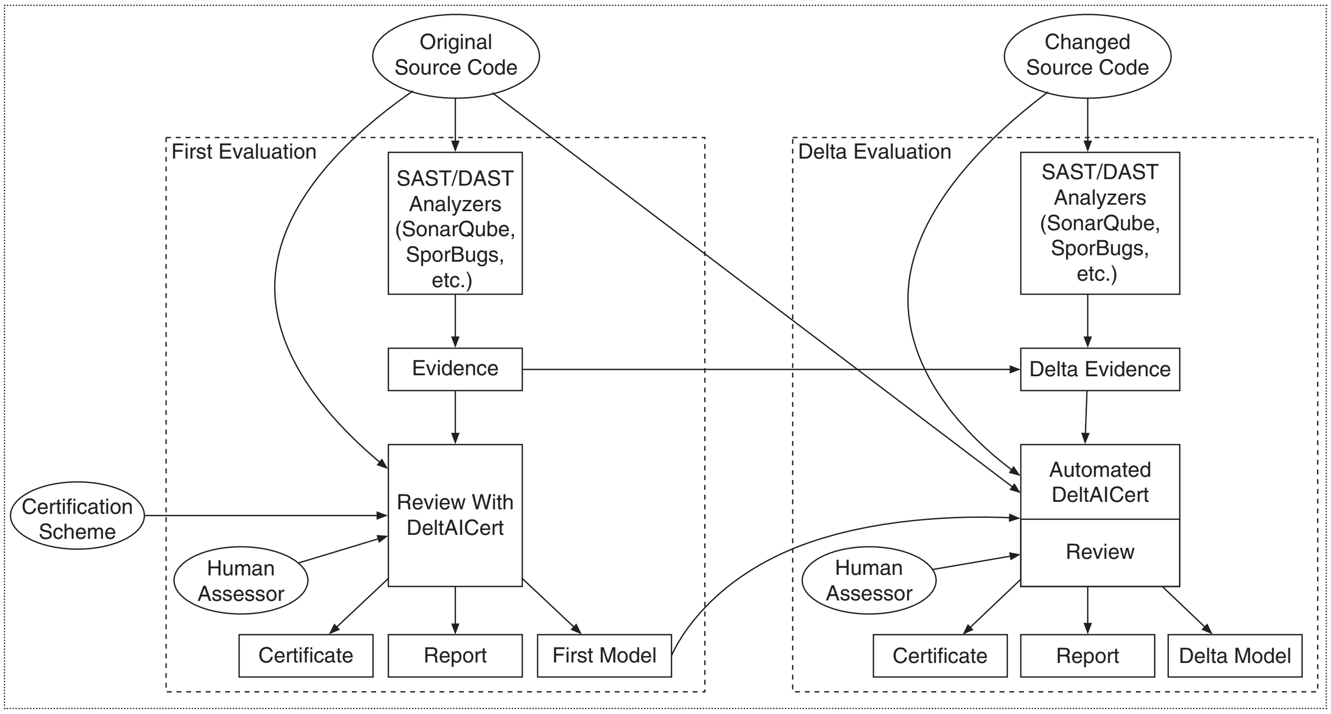

Figure 1. Delta-certification workflow.

DeltAICert also provides the evaluators with the automation capability of analyzing dependencies among the software packages. Namely, it provides the framework to achieve these goals in a unified way. Checking dependencies among the software packages is implemented by using existing tools [e.g., we use the QualityGate (https://www.quality-gate.com/webpage)]. We process their outputs comparing the current state to their certified state so that we know whether any of those changes affect the requirements and the security of the target.

The results of this automated step are presented to the analysts for a final review and for a final decision on the assessment. (While the rules presented in this work are based on simple heuristics, the approach can be extended in the future by running SAST tools to help collect the evidence.) The role of the analyst is to leverage the provided evidence (i.e., create explicit trace links to the evidence files) and make the final assessment for each security requirement.

