Hochschule
Augsburg University of
Applied Sciences

Tools for Security Testing in Continuous Integration Pipelines

Michael Wager
Faculty of Computer Science
University of Applied Sciences Augsburg
Augsburg, Germany
mail@mwager.de

***Abstract*—Nunc viverra nibh at magna vulputate, sit amet accumsan purus elementum. Aenean rhoncus augue ac pretium pretium. Integer facilisis vestibulum porta. Curabitur nisl ligula, sollicitudin non eleifend sed, accumsan interdum mi. Nullam semper vestibulum eros vitae tempor. Vestibulum magna ipsum, auctor ut sem vitae, imperdiet egestas nibh. Sed ante nunc, blandit id tortor sit amet, consectetur cursus purus. Sed id velit laoreet, accumsan mi non, congue nulla. Morbi magna libero, vestibulum quis ipsum a, malesuada pulvinar eros. Mauris vulputate, nulla vitae lacinia efficitur, purus augue sollicitudin ante, non dapibus est massa a tortor.**

I. Introduction

Software development is still a very young craft. But we are writing the year 2022 and the amount of devices containing more and more complex software is growing rapidly. Comprehensive quality assurance, especially in the field of web application development, also could deserve a lot more attention. Nevertheless, while a few years ago it was quite common practice to edit files on a production server directly via FTP in order to deploy a fix or a change, today methods and processes to assure a certain level of quality are used quite often. One of that method is so called Continuous Integration (CI). The idea is to provide immediate feedback to the developers directly after the integration of changes to a codebase. On every push to the repository a so called CI server starts executing a pipeline of predefined tasks to ensure high quality and to make sure the changes do not break existing functionality. Common tasks are static code analysis to ensure a clean coding style or to catch simple development mistakes or unit testing to make sure the codebase maintains a high quality standard.

As this is also still a very young process, it is pretty obvious that software security also does not play a big role. But as the growing interconnectivity through the pandemic and also recent attacks against critical infrastructure moving through the media, cyber security gets a greater attention in recent times.

- Früher common DANACH zb durch pen testing, -¿ dann intro zum SDL! elcher schon eher den prozess fixen will! next satz is bullshit:

In the context of the Security Development Lifecycle [1], one idea to ensure a great level of application security is to automate as much as possible directly during the CI pipeline.

This work tries to provide an overview of the current state of continuous security testing, its categories, possibilities and limits and will also present some existing tools. The paper is structured as follows: First it will give an overview of related work and some technical background before going through tools. Finally a discussion of the findings will be ended with a conclusion.

II. Related work

[2] [3] [4] [5] [6] [7] –¿ halt checken was da ggf fehlt, bissl drauf eingehen und zusammenfassen

Summarizes work on related problems/approaches, but no background knowledge Differences of previous work and own work are clearly stated

- Check andere Papers -¿ SO: Was macht anderes Paper und was mach ich hier anders? nur 1-3 Sätze pro Paper! -¿ auch: von wann sind die papers? Sind die aktuell? Ist das noch relevant und/oder was hat sich geändert? -¿ nur die die ziemlich ähnlich sind? -¿ Ähnliches Ziel. So wie alle die ich bis jetzt gelesen hab :D

III. Background

This chapter provides a quick overview of basic concepts the reader has to understand in order to further grasp the contents of this paper. It will focus on the concept of continuous integration which is an important part of so called ”DevOps” (Development and operations) as well as current efforts to

automate application security assurance in modern software development products ("DevSecOps").

A. DevOps & Continuous Integration

Back in the days of early software development it was quite common to develop a product for months or even years and finally start to integrate it - i.e. preparing the release of the product. Stakeholders and developers had no clue how long this could take and lots of problems where introduced with this kind of process. To solve this problem, the idea of integrating often was first introduced in the extreme programming software development process [8]. The idea is simple: after making any change to a software product, be it a new feature of a bug fix, implementing and testing it locally on a developer machine, it will be commited to the source code control repository. Once a so called CI server recognizes my changes, it starts building and testing it again but this time it may also contain changes from other developers. The CI server will execute a so called build, including a pipeline of tasks to be executed in order to check if everything is ok. If this build fails the developer making the change will get notified immediately. And this is the whole point: integration errors are detected immediately and can get fixed right away, making the integration process much more transparent.

