jamk.fi

Web application security testing as part of continuous integration in .NET projects

Joona Immonen

Master's Thesis
December 2015

Master´s Degree Programme in Information Technology
Cyber Security

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author	Type of publication	Date
IMMONEN, Joona	Master's Thesis	25.11.2015
	Pages 43	Language English
		Permission for web publication (x)

Title

Web application security testing as part of continuous integration in .NET projects

Degree Programme

Master´s Degree Programme in Information Technology

Tutors

SALMIKANGAS, Esa

Assigned by

Solita

Abstract

The purpose of this study was to investigate security controls for continuous integration in .NET projects. The thesis introduces common theory related to the area; web application security testing, continuous integration, continuous delivery and DevOps. In the problem statement the defense in depth model is introduced and typical threats for modern web applications are introduced. The study introduces two example projects and several security controls.

Current state analysis was done for both example projects and a research was conducted about suitable security controls that market has to offer for .NET projects. After having the initial situation cleared a mitigation plan was done for the risks and to control different types of threats. Each of the security controls was evaluated for usage in Jenkins. For most of the security controls implementation is also introduced and evaluation about usage of the tool. Security controls usefulness is evaluated against different types of threats that web applications are facing and against feelings that example projects´ team members have towards each control.

As a results the thesis introduces usefulness estimates for each security control and also guidance how they can be taken into use in a Jenkins continuous integration environment. As the main result the study presents how combining different kind of tools can help in mitigating typical risks that a web application project is facing. Education is found to be in a crucial part in facing the threats. Software developers in example projects felt that static code analysis tools and performance analysis tools were the most useful ones for their projects.

Keywords

Continuous integration, Continuous delivery, security, security testing, DevOps, Jenkins, .NET, OWASP

Miscellaneous

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä IMMONEN, Joona	Julkaisun laji Opinnäytetyö, ylempi AMK	Päivämäärä 25.11.2015
	Sivumäärä 43	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty (x)

Työn nimi
Web application security testing as part of continuous integration in .NET projects

Koulutusohjelma
Informaatioteknologian koulutusohjelma

Työn ohjaaja(t)
SALMIKANGAS, Esa

Toimeksiantaja(t)
-

Tiivistelmä

Tutkimuksen tarkoitus oli tutkia erilaisten tietoturvakontrollien toteuttamista Continuous Integration -ympäristössä .NET-projekteissa. Tutkimuksen kohteena oli kaksi esimerkkiprojektia, joille tehtiin tietoturvatilannekartoitus. Tilannekartoitus tehtiin myös markkinoilla oleville tietoturvatuotteille.

Tämän jälkeen tutkittiin teknisiä tietoturvakontrolleja, joita projektit voisivat mahdollisesti ottaa käyttöön. Tutkittuja tietoturvakontrolleja verrattiin erilaisiin uhkamalleihin ja ne sijoitettiin tietoturvallisen sovelluskehityksen elämänkaaressa sopivaan vaiheeseen ja lopuksi arvioitiin esimerkkiprojekteilla niiden hyödyllisyys projektin kannalta.

Tutkimuksen tuloksena saatiin arvio eri tietoturvakontrollien hyödyllisyydestä ja osatuloksena ohjeistus kuinka ne voidaan ottaa käyttöön Jenkins Continuous Integration -ympäristössä. Koulutus tunnistettiin tärkeäksi osaksi sekä tietoturvauhkien tunnistamisessa että tietoturvauhkilta suojautumisessa. Projektiryhmien mielestä tärkeimmät työkalut olivat suorituskykytestauksen mittaamiseen ja lähdekoodin analysointiin käytettävät työkalut. Tutkimuksessa tuli selkeästi esille, miten eri työkalut tarkastelevat täysin eri osia sovelluksen ajoympäristöstä ja siten myös mitätöivät täysin erilaisia uhkia tehokkaasti. Lopputuloksissa erilaisten tietoturvatuotteiden rinnakkaiskäytön hyödyllisyys näkyi siten, että yhtäkään tuotetta ei koettu tarpeettomaksi.

Avainsanat (asiasanat)
Jatkuva toimitus, DevOps, Jenkins, Jatkuva yhdentyminen, .NET, tietoturva, tietoturva testaus, OWASP

Muut tiedot

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

1.1 Necessity of thesis

Continuous integration and security testing are trendy topics in software industry at the moment. They are both important aspects of software development but only rarely brought together. This thesis focuses on how to integrate automatic security testing into a project´s continuous integration processes. Goal of this study is to understand the benefits of different kinds of security controls in a continuous integration pipeline.

According to Stuttard and Pinto (2011, 1) stakes are high with web application security. Users trust web applications with their own sensitive information, and meanwhile criminals are trying to compromise payment details. Reputation plays a critical role in this area of business.

1.2 Structure of thesis

The introductory chapter presents the theories related to the thesis. The second chapter presents the problem that the thesis discusses, and how analysis for current situation in the example projects was done. It also includes a mitigation plan for risks found in the analysis. After that there is security control implementation that goes through of different actions that were taken. Security controls and threats are compared, and finally there are chapters for personnel education and a retrospective questionnaire on how things were felt in example projects.

1.3 Commissioner

Solita is a Finnish digital business consulting and services company. Solita is developing new business and digital services for private and public sector customers. Solita had around 400 workers in 2015 and has earned 6th place in the Great Place to Work Finland, medium-sized businesses ranking in the same year.

1.4 Continuous integration

Continuous integration has been a driving idea for software development lately. The core idea behind continuous integration is to merge all developer work on a software project multiple times a day. To be able to forecast the quality of the merge there must be different stages of building, testing and deployment of code. To keep this cycle time efficient there must be many levels of automation. This thesis will focus on how to add web security test automation into that cycle.

Developing good software is to consistently follow good practices regardless of the chosen technologies. Continuous integration entails automation of some good practices. Integrating code multiple times a day will reduce the risks on a project when tests and inspections are run. Defects are easier to detect when they are introduced, and it is more expensive to fix them the later they are noticed. Continuous integration will also make the health of software to be measurable, which makes it possible to govern the maintainability of software. Another important point is that with building and testing in a clean environment continual basis assumptions are reduced. Assumptions are tedious since wrong assumptions are actually introducing a bug. With continuous integration repetitive processes are also reduced since processes are run in the same way every time. One of the main ideas of the continuous integration is also to make software deployable anytime which makes it cheaper and faster to get new features live. (Duvall, P. M., Matyas, S. and Glover 2007, 23-31)

In this thesis Jenkins is used as a continuous integration and continuous delivery application. Jenkins is a cross-platform application that is widely used. Jenkins can be used for a wide variety of languages and technologies, including .NET, Ruby, Groovy, Grails, PHP and Java. Jenkins has a low learning curve and will fit for the teams of all sizes. Jenkins has also hundreds of plugins for e.g. version control systems, build tools, code quality metrics and build notifiers. (Smart, J. F. 2011, 3-4)

1.5 Web application security testing

Security testing is done to reveal security flaws in software projects by validating and verifying the effectiveness of application security controls. Vulnerability in web application can be weakness in system design, implementation, operation or management. (Meucci, M. and Muller, A. 2014, 27)

OWASP testing framework is separated to five different phases Figure 1 presents these five phases of secure development lifecycle process as a continuum.