Listing 1: Finding logging evidence.

```
import  os
import  sys
def  checkMapping(fname):
     fl = open(fname, 'rt').readlines()
        for i in range(len(fl)):
            for check in ['log.']:
            if (check in fl[i]):
              print ('%s:%d'%(fname, i))
              print ('␣␣line:␣%s'%fl[i])
for root, subdirs, files in os.walk
(file_name):
    for filename in files:
      if filename.endswith(".java"):
        path = os.path.join(root, filename)
         checkMapping(path)
```

Listing 2: Finding end points.

```
def  checkMapping(fname):
     fl = open(fname, 'rt').readlines()
     for i in range(len(fl)):
     for check in ['@RequestMapping',
         '@GetMapping', '@PostMapping',
         '@PutMapping', '@DeleteMapping',
         '@PatchMapping']:
            if (check in fl[i]):
              print ('%s:%d'%(fname, i))
for root, subdirs, files in os.walk
(dir_name):
    for filename in files:
      if filename.endswith(".java"):
         path = os.path.join(root, filename)
         checkMapping(path)
```

Evaluation

In this section, we report our evaluation with a case study, an evaluation with industry experts, and the experiment with 69 students. We end this section with a brief discussion of the threats to validity.

Case Study

ToE. We evaluated the DeltAICert tool with PiggyMetrics (https://github.com/sqshq/piggymetrics), an open source financial advisor app implementing a microservice architecture. The choice of the project was motivated by the need to evaluate the tool with a medium-size project (7,000 lines of code) with enough complexity to be challenging from the perspective of evaluating its security posture. The PiggyMetrics project captures well the MOSS concept that the new certification scheme at SLAB aims to certify, and is amenable to security analysts covering the manual evaluation due to the good balance of code quantity and number of technologies and features (including dependencies of the software supply chain in the open source). In addition, the projects' infrastructure automation serves as a representative example of the continuous integration workflows typically used in the software supply chain.

Overall, we have identified 35 security requirements, including requirements related to securing the architecture, data flows, and use of cryptographic primitives, authentication, communication, and environment. For the purpose of the baseline assessment, we focus on a subset of the identified requirements.

Baseline assessment. To support the human analyst in conducting the baseline assessment, DeltAICert implements filtering rules. Below, we show some examples of

Listing 3: Finding external components

```
for o in piggy_metrics_services.elements:
    for t in
    o.class_object_class.stereotype_
    instances:
        if (str(t)=='External␣Component'):
            for l in o.linked:
            print (str(l) + '␣- > ␣%s'%(o))
```

the python scripts ("Listings 1 to 5") used to go through all of the files: e.g., log files, abstract syntax tree, source code, list of third-party libraries, etc. For example, the list of Java files and locations with occurrences of the "log" keyword ("Listing 1").

Listing 4: Finding HTTP and HTTPS calls.

```
from piggy_metrics_v1 import *
for o in piggy_metrics_services.elements:
    for l in o.links:
        for t in l.stereotype_instances:
            if (str(t) == 'HTTP' or
                str(t) == 'HTTPS'):
                print ('%s␣- > ␣%s,␣%s'%
                  (str(o),str(l), str(t)))
```

The authors analyzed all of the requirements using these filtering rules. The result of the baseline assessment revealed eight failing security requirements,

Listing 5: Finding vulnerabilities in dependencies.

```
b = open(sys.argv[1], 'rt').read()
deps = json.loads(b)

for d in deps['dependencies']:
    if ('vulnerabilities' in d):
        if ('packages' in d):
            for v in d['vulnerabilities']:
                print ('%s␣-␣%s'%(d
                ['filename'],
                   v['name']))
```

Table 2. Summary of baseline and delta-assessments.