Some practices necessary to make this work are:

- Usage of a single source code repository
- Automation the build using a CI server - every commit should build the product on the integration server
- Adding tasks to the build pipeline like static code analysis and unit tests which are executed after building
- Notifying all relevant parties of the project about failing builds

Nowadays, CI is quite heavily used in modern development projects and there are also ideas and efforts taken to automate security assurance inside the build pipelines of a CI server.

B. DevSecOps

Traditionally, security assurance if a process done mostly at the end of the software development lifecycle. Security testing methods like penetration testing will concentrate on finding vulnerabilities in already (or soon to be) released products. This is good and needed, the term DevSecOps just suggest that security assurance may be automated as well as things like unit testing inside the CI server build pipeline - all while the development process is still ongoing. This work will focus on investigating tools and methods to achieve that.

IV. Tools

Hauptaufgabe des Papers: Ordnung in Chaos

- Kategorisierung hier! dann pro Kat ein paar Tools - auch auf CIs eingehen? Jenkins, GitLab etc

Own ideas and contributions constitute main part of the content

A. SAST

Hier ggf fokus auf JavaScript, PHP und Rust oder so.

Hier auch dependency checks?

B. DAST

Fokus auf OpenSource!

C. IAST

??? google!

D. Analysing containers???

V. Discussion

Kapitel eher nennen Discussion oder Evaluation

- und wird analysiert was so vorher zusammengetragen wurde! - Eigenschaften und Vergleiche der Tools

- zB Languages -¿ unter zb diesem Punkt könnte man dann evaluieren. Zb 90prozent JavaScript

OSS gibts nix für Go, bla bla

- Faktor Mensch: Es gibt studien mit Interviews, was sind Knackpunkte, wie reagieren Entwickler, weil Zeitgründe, viele Warnungen etc. Was kann man hier tun.?

- Was hab ich denn sehen nach meiner Tool recherche!?????????

- EMbedded / WebDev / Desktop dev etc?!?!?

- Wie gehts weiter mit embed, Industrial Bereich? etc.....

- hier mussen paar Fragen entstehen die ich dann in Conclusion beantworten kann!!!!!!!!!

- dieses literatur paper hat einige sachen drin für hier! zb missing knowledge unter devs, inter team collaboration issues ETC!!!!!!!!!

VI. Conclusion

- Fragen aus Discussion nun beantworten!

Summarizes results of the paper States relevance of results in a broader scope

References

[1] M. Howard and S. Lipner, "The security development lifecycle," 2006.

[2] J. Kuusela, "Security testing in continuous integration processes," 2017, https://aaltodoc.aalto.fi/bitstream/handle/123456789/27065/master_Kuusela_Juha_2017.pdf?sequence=1&isAllowed=y.

[3] P. Thulin, "Evaluation of the applicability of security testing techniques in continuous integration environments," 2015, https://www.diva-portal.org/smash/get/diva2:784545/FULLTEXT01.pdf.

[4] P. Koskela, "Automated security testing utilizing continuous integration and continuous delivery technologies," 2021, https://www.theseus.fi/bitstream/handle/10024/502952/Opinnaytetyto_Koskela_Pyry.pdf?sequence=2.

[5] J. Immonen, "Web application security testing as part of continuous integration in .net projects," 2015, https://www.theseus.fi/bitstream/handle/10024/103333/Immonen_Joona.pdf?sequence=1&isAllowed=y.

[6] E. Viitasuo, "Adding security testing in devops software development with continuous integration and continuous delivery practices," 2020, https://www.theseus.fi/bitstream/handle/10024/342349/Opinnaytetyo_Ella_Viitasuo_pdfa.pdf?sequence=2&isAllowed=y.

[7] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting devsecops: A systematic review," 2022, https://www.sciencedirect.com/science/article/pii/S0950584921001543#d1e1647.

[8] K. Beck, "Extreme programming explained: Embrace change," 1999.