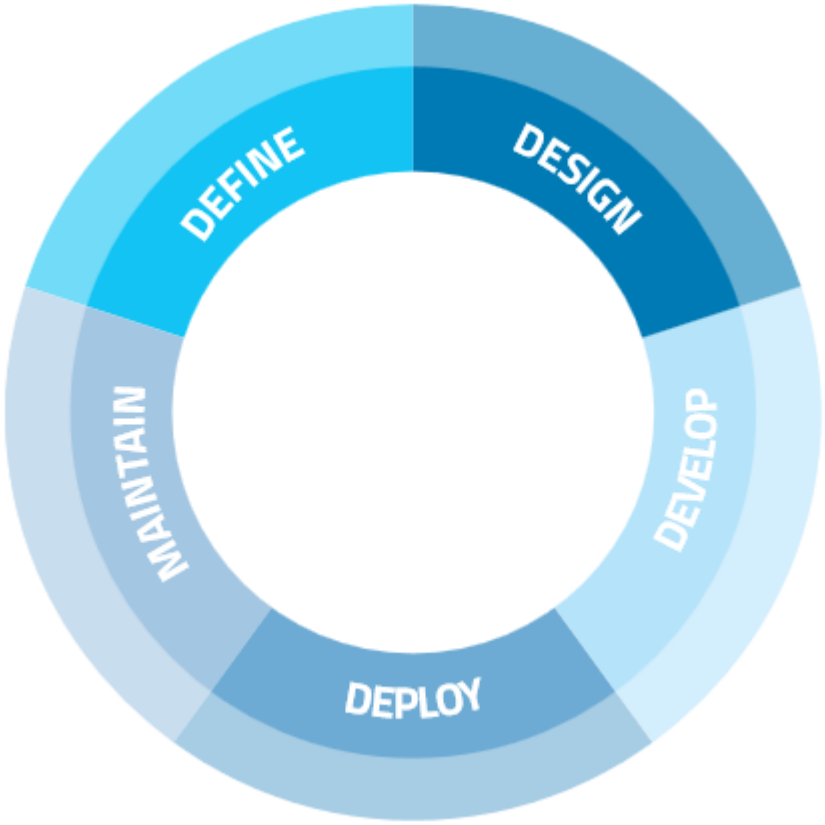

Figure 1 - Secure development life cycle (Meucci, M. and Muller, A. 2014, 12)

Before the development begins the development team should review policies and standards that they can follow during the deployment phase. This helps the team to face typical situations later on. During the design phase the team should make sure that they need to consider security mechanisms like user management, authentication, confidentiality and transport security. During development the team should make code reviews against a set of checklists containing common vulnerabilities,

known framework issues, industry specific requirements and business requirements for confidentiality, integrity and availability. In the deployment phase the team should carry out penetration testing and configuration management testing against the deployed application to ensure that it was deployed securely. Once the system has gone live the security testing should not stop there; instead, during maintenance and operations phases there must be health checks, change verifications and regressions tests setup for the system. Figure 2 presents how different actions are spread into different phases of secure development lifecycle. (Meucci, M. and Muller, A. 2014, 24-26)

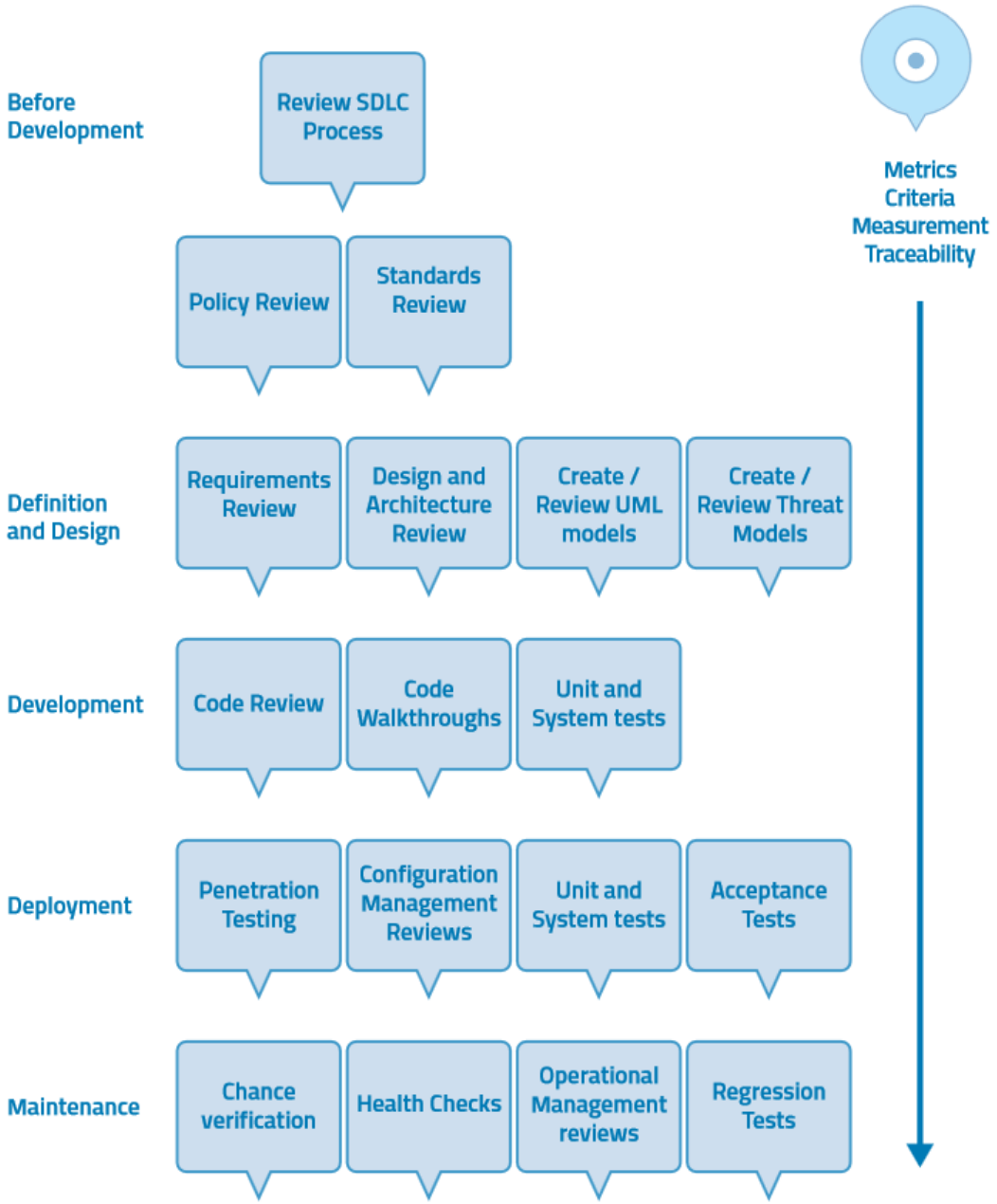

Figure 2 - OWASP Testing framework workflow (Meucci, M. and Muller, A. 2014, 26)

This thesis focuses especially on development and deployment phases of the OWASP testing framework. Yet many of the security controls introduced in this thesis can be used in other phases of the testing framework.

1.6 Continuous delivery and deployment pipeline

Continuous delivery focuses on getting new features for end users as soon as they are implemented and tested. Deployment pipeline consists of various tools that help to meet the delivery for the customer. By adding metrics to the equation; development team can find parts of pipeline that are not working efficiently or are unnecessary and do not provide any value. (Lehtonen, T., Suonsyrjä, S., Kilamo, T. and Mikkonen, T., 1-3)

Deployment pipeline metrics can be divided into two categories: implementation level and pipeline level. Deployment pipeline analyses how long it takes to develop a feature, to deploy it and to get it into use. Pipeline level instead calculates how many features the team can deliver to the end user per month. Figure 3 presents how different metrics are molded into feature branch driven development pipeline. (Lehtonen, T., Suonsyrjä, S., Kilamo, T. and Mikkonen, T., 9-11)

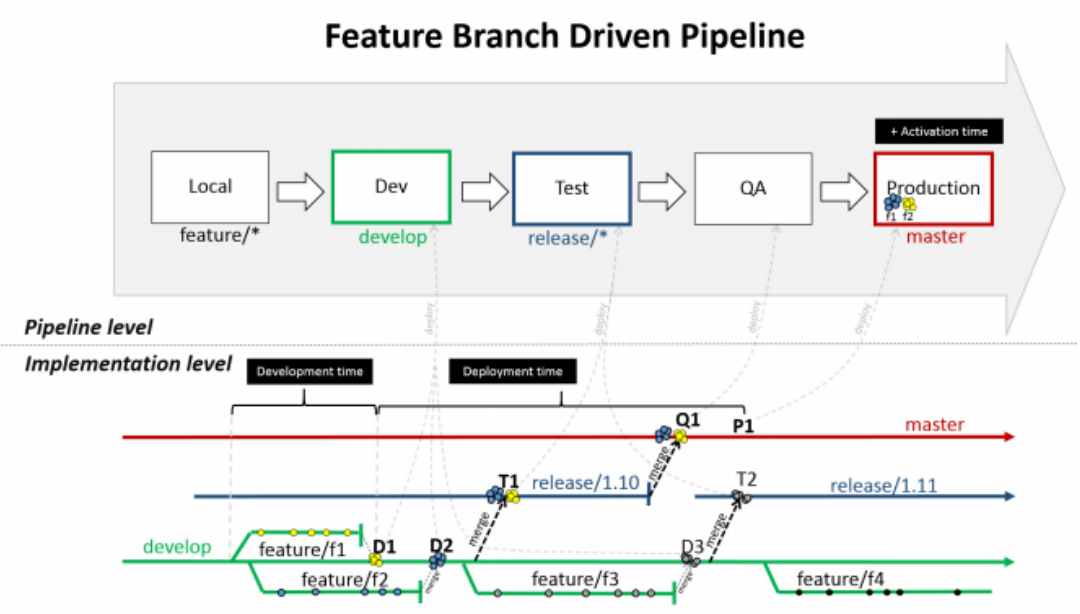

Figure 3 - Pipeline and development level metrics in feature branch driven pipeline (Lehtonen, T., Suonsyrjä, S., Kilamo, T. and Mikkonen, T. 7)