ID	Requirement	Baseline	Assessment Rationale	Change	Delta
DATA.1	No sensitive data should be stored outside of the secure container.	Failed	Sensitive data is stored in MongDB without encryption, but authentication is enabled	Docker encryption, auth protection	Passed
DATA.2	No sensitive data is written to logs.	Failed	E-mail was logged	Logging logic was adapted to remove e-mail logging.	Passed
DATA.4	The software does not hold sensitive data in memory longer than necessary, and memory is cleared after use.	Failed	Depends on MongoDB, but Java does not meet this requirement by default	—	Failed
AUTH.4	The remote endpoint terminates the existing session when the user logs out.	Failed	The token was deleted only from the local storage.	—	Failed
AUTH.6	The remote endpoint implements a mechanism to protect against excessive submission of credentials.	Failed	Rate limiting was not configured in the Zuul proxy	Change in Zuul configuration	Passed
COMM.1	Outgoing data is encrypted on the network using transport layer security.	Failed	HTTP only endpoints found	Change in configuration	Passed
QUAL.3	All third party components used by the software are identified and checked for known vulnerabilities.	Failed	254 CVEs found	—	Failed
QUAL.4	The exceptions are handled properly.	Failed	Qualitygate warnings	—	Failed
CRYP.3	Cryptographic primitives are used according to industry best practices.	Passed	BCrypt was used for password hashing	Purposefully insecure MD5 hashing algorithm configured	Failed

The project version used for the baseline assessment can be found at commit 6bb2cf9ddbca980b664d3edbb6ff775d75369278.

as depicted in the first four columns of Table 2. For brevity, we include one passed requirement (actually 15 requirements passed the baseline assessment) and focus on the failing requirements. For instance, the requirement DATA.2 failed the baseline assessment. Security-sensitive data (e.g., e-mail, passwords, keys, certificates) and privacy-sensitive data (e.g., PII) should not be written to log files, even in case of crashes, since access to the system or application logs could provide valuable information to the attacker. "Listing 6" shows an excerpt of the output of the filtering rules.

Since e-mail addresses were logged without encryption, personal data were logged and the authors concluded that this requirement failed the baseline assessment.

Project changes and delta-assessment. The delta-assessment was fully automated based on the rules defined by the analyst. To show how the delta-certification method facilitates the automated assessment, we modified the PiggyMetrics project by introducing changes that correct previously failing requirements. For the purpose of the case study, we injected one change that failed a previously passing requirement. The results of the delta-assessment are shown in the last two columns of Table 2. For all of the introduced changes, the automated rules correctly assessed the requirements. Since more than one requirement failed the delta-assessment, such a project state would not result in obtaining the desired certificate.

Validation With Experts

To achieve industrial relevance, we have defined and measured key performance indicators (KPIs) presented in Table 3.

Listing 6: Output of assessment rule for DATA.2.

```
/piggymetrics–master/account–service/
     src/main/java/com/piggymetrics/
     account/service/AccountService
     Impl.java:87

line: log.debug (" account changes
has been saved ", name);

/piggymetrics–master/notification
     –service/src/main/java/com/
     piggymetrics/notification/
     service/EmailServiceImpl.java:51

Line: log.info (" e–mail notification
has been sent to ", type, recipient.
getEmail ());
```

First, we gathered feedback from an expert advisory board on the industrial relevance of DeltaAICert during a yearly project review meeting. (The advisory board included, among others, members of the U.S. Department for Homeland Security, open source, multinational automotive corporation, and standardization body.) The advisory board confirmed that the produced outcomes of the DeltaAICert tool can be useful for security researchers, developers, managers, and security auditors/evaluators. When asked to rank the importance of KPIs (from most important to least important), the experts chose the following order: accuracy of the new certification, detection accuracy of failed requirements, usability and functionality satisfaction, the time needed for the delta-assessment, the time needed for the baseline evaluation, and difficulty in completing the process. This confirms that the measures from the experiment are practically relevant and that reducing the time needed to perform the delta-assessment while maintaining a high accuracy of the new certification is a top priority.