This thesis is interested in implementation level metrics of the deployment pipeline. Web application security testing can be a burdensome task if done manually. Security and code quality aspects will effect both development and deployment time.

1.7 DevOps

DevOps is about managing infrastructure as code. Nowadays infrastructure needs to be reproducible and testable quickly and repeatable, which means that an identical environment is needs be built up multiple times on continuous basis. This includes running virtual machines to shield runtime environment, automation of configuration for the virtual machine and monitoring tools to ensure that environments are running properly. Fault tolerance environments needs to be distributed and in cloud environment scaling has to be done on demand. (Loukides, M. 2012)

Web application security testing and continuous integration are tightly coupled with DevOps. In DevOps the configuration can be changed and multiple different variations of environments built up. Still all the environments needs to be secure thus security testing is needed in the process. This means that security testing automation has to be part of the DevOps and also monitoring naturally has also links to security too. Environment and its use dictates how heavily it needs to be monitored. This thesis is interested in deploying a web application into a web server and making sure that the environment from network level to application binaries is secure to use.

2 PROBLEM STATEMENT

2.1 Defense in depth model for .NET development

Onion model of defense in depth was changed to match the development perspective more. The end result was a pyramid structure as described in Figure 4 that covers network, host, application server, application, configuration and code.

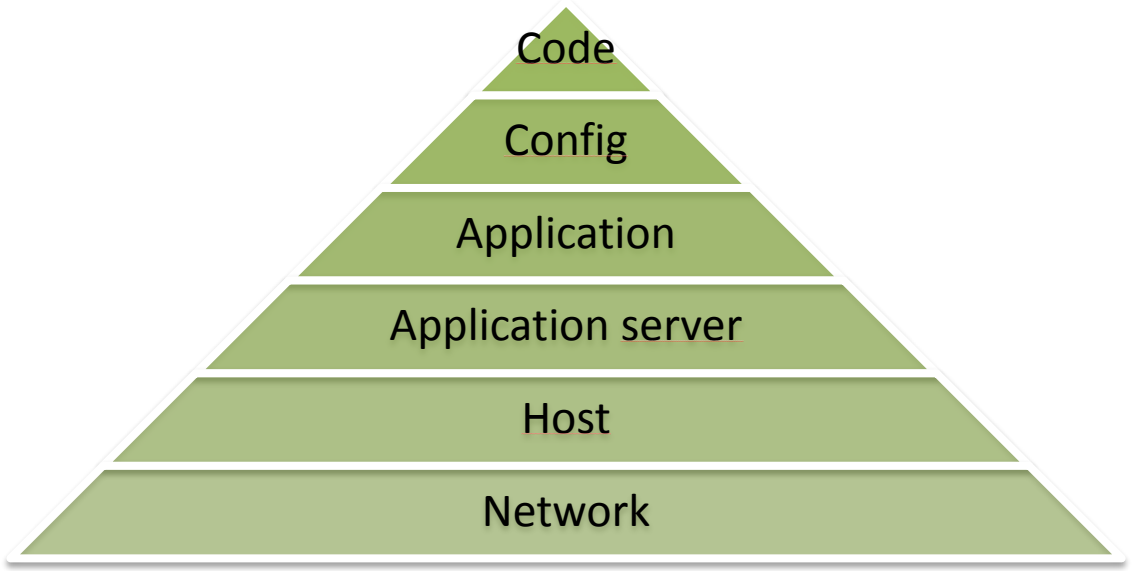

Figure 4 - Defense in depth model from .NET development perspective

In production environment there always has to be a network from where users can connect to an application. There must also be a host to be base for the application server that actually runs the application. The application always has configuration and code. To restrict malicious users from impacting business related application parts, any of these six layers can be used as a restrictive layer to forbid access further.

One could argue that data needs to be on top of the pyramid; however it has been left out for now. The only security controls that can be set on data level are authorization and encryption and both can be checked with unit tests in continuous integration so the data layer is not needed in this scenario.

2.2 Threats from OWASP Top 10

The Open Web Application Security Project (OWASP) is a non-profit organization that is based on volunteering. OWASP is dedicated to helping companies to develop, purchase and maintain secure applications. OWASP Top 10 threats is based on 500 000 vulnerabilities across many organizations and applications. The threats that have been brought on the list have been measured from perspective of exploitability, detectability and impact estimations.

On top of the OWASP Top 10 list is a threat called injection, which means that attacker can include in commands or queries into input data and will trick the system into executing unintended commands. Maybe the most common of this type of threats is SQL injection where malicious user may be able to fetch an unlimited amount of data from the database.

The second on the list is broken authentication and session management threat. This means that it is common that applications may leak passwords to malicious users or malicious users may be able to steal legitimate users´ session tokens and act on someone else's identity.

The third on the list is Cross-Site Scripting (XSS). XSS is a flaw where an application will take user input and send it back to web browser without proper validation or escape mechanisms. This allows attackers to run malicious scripts on someone else's web browser or redirect users into attacker's malicious website.

The fourth on the list is insecure direct object reference, which means that an application or application server does not have proper access control over implementation objects, files and folders. This lets attacker to change parameter value to directly refer instead of correct system object another system object, which leads to attacker being able to compromise data from the server.

The fifth on the list is security misconfiguration. This one is pretty generic since it covers configuration on operating systems, web servers, database servers, application itself and application frameworks. Depending on what was misconfigured the user can compromise that part of the system or the whole system.

The sixth on the list is sensitive data exposure. Sensitive data e.g. credit cards, identity information and authentication credentials must be protected with extra caution. Many web applications fail to do this. Attacker can steal or modify weakly protected data.

The seventh on the list is missing function level access control. Web applications need to verify user's right to access function on server side when privileged function

is called. If there is no verification then attackers can forge requests and possibly gain access to details that they are not permitted to.

Number eight on the list is Cross-Site Request Forgery (CSRF). Attackers force a logged-on user to send a HTTP request to a vulnerable web application. This forged request needs to contain victim's authentication information. Attacker then makes victims browser to generate requests for the vulnerable application trusted by that application because they are generated from a legitimate user´s browser.

The ninth on the list is using components with known vulnerabilities. Application servers, modules, libraries, framework and software can have known vulnerabilities, which make it possible for attacker to exploit such a known vulnerability which then may lead to any result depending of the type of the vulnerability.

The tenth on the list are unvalidated redirects and forwards. Some sites for example can take URL as a parameter where to redirect after login has occurred. If this parameter is not sanitized the attacker can forge any URL into request and redirect victims to malicious websites. (OWASP Top 10 – 2013, 1-6)

2.3 Threats from Cloud Security Alliances Notorious Nine

Cloud security alliance is a non-profit organization which tries to promote the use of best practices from security perspective within cloud computing to ensure security in cloud. CSA has published a list of top threats in cloud computing which is called "Notorious nine".

The first on the list are data breaches. In multitenant cloud environment the risks for data breaches are higher because of wider attack vector. In addition to traditional security risks cloud also introduces hypervisor level vulnerabilities that means that one virtual machine on same physical server might be able to extract information from another virtual machine. Also, a design flaw in SaaS can make it possible to leak information from one customer to another. What makes this even worse is that in special kind of cloud environment competitors might share the same SaaS application which means that they might leak their information directly to their competitor. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 8).

The second on the list is data loss. In addition to traditional threats like malicious attackers and natural disasters, cloud introduces new ways to lose data. Because of multitenancy a greater deal of information is encrypted on the cloud. Losing the encryption key may lead to loss of data. Losing credentials for cloud service management may also lead to tremendous amount of data loss, which happened to a journalist of Wired magazine: he lost all data he had in his Apple, Gmail and Twitter accounts. This threat also is present in each service model: IaaS, PaaS and SaaS. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 9).

The third on the list is account or service traffic hijacking. The threat itself does not differ much from on-premises environments. Losing credentials is always a bad thing. The reason why this is especially threatening in a cloud environment is the gain the attacker will have. With stolen service account attacker can easily set up new machines for botnets or obfuscate his true identity when attacking some other party. Two-factor authentication is strongly advised for mitigating this threat.

(Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 10).

The fourth on the list are insecure interfaces and APIs. Cloud computing provides a set of APIs their customers are using to interact with the cloud services. Customers are creating new environment, creating credentials, monitoring their environments and doing this all through APIs. A vulnerability in these APIs can have a huge impact on the cloud service. Cloud service provider might even need to take the whole cloud infrastructure down if they cannot be sure what legitimate requests are and what not. It is also important for the customers that they understand the seriousness of securing these APIs with strong credentials, encryption and multifactor authentication. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 12).

The fifth on the list is Denial of Service (DOS). While cloud environment has scaling capacities to mitigate the threat it also provides new problems into this domain. Successful DOS attack will cut all possibilities to manage the service since customer might not be able to even connect the system since the DOS has brought it down.

With automatic scaling the cloud service can also start to reserve so many resources that the bill for a customer rises in magnitudes. The problem in this situation is that the ability to cut out the attacker is limited. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 14).

The sixth on the list are malicious insiders. Malicious insiders are always a serious threat but the malicious insider with system administrator rights in cloud environments might be able to hijack or destroy all your environments in few clicks. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 16).

The seventh on the list is abuse of cloud services. This threat is about the fact that buying hardware for brute forcing encryption might be expensive. A cloud platform for that is most likely much cheaper, which interests malicious cloud users to harness the power of cloud to malicious use which leads cloud service providers into a situation where they need to have strict policies for legitimate usage. Cloud service provides will also need to detect this kind of activity which most likely leads to monitoring of cloud customers. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 18).

Number eight on the list is insufficient due diligence. Moving from internal network to cloud environment might lead customers into a situation where their applications will no longer meet the security criteria they met before because of missing internal network-level security controls. Customers need to understand the cloud service provider´s environment thoroughly; otherwise they might take risks of unknown level in their cloud implementation. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 19).

The ninth on the list are shared technology vulnerabilities. Compromise in any shared technology level such as hypervisor or SaaS application exposes multiple customers to threat. Single vulnerability or misconfiguration could possible take down the cloud service provider´s cloud infrastructure, which means that in a cloud both customers and cloud service provides must have a defense in-depth strategy to minimize the impact from such vulnerability. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013, 21).

In this thesis cloud is not a key topic; nevertheless, many of these threats affect software development since it might not be up to developer´s decision if the application is hosted in the cloud or not. Instead, developers are in the key position in making sure that their application meets the security needs for cloud services.

2.4 Overview of projects

Two projects were measured for this thesis. Both were public projects with the same kind of demand for security. Servers were hosted on the third party servers and there were no money transfers on the website. Both projects were given a security questionnaire (Appendix 1) to measure the starting level of the security testing.

First project

No security testing was implemented on the project and there was no guidance for secure development available. Project security requirements were not felt to be strict but rather very loose. Project members were quite happy about the situation and did not want to add any security automation into their project. The project was .NET CMS solution built in on top of EPiServer CMS for a rather large company.

Second project

The second project had security testing implemented by a third party. Nevertheless, the security testing report was not public information in the project, and it was not shown to all project members. Project members felt confident about fitting into customer's security policy; however, they were unhappy about the level of security testing and its automation. Project was .NET CMS solution built in on top of EPiServer CMS for a rather large company.

Summary of questionnaire

It was clear from the answers that no security testing had been done in the projects. The feelings about need of security were mixed. The first project team reflected that it needs none, and it felt that their customer was not that interested. The second

project felt that they need a higher level of security and they felt that they had a disappointing level of testing done; however on the other hand they felt that their level of security matches their customer's security policies.

Hosting

For both projects an external party is responsible for hosting business, which means that there is no control over network from Solita. On host Solita has only limited control and should not do any installations or upgrades without permission. The application server itself is more a responsibility of a hosting partner; Solita's responsibility is to configure it for the application, which means that the study is not very interested in network, host or application server level security since there is limited or no control over them.

Risk analysis

Brief analysis showed that the example projects were similar to each other. Both were CMS systems published on open internet. There was no sensitive data in the system, and the parts of the system demanding authentication were for content editors and administrators. This simplified the threat modeling since both projects were just websites among others. Most likely he threat agents would be script kiddies, bots or disappointed customers. The business impact for the site was more or less a strike against the brand, and no actual money flow would be affected. Based on this the risk analysis was done on a technical basis. (See Figure 5)

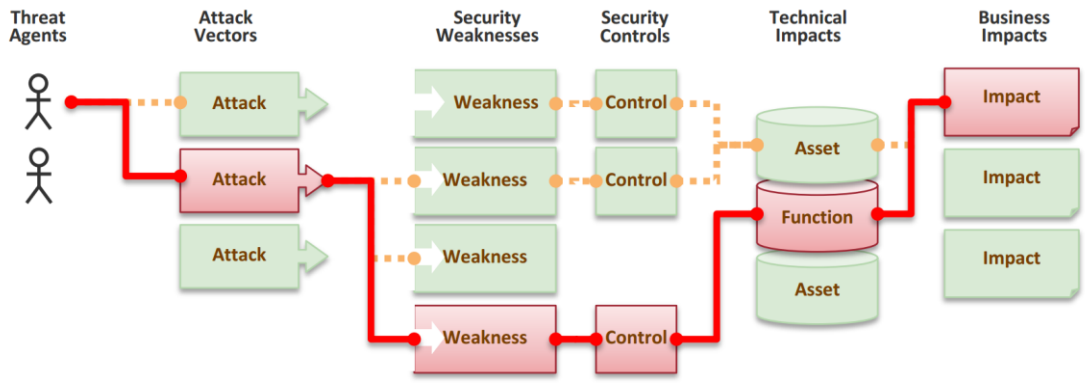

Figure 5 - Web application security risk factors (OWASP Top 10 – 2013, 5)

It was measured that the major risks in the security testing in both projects where somewhat the same. There was no guidance for developers available, no internal security testing done, and the threat modeling had not been done; thus both projects were lacking security testing automation.

The OWASP TOP 10 from 2013 were used as a threat reference.

3 IMPLEMENTATION PLAN

3.1 Plan for mitigating the risks

The plan to mitigate the risks can be divided into three types of actions.

1. Embrace the security perspective in project steering
2. Implement technical security controls
3. Educate personnel

It was clear that there was not enough security testing done in the projects. The personnel were aware of security threats; however in the project they were not maybe treated as seriously as they could have been. The first step to take would be to make the project steering personnel to put effort on managing the security on the project. Getting the support of the steering committee makes it relatively easy to get budget for implementing security controls in place. Last but not least, the personnel has to be educated to understand what kind of threats modern web applications are facing.

3.2 Controls for risk mitigation

Focusing on technical controls building on top of the continuous integration the biggest risks facing web applications are user inputs. This is in top of OWASP Top 10 list (OWASP TOP 10, 2013, 6) called as injection. The majority of the security controls tries to mitigate or at least narrow that risk. Some of the other risks in OWASP Top 10 list are also exploited via user inputs or fake inputs, e.g. insecure direct object reference and XSS vulnerability. Some of the other risks are hard to control via automated scripts since they are more related to architectural decisions. For example,

missing function level control and sensitive data exposure are hard to notice with scripts since it is hard to say what sensitive data is and how function level user access should be from automated perspective. Instead, these decisions should be reviewed in projects together with customer and development team.