Second, we conducted a short (1 h) in-person workshop with four experts from the ETSI CYBER group at a project plenary meeting. The structure of the workshop followed a short introduction to the context of delta-certification, an explanation of the tool, a tool demo, a brief quiz, and a roundtable discussion. Notes were taken during the workshop by a dedicated person. ETSI welcomed the concept of delta-evaluation and more emphasis on automation. For instance, participants agreed that the tool can be useful for developers and that it could help increase the security evaluation speed. The main topic of interest during the roundtable was the scalability of delta-assessments and security evaluation at the system level. For instance, the experts pointed out that the runtime environment can also

Table 3. KPIs for measuring DeltaAICert industrial relevance.

KPI	Description
KPI_1	Time needed for the first evaluation (technical)
KPI_2	Time needed for the delta-evaluation (technical)
KPI_3	Detection accuracy of failed requirements (technical)
KPI_4	Accuracy of the new certification (technical)
KPI_5	Difficulty in completing the process with respect to skill level (qualitative)
KPI_6	Usability and functionality satisfaction (qualitative)

influence the security properties, which are captured among the environment-related requirements at various (component, guest, host) levels. For the PiggyMetrics project, the automated rules returned the result on average in 2 s, which is reasonable to support the assessor in making the decision. However, the scalability of the approach on a larger code base and with more custom rules is subject to future work.

Answer to RQ1. The experts confirmed that reducing the time needed to perform the delta-assessment while maintaining a high accuracy of the new certification is a top priority and that the tool can be useful for developers and that it could help increase the security evaluation speed.

Experiment

We conducted an experiment with 69 computer science M.Sc. students and six practitioners to investigate whether the automated rules effectively lower manual effort in performing the delta-assessment.

Intervention. To this aim, we randomly assigned the participants to two groups. Group A (control) used the DeltaAICert tool to perform a delta-assessment without using the automated rules and group B (intervention) used the tool to perform the delta-assessment with automated rules.

Measures. We measured the actual effectiveness of the delta-assessment and the time spent on the task. The actual effectiveness was calculated as the precision of final assessments for the requirement (passed versus failed), namely the number of correct requirement assessments divided by all of the assessments. We also measured perceived usefulness of the tool in a short posttask questionnaire.

Setup and training. We conducted the experiment during a course taught at the university premises. The data gathered for research purposes were volunteered by the students. To control the environment, we set up virtual machines (VMs) with a preconfigured DeltaAICert tool, and on the day of the experiment, we collected the data with an online survey tool (www.qualtrics.com). We provided training to prepare the students for the task. All the participants finished before the allotted time of 90 min.

Task. The groups were tasked with performing a delta-assessment for one requirement with the DeltaAICert tool on a new project version. Group A (control) was tasked with analyzing AUTH.6 and group B was tasked with analyzing DATA.2 (see Table 2). The task involved running the assigned VM (with a preconfigured tool) and either manually inspecting the project changes (group A) or running the automated rules (group B) and finally making an assessment (pass/fail) for the requirement.

Execution with practitioners. Finally, to extend the evaluation and measure the KPIs with experts, we conducted an internal evaluation at SLAB (following a similar experimental setup) with six experts. In addition, we measured the time spent on the initial and delta-assessments of the PiggyMetrics project.

Results. Table 4 depicts the results (means) of the main measures of the experiment. First, students from both groups had a very low experience (on average, 0.4 on a scale of 0 to 4) with programming in Java, container technologies, pen-testing, and security standards.

In both groups, the correct final assessment is that the requirement is passed in the new project version. On average, the participants across the two groups reported the results with the precision of 0.85. However, the group using automated rules (group B) used less time (≈33% less) to achieve the same quality of result. In addition, we measured to what extent the participants agreed that the tool was useful to identify certification compliance or compliance violations for the new changes in the repository. On average, the experimental group (group B) agreed with this statement (2.35) while group A was neutral (1.97).

Table 5 presents the results from running the experiment with practitioners. We found that the

Table 4. Results from the experiment.

Group	No. of Students	Experience [0–4]	Precision [0–1]	Time [min]	Usefulness [0–4]
A	35	0.42	0.85	43.2	1.97
B	34	0.41	0.85	32.6	2.35

The values in bold indicate improvement of the experimental group (B) compared to the control group (A).