3.3 Requirements for tools

Toolset for the study was done by finding out what kind of tools are available at the market. There is plenty of tools available, which lead to need to limit the number of controls studied. Technical security controls had few requirements that needed to be filled for study.

- Tool needs to run in Windows environment without too much effort
- Tool needs to be attachable to Jenkins (has a Jenkins plugin or command-line interface)
- Tool needs to be suitable for .NET C# projects (native Microsoft environment tool or easily installable on Windows)
- Tool needs to be cheap or a great value for money

With these requirements in mind, a great deal of possible security tools were cut out.

4 SECURITY CONTROL IMPLEMENTATION

4.1 Education

The company has decided to have mandatory OWASP TOP 10 education for all of its personnel. It is advised that the personnel goes through this education every two years to keep up to date with current security demands. There has been this kind of education also earlier but it has not been mandatory and it has not been ongoing.

With this action it should be easier for the project personnel to understand security threats they are facing and to create threat modeling for their project. Derived from understanding and threat modeling, it should be straightforward to also communicate with customers about their security policies and threat modeling.

From threat modeling it is also possible to continue into planning security controls that would mitigate risks from threat modeling.

4.2 Static code analysis with FxCop

4.2.1 Choosing the FxCop

FxCop is Microsoft application designed for code analysis. FxCop contains multiple rules, and Microsoft has provided multiple preset rule sets for it. The reason to choose FxCop was simply its easiness to integrate into solution. It is basically one click from the Visual Studio and it is present in both CI and local builds.

4.2.2 Implementing static code analysis with FxCop

Despite the various rule sets, none of them really fit the project needs therefore it was decided to plan a rule set of one's own for the project. The problem with the original rule sets is that they test various other things as well in addition to security. Some of the checks, e.g. checking property naming convention for right usage of English plural can be really frustrating when there is a huge amount of false positives that will not actually affect the application in any way. Since each rule can be cherry-picked as its own into FxCop rule set it is a wise thing to do to study the rule set and pick those that best fit the need. There are two actions to choose for rule: warning and error. If there is something that really should not happen then error should be used. If there is fear that there might be a huge amount of false positives it is suggested to use warning. Figure 6 presents how easy it is to enable code analysis in the project.

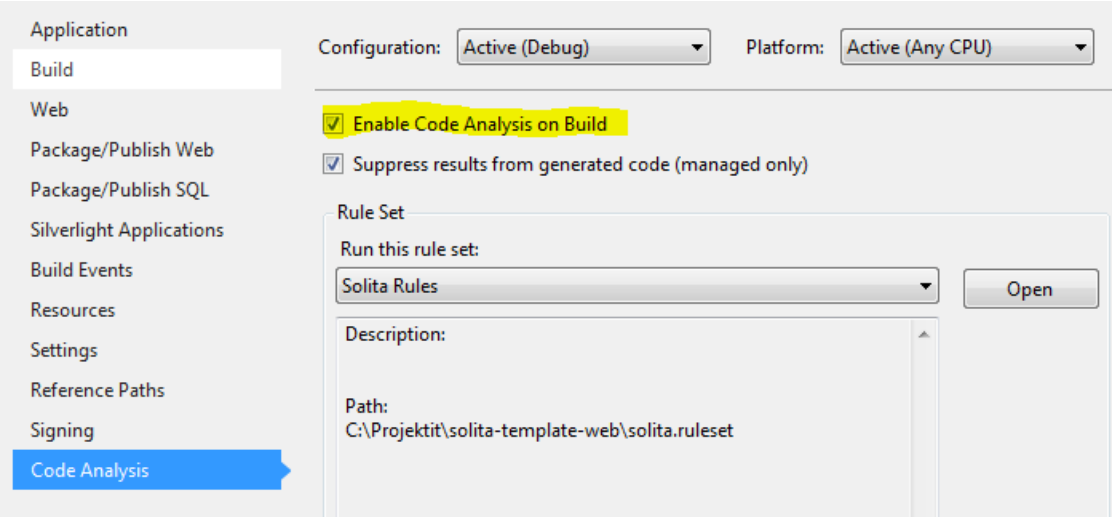

Figure 6 - FxCop is so easy to enable that it is plain stupidity to not to use it

4.2.3 Using static code analysis with FxCop

In the build server needs to have full installation of Visual Studio to get the right binaries there, or another way around is to copy a great deal of code analysis DLLs into path where Jenkins can find them during build.

If there is no "treat warnings as errors" option on the project then the actual warning on CI will not affect a single thing unless users are gathering information about the amount of warnings. It is stupid to not to handle warnings; therefore, treating warnings as errors would be wise thing to do if a suitable rule set for can be found. Treating warning as errors will stop compiling the project and fail the whole CI build. That is good since that means that the developers that created bad code in the first place will get notified of failed build, and the application with bad code in it will not even go into the test environment.

4.3 Static code analysis with VisualCodeGrepper

4.3.1 Choosing the VisualCodeGrepper

VisualCodeGrepper was found by accident when trying to figure out different static code analyses to the .NET. It is an open source project and is available at http://sourceforge.net/projects/visualcodegrepp/. It tries to be a tool that does not

return many false positives and is able to search intelligently for bad code or comments that are stating that the code below might not be ready. It was chosen to study because it fulfilled the main requirements.

4.3.2 Implementing script for continuous integration

The basic script for VCG was really easy to implement. There was a need only for setting few command line parameters to the script. The harder part occurred when getting the results into a format that was understood by CI build server. Unit test results in jUnit format were chosen, and the mapping was conducted with PowerShell function from one XML format to another.

4.3.3 Using VisualCodeGrepper in continuous integration

After the initial plumping was done it was straightforward to get the tool into use in CI build server. Of course the tool had to be installed into the server and Jenkins needed PowerShell plugin for running the script. Figure 7 presents how the example script was integrated into Jenkins build.

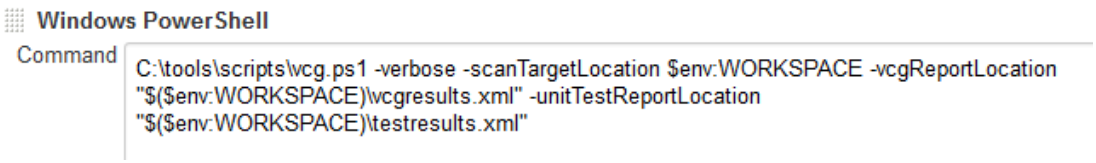

Figure 7 - Using VisualCodeGrepper in CI Build

4.4 Static code analysis with ReSharper command line tools

4.4.1 Choosing the ReSharper command line tools

ReSharper is a product of JetBrains. It can be integrated into Visual Studio where it immediately tells developer about the problems in his code without even building the code. ReSharper is really commonly used with Visual Studio and it has also command line tools available. The command line tools that were studied were Inspectcode.exe and DupFinder.exe. InspectCode.exe is able to make the static code analysis on a given solution. In addition to another static code analysis tools, it is able to scan also .js and .css files for problems, whereas DupFinder.exe is a tool for finding duplicated code in the solution.

4.4.2 Implementing ReSharper command line tools analysis for continuous integration

There is no actual Jenkins plugin for ReSharper command line tools available. The tool itself is relatively easy to run from command line in Jenkins build by just typing inspectcode.exe mysolution.sln /o=C:\Temp\inspecresults.xml or dupfinder.exe mysolution.sln /o=C:\Temp\dupresults.xml. The problem is with getting information out of the tools. Jenkins does not support either of the output xml file format and if users are willing to use them in Jenkins they most likely need to transform them into some other valid format as it was done with VisualCodeGrepper.

4.4.3 Using ReSharper command line tools in continuous integration

The actual usage of ReSharper command line tools was skipped since there was a plugin in SonarQube to integrate these tools there. Look for chapter 4.5 about SonarQube illustrates how ReSharper command line tools was integrated into Jenkins and SonarQube.

4.5 Static code analysis with SonarQube

4.5.1 Choosing the SonarQube

According to Campbell and Papapetrou (2013), SonarQube is a free open source code quality platform. It is able to produce moment-in-time snapshots of code quality as well as trending indicators. It does not only show what is wrong but it also provides quality-management tools, integration to Jenkins and plugins for code quality tools. SonarQube measures code quality against following seven axes:

Potential bugs
Coding rules
Tests
Duplications
Comments
Architecture and design
Complexity

It was an easy decision to try SonarQube out. It surely fit met the requirements well.