Table 5. Evaluation with six SLAB experts.

Expert	Group	Baseline Evaluation [hours]	Delta Evaluation [hours]	Quality of Evaluation Report [1–10]	Accuracy of Assessment Results [1–10]	User Experience [1–10]	Perceived Difficulty Level [1–10]
1	A	6.5	4.5	7	8	6	7
2	A	7	5	7	7	6	8
3	A	6.8	4.7	8	8	7	7
4	B	6.5	3.5	9	9	8	5
5	B	6.7	3.3	9	9	7	5
6	B	6.6	3.4	9	9	6	6
Average improvement in B		+0,03%	+0,28%	+0,19%	+0,15%	+0,21%	+0,27%

Group B was using the automated rules for delta-assessment, while group A performed the same task without the rules.

baseline assessment takes the same amount of time whether performed with (on average, 6.6 h) or without (on average, 6.67 h) DeltaAICert. Consistent with our result in the experiment, the experts were able to perform the delta-assessments ≈30% faster (3.4 h with and 4.7 h without DeltaAICert). In addition, we found that the quality (self-reported by experts on a scale from 1 to 10) of the evaluation report and accuracy of the assessment increased (without: 7.6, with DeltaAICert: 9).

Answer to RQ2. The evaluation with experiments shows that, compared to a manual assessment, the automated rules achieve at least ≈30% speed-up in delta-assessment. The student groups using automated rules took ≈33% less time to achieve the same quality of result. In addition, practitioners using automated rules experienced similar benefits in terms of speed-up (≈30%) compared to the students.

Threats to Validity

A few students (four in group A and two in group B) have performed a correct delta-assessment but have not selected the corresponding final assessment as passed. Based on the students' own justification of the assessment, we corrected their final assessments as "passed" to reflect that. A possible explanation is that the students have reconsidered their assessment but forgot to change their initial final assessment, which could be solved by improving the user interface to detect a change of justification text and prompt the user to verify their final assessment. We speculatively removed these data points, but found only a small drop in precision for group A (to 0.83) and small increased precision for group B (to 0.87), which does not impact the main results of the evaluation.

To avoid the risk of exposing the participant to a complicated task, we included control questions that measured their levels of understanding of the material presented and the task they were asked to perform (which were good in both groups).

This article presents an approach to resolve the problem of delta security certification for open source software supply chains. Resolving the delta-certification could help organizations to reuse open source components and manage (re)certification of components and their compositions up the software supply chain. In addition, tools that bring the cybersecurity requirements closer to development could help reduce the manual effort for the regulatory practice or security certification.

Our validation shows that even for nonexperts, using the automation features of DeltAICert resulted in at least 30% reduction in the time required for finishing the delta-assessment while maintaining the level of accuracy of the manual delta-assessment. Investigating the scalability of the approach with more custom rules and a larger code base is planned for future work. ■

Acknowledgment

This work involved human subjects or animals in its research. The authors confirm that all human/animal subject research procedures and protocols are exempt from review board approval. This work was funded by the European Union under the H2020 Program under

Grant 952647 (AssureMOSS). We would like to thank Security Evaluation and Research-Laboratory developers of the DeltAICert tool for their effort and the members of the AssureMOSS project for their support and useful discussions.