4.5.2 Implementing SonarQube analysis for continuous integration

The actual implementing process was slightly heavier compared to many other continuous security controls because a whole new platform actually needed to be installed for code quality measurement. Following steps were needed before SonarQube was successfully installed

- Installing SonarQube (the platform itself)
- Installing SonarQube-runner for MSBuild (tool that gathers analysis report)
- Installing Jenkins plugin for SonarQube
- Configuration Jenkins plugin for SonarQube
- Configuration of SonarQube itself
- Installing plugins to SonarQube (MSBuild, FxCop and ReShaper command line tools)
- Configuration of Resharper command line tool usage

After all the initial plumping was done it was possible to actually configure a build that sent information to SonarQube. The build setup consists of four phases which are illustrated in Figures 8, 9, 10 and 11.

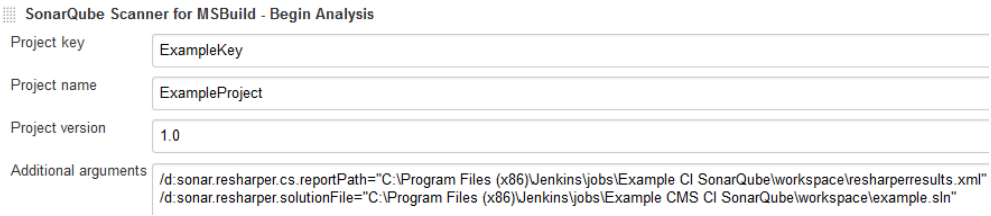

Figure 8 - Setting up SonarQube analysis

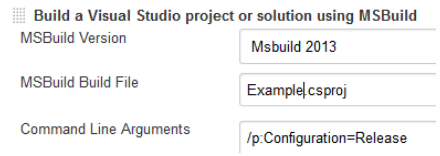

Figure 9- Building the application with MSBuild for SonarQube

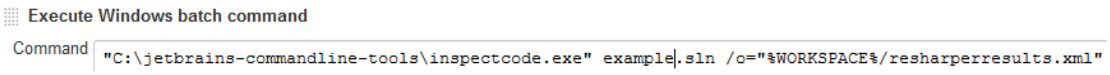

Figure 10 - Running ReSharper command line tools for SonarQube

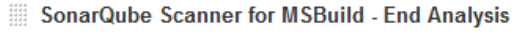

Figure 11 - Telling the SonarQube to finish the analysis

Luckily there was a good blog post about the setup in the internet:

http://www.happiestminds.com/blogs/how-to-set-up-sonar-on-jenkins-for-net-projects/

Before everything was ready for the build one last thing was needed - to disable CSS files checks from ReSharper command line tools. Without this action the build process hanged to the CSS file checks for too long a time.

4.5.3 Using SonarQube in continuous integration

The hard work paid off. The setup was finished and after the build was initiated in Jenkins; a fancy dashboard was shown in SonarQube. (See Figure 12)

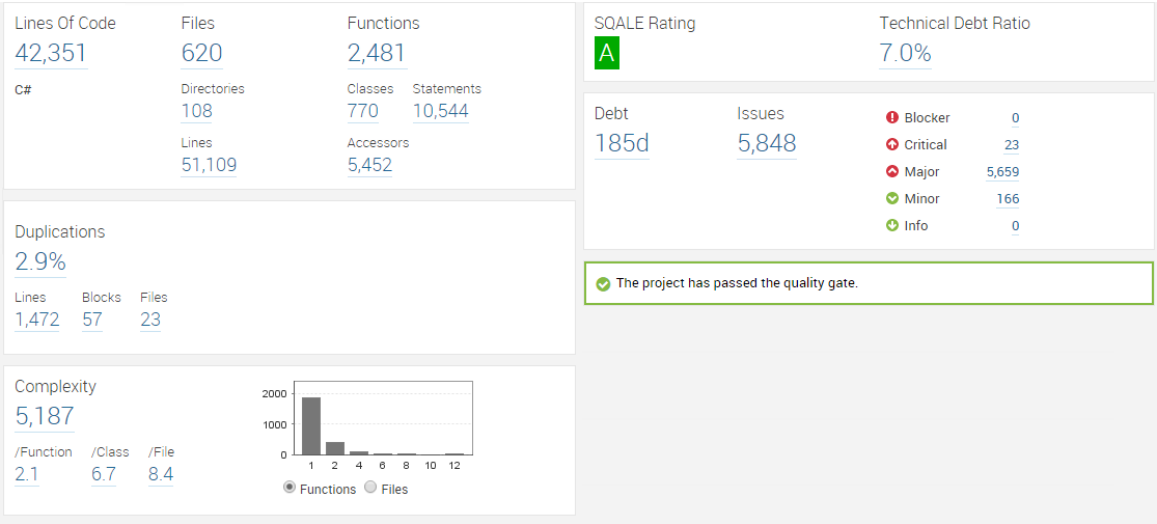

Figure 12 - SonarQube dashboard

Dashboard gives an ability to go deeper into every detail and inspect what it consists of. Technical debt ratio might seem huge in this figure; however it should be remembered that the amount of issues is a sum of three different tools with pretty much everything turned on. There is most likely some fine tuning to be done to cut unwanted results from showing. It is also easy to disagree with the ratings of the issues that the tool has brought up. The initial debt ratio for the project was 0.4% when only SonarQube's own analysis were set on. This means that FxCop and ReSharper command line tools brought a huge deal of issues on the board.

4.6 Calculating code metrics

4.6.1 Choosing the Code Metrics

Code Metrics is a functionality of Visual Studio that let developer to calculate cyclomatic complexity, class coupling, depth of inheritance, lines of code and maintainability of code. The results are shown from root level where user can dig deeper into classes or even functions to see where the complexity is coming from in the solution. Code Metrics is built-in tool so it is quite easy to test, and therefore it was given a try.

4.6.2 Implementing Code Metrics for continuous integration

Implementing Code Metrics into Jenkins was easy. Downloading a plugin and configuring few things was straightforward. After that Jenkins build configuration needed to know which code metrics tool use, what the assembly files are to investigate and where to output the results (See Figure 13 and 14). The results were afterwards given to a report tool which drew a result set to the build instance about its metrics.

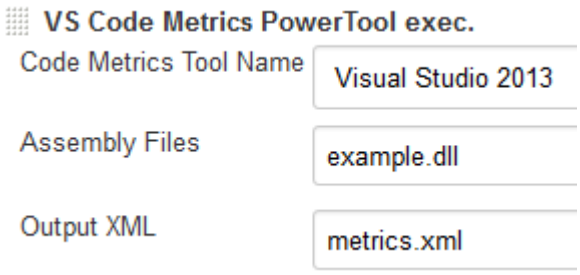

Figure 13 - Configuration of Code Metrics into Jenkins build

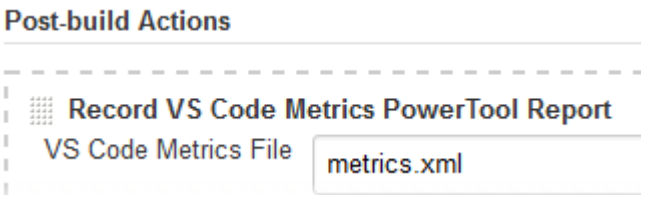

Figure 14 - Setting up the report from the Code Metrics calculation

4.6.3 Using Code Metrics in continuous integration

After setting up the build configuration and building the project a graph and an interactive table appeared on the project. (See Figure 15 and 16)

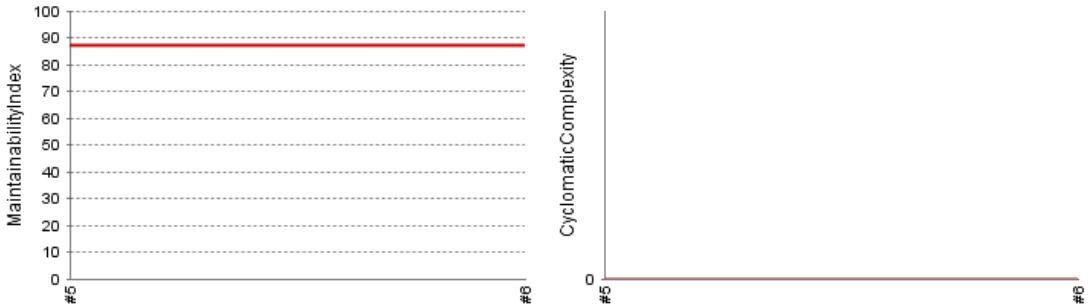

Figure 15 - Code Metrics graph

MaintainabilityIndex	(diff)	CyclomaticComplexity	(diff)	ClassCoupling	(diff)	DepthOfInheritance	(diff)	LinesOfCode
83		48		37		4		57
81		74		68		3		144
100		3		2		1		3
82		15		30		3		27
56		97		109		1		607
92		8		8		6		11
86		110		56		3		200
98		2		2		2		2
90		11		15		2		13
88		83		47		2		125
76		12		63		1		37
67		34		31		1		62
83		75		37		1		119
88		48		28		6		69
74		36		53		3		78
72		4		10		1		12

Figure 16 - Code Metrics result set

4.7 Web application security testing with OWASP Zed Attack Proxy

4.7.1 Choosing the OWASP Zed Attack Proxy

OWASP Zed Attack Proxy is an open-source web application security scanner. Why it was chosen was its JSON API which makes it possible to control the web application security testing from CI as scripts. Choosing a web application security scanner into the project makes it easy to test security vulnerabilities, although security testers should always remember that there is no one tool that suits all purposes but instead there should be multiple tools in a toolbox.

4.7.2 Implementing script for continuous integration

OWASP ZAP presented an API that could be used with scripts in CI; but the script needed to be implemented. It was decided to create it as a PowerShell module due to the synergies with the .NET platform. It is publicly available at GitHub https://github.com/solita/powershell-zap.

The module consists of following aspects:

1. Configure the environment (e.g. where OWASP ZAP is and where to store results)
2. Ensure that OWASP ZAP is running as daemon, if it is not it must be started
3. Set scanning policies for web application security testing
4. Spider the site with normal spider and then with AJAX spider to get the whole attack surface available
5. Make vulnerability scanning against the attack surface
6. Save report
7. Clean up spider and scanning current data
8. Stop the daemon

In the example there are two spidering examples because the example projects were using JavaScript so heavily that normal spidering would not have worked well enough. AJAX spidering is able to execute JavaScript thus follow links that would not be available without JavaScript execution. Scanning is straightforward as in OWASP ZAP current state.

Saving the report was relatively complicated. The Jenkins build needed to have some kind of results in to be shown from example radiator, which means that there was a need to translate the normal scanning results into a format that the CI server would understand. jUnit was chosen as format for the results.

4.7.3 Using OWASP Zed Attack Proxy in continuous integration

There are some points that might make it burdensome to use web application security scanning in continuous integration. First of all user has to be sure that spidering is effective enough; otherwise parts of the application will not be scanned. Another reason is that there might be false positives, and the user will then need to create some kind of blacklist to disable those warnings. The third reason is that the transformation from scan results to unit test results works better for some warnings than for

others. The reason for that is that the most crucial information in scan results might be in a different field depending on the type of the warning.

4.8 Configuration analyses with Microsoft Baseline Security Analyzer

4.8.1 Choosing MBSA

Microsoft Baseline Security Analyzer is able to check the status of operating system updates, Microsoft Data Access Components, Microsoft XML Parser, .NET Framework and SQL Server. It is also able to scan for IIS and SQL administrative vulnerabilities in the configuration. (“How To : Use the Microsoft Baseline Security Analyzer, 2015”).

4.8.2 Implementing configuration analyze with MBSA

During the implementation phase of MBSA an error was faced that stopped testing. MBSA was launched from command line with mbsacli.exe; however it failed every time with “Fatal Error 1 while loading language”, which indicated that it might not be compatible with test setup and it was looked into no further in the study.

4.9 Configuration analyses with Microsoft Baseline Configuration Analyzer

4.9.1 Choosing MBCA

This tool was found in some blog post with links to Microsoft download links. It was really odd that other information about the tool could not be found except the help it had inside itself. It seems that this might have been deprecated long time ago. Yet it was decided to give a try since it was Microsoft tool and it covered configuration analysis which was interesting topic.

4.9.2 Implementing configuration analyze with MBCA

During implementation phase of MBCA a problem was faced that stopped testing. MBCA can be launched from PowerShell by first importing BaselineConfigurationAnalyzer module. After import of BaselineConfigurationAnalyzer module Get-MBCAModel cmdlet can be used for getting IDs for different kind of models. After getting models the scan can be started by piping one or multiple models to Invoke-MBCAModel cmdlet. In the test setup MBCA had not installed any models and thus scanning could not be started.

4.10 Configuration analyses with Attack Surface Analyzer

4.10.1 Choosing ASA

Attack surface analyzer is meant to be used in two phases. At first user should use ASA to get baseline of the computer by scanning it before installations. After initial report is ready a line-of-business application should be installed and a new scan started with attack surface analyzer. The attack surface analyzer can then compare the situation before and after the installation of the software and find out if there are any security risks involved in the product. This software has also command line interface so it can also be added into CI pipeline. It was not used in this study since there were no actual software installations involved in the CI process.

4.11 Security scanning with Nessus

Security scanning with Nessus has been done in other projects in Solita. Since the servers in the example projects were responsibility of third party it was decided that Nessus scanning is not implemented. Initial knowledge also stated that Nessus is not built for web application security testing in a manner that would help developers in their work. It is more of a toolss for checking that the whole environment is running in a safely manner.

4.12 Performance analyze with jMeter

4.12.1 Choosing jMeter

One of the vital perspectives of security was availability. If software has performance issues it is easier for hackers to create a denial of service attack. Thus, the performance analysis can be thought as a security control as well. jMeter is a performance testing tool designed for performance measurements and load test functionalities. It has capabilities of multithreading and simulating simultaneous users in site. It does not do everything that a normal browser does, and thus its results are not absolute truths from end user perspective; however, they are still indicating how well a server is performing when having a multiple request on the site. By default jMeter does not interpret JavaScript, which has to be considered when planning the test, in single page-applications it is mandatory to either plug in a JavaScript interpreter or to call directly those API endpoints that JavaScript would be calling.

4.12.2 Implementing jMeter performance analysis in continuous integration

There is an example implementation for easy jMeter command line usage available in GitHub at https://github.com/dratini/jmeter-perfotrator. This jMeter-perfotrator project shows an example how to easily run jMeter from command line by using pre-configured jmx-file and injecting URLs from a text file for a performance analysis. Running the tool is a one liner (calling the start-run.bat in project, see Figure 17) after setting up urls.txt in the project folder.

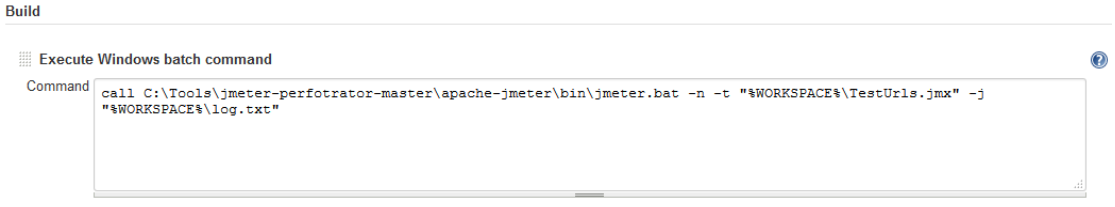

Figure 17 - Example task how to run jMeter-perfotrator in Jenkins

Results can be afterwards shown in Jenkins with Performance-plugin. (See Figure 18 for plugin configuration)