References

1. S. M. Kerner, “Colonial pipeline hack explained: Everything you need to know,” *Tech Target*, Apr. 2022. [Online]. Available: https://www.techtarget.com/whatis/feature/Colonial-Pipeline-hack-explained-Everything-you-need-to-know
2. A. Sharma, “What you need to know about the codecov incident: A supply chain attack gone undetected for 2 months,” *Sonatype*, Apr. 2021. [Online]. Available: https://blog.sonatype.com/what-you-need-to-know-about-the-codecov-incident-a-supply-chain-attack-gone-undetected-for-2-months
3. FireEye, “Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor,” *Mandiant*, Dec. 2020. [Online]. Available: https://www.mandiant.com/resources/blog/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor
4. I. Gartner, “Gartner identifies top security and risk management trends for 2022,” *Gartner*, Mar. 2022. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2022-03-07-gartner-identifies-top-security-and-risk-management-trends-for-2022
5. L. Vaas, “One year after log4shell, firms still struggle to hunt down log4j,” *Contrast Security*, 2022. [Online]. Available: https://www.contrastsecurity.com/security-influencers/one-year-after-log4shell-firms-still-struggle-to-hunt-down-log4j
6. W. Enck and L. Williams, “Top five challenges in software supply chain security: Observations from 30 industry and government organizations,” *IEEE Security Privacy*, vol. 20, no. 2, pp. 96–100, Mar./Apr. 2022, doi: 10.1109/MSEC.2022.3142338.
7. K. Tuma and M. Widman, “Seven pain points of threat analysis and risk assessment in the automotive domain,” *IEEE Security Privacy*, vol. 19, no. 5, pp. 78–82, Sep./Oct. 2021, doi: 10.1109/MSEC.2021.3093137.
8. J. L. Hernandez-Ramos, S. N. Matheu, and A. Skarmeta, “The challenges of software cybersecurity certification [Building Security In],” *IEEE Security Privacy*, vol. 19, no. 1, pp. 99–102, Jan./Feb. 2021, doi: 10.1109/MSEC.2020.3037845.
9. *Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and General Model*. (Apr. 2017). Common Criteria Portal. [Online]. Available: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
10. “EUCC scheme: Crossing a bridge: The first EU cybersecurity certification scheme is availed to the Commission,” European Union Agency for Cybersecurity (ENISA), Athens, Greece, 2021. [Online]. Available: https://www.enisa.europa.eu/news/enisa-news/crossing-a-bridge-the-first-eu-cybersecurity-certification-scheme-is-availed-to-the-commission
11. “EUCS – Cloud services scheme,” ENISA, European Union Agency for Cybersecurity (ENISA), Athens, Greece, Dec. 2020. [Online]. Available: https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme
12. H. Santos, A. Oliveira, L. Soares, A. Satis, and A. Santos, *Information Security Assessment and Certification Within Supply Chains*. New York, NY, USA: Association for Computing Machinery, 2021.
13. T. Arnold. *Common Criteria: Delta Evaluation*. (2006). Common Criteria Portal. [Online]. Available: https://www.commoncriteriaportal.org/iccc/7iccc/t1/t1201130.pdf
14. D. Distefano, M. Fähndrich, F. Logozzo, and P. W. O’Hearn, “Scaling static analyses at Facebook,” *Commun. ACM*, vol. 62, no. 8, pp. 62–70, Aug. 2019, doi: 10.1145/3338112.
15. E. Jeges, B. Berkes, G. Eberhardt, and B. Kiss, “MEFORMA security evaluation methodology - A case study,” in *Proc. 4th Int. Conf. Pervasive Embedded Comput. Commun. Syst. MeSeCCS, (PECCS)*, Setúbal, Portugal: SciTePress, 2014, pp. 267–274, doi: 10.5220/0004919902670274.

Ákos Milánkovich is a security researcher at Security Evaluation and Research-Laboratory Ltd., 1117 Budapest, Hungary. His research interests include wireless sensor networks, agricultural monitoring, UWB localization, and security. Milánkovich received a Ph.D. in energy efficiency of wireless sensor networks from Budapest University of Technology and Economics, Hungary. Contact him at akos.milankovich@search-lab.hu.

Katja Tuma is an assistant professor at the Vrije Universiteit, 1081 Amsterdam, The Netherlands. Her research interests are at the intersection of software engineering, security, and risk analysis, with a particular interest in human aspects. Tuma received a Ph.D. in computer science and engineering from the University of Gothenburg, Sweden. She is a Member of IEEE. Contact her at k.tuma@vu.nl.