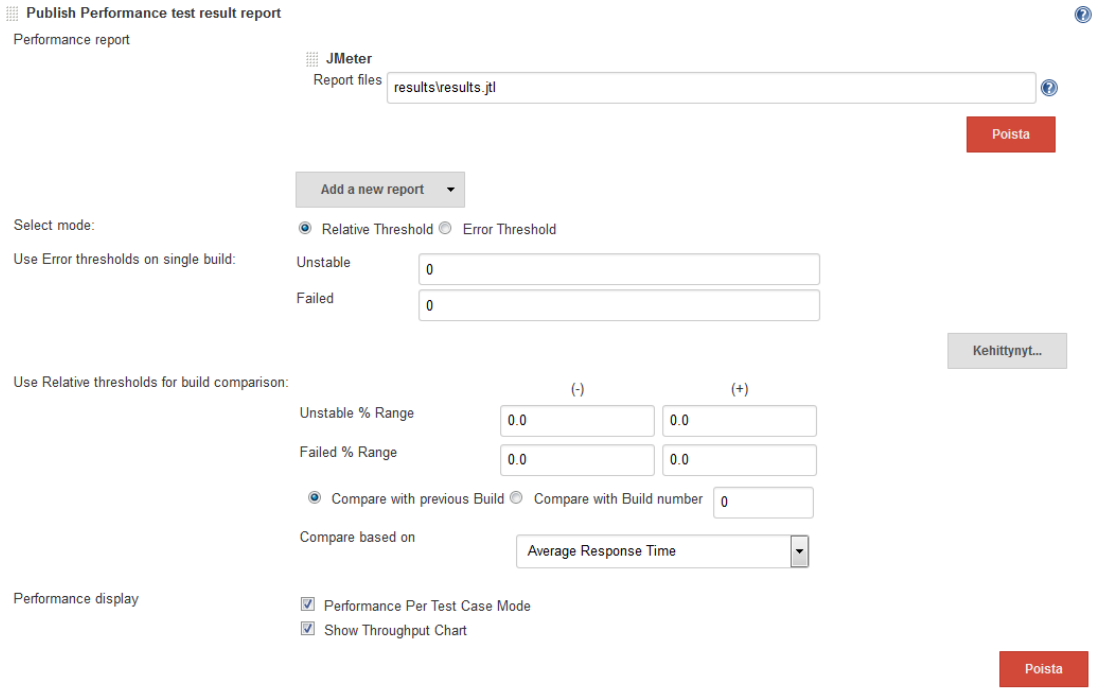

Figure 18 - Jenkins Performance plugin configuration for jMeter

Results.jtl file has been generated by jMeter-perfotrator.

4.12.3 Using jMeter in continuous integration

The plugin shows both trends and the actual data of the performance test results. Below are charts for throughput, responding time and error percentage.

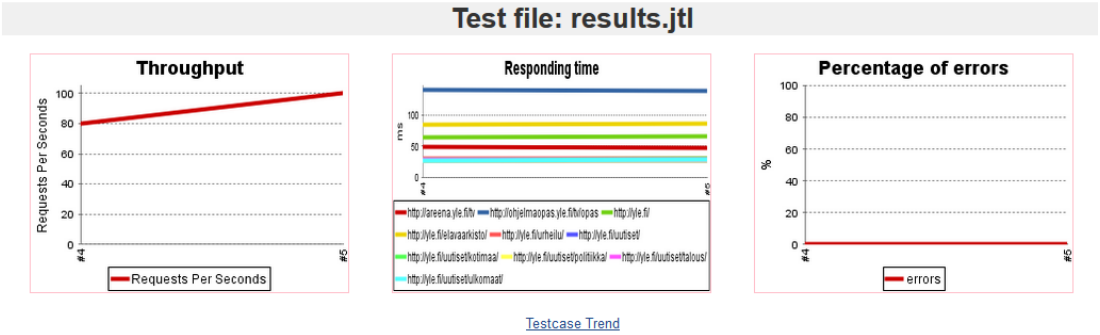

Figure 19 - jMeter trend chart in Jenkins with Performance plugin

5 COMPARISON OF CONTROLS AND RISKS

5.1 Defense in depth onion model for .NET development

Security controls are mapped on the matrix against defense in depth onion model and this was the result.

Table 1. On what layers CI security controls are in defense in depth model

	Network	Host	Application server	Application	Web.config	Source code
FxCop					Y	Y
VisualCodeGrepper					Y	Y
SonarQube			*	*	Y	Y
Code Metrics						Y
OWASP ZAP			Y	Y		
MBSA		Y	Y			
ASA		Y	Y			
Nessus	Y	Y	Y	Y		
jMeter			Y	Y		

* SonarQube can import security scanning results from ThreadFix plugin which in turn is capable fo using multiple different web application scanners

There are few interesting issues on this spreadsheet. One is that there are actually no tools that would combine using an application and reading the code of the application. SonarQube might be able to do that by having separated tools for scanning and static code analysis and having correlation done in third tool. There is clearly an open market area for a tool which would combine these two aspects.

Another interesting issue is that clearly the source code, application server and application itself were the most interesting topics with these tools.

5.2 OWASP TOP 10 threats

OWASP TOP 10 threats (OWASP Top 10 – 2013) were written down and then estimated it was estimated how well the tools would find vulnerabilities from each threat category. The results are indicited in Table 2 as follows:

Table 2. How well CI security control mitigate OWASP TOP 10 threats

	Injection	Broken authentication	XSS	Insecure Direct object reference	Security misconfiguration	Data exposure	Missing function level authorization	CSRF	Known vulnerabilities	Unvalidated redirects
FxCop	1		1	1	1					
VisualCodeGrepper	1		1		1					
SonarQube	1		1	1	1					
Code Metrics										
OWASP ZAP	2	2	2	2	2	1		2	1	2
MBSA					2				2	
ASA						1				
Nessus	1	1	1	1	2	1		1	2	1
jMeter										

empty=no, 1=maybe, 2=meant for that

The first interesting matter that jumped out from the results was that actually web application security scanners face the threats in OWASP TOP 10 best. Another interesting issue was that the code analysis does not directly help with facing threats; they do it indirectly via sanitizing the code. Code quality metrics and performance monitoring were useless controls from OWASP TOP 10 threats perspective. Those did not help at all in facing threats. One more noteworthy fact was that the missing function level authorization was a threat that will not be controlled by any of these security controls.

5.3 Cloud Security Alliances "Notorious nine"

Cloud Security Alliances notorious nine is a less known threat list. It was brought up since cloud computing is trendy in IT business and many are still wondering about starting business in the cloud. CSA's report introduces the common risks in cloud to help companies manage risks in cloud based solutions. Table 3 illustrates how well the introduced security controls do when facing cloud computing threats. (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013)

Table 3 - How well CI security control mitigate the CSA´s Notorious Nine Threats

	Data Breaches	Data Loss	Account or Service Traffic Hijacking	Insecure interfaces and APIs	Denial of Service	Malicious Insiders	Abuse of cloud services	Insufficient Due Diligence	Shared Technology Vulnerabilities
FxCop				1				1	
VisualCodeGrepper				1				1	
SonarQube				1				1	
Code Metrics								1	
OWASP ZAP	1			1				1	
MBSA	1							1	
ASA	1							1	
Nessus	1			1				1	
jMeter					1			1	

empty=no, 1=maybe, 2=meant for that

From the estimation above can be seen that cloud computing top threats are hardly controlled with introduced security controls. All of the tools help with insufficient due diligence. Many help with preventing data breaches and securing interfaces and APIs. Finally, jMeter also found its purpose in the security control list. Its sole pur-

pose is to make users familiar with bottlenecks and capability limitations of applications in performance. Denial of Service attacks in particular want to target heavy operations to bring down applications.

Many of the threats introduced by Cloud Security Alliance are threats that are controlled more with policies and best practices in management than in application level. Detailed explanations how to control these threats can be found in CSA's report (Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013).

5.4 OWASP Testing framework workflow

OWASP Testing framework was introduced in introduction chapter. It is useful to understand how these different tools are linked into testing framework different phases to understand when they can be used.

Table 4. In what phases in OWASP Testing framework the introduced CI security controls are linked to

	Before development	Definition and design	Development	Deployment	Maintenance
FxCop			x		
VisualCodeGrepper			x		
SonarQube			x		
Code Metrics			x		
OWASP ZAP			x	x	x
MBSA				x	x
ASA				x	
Nessus				x	x
jMeter			x	x	x

The table above clearly visualizes that none of these tools will help in initial stages of development where guidelines, policies and architectural decisions are preliminarily

made. In a product´s or project´s lifecycle there is no continuous integration pipeline before the development starts since there is nothing to be built.

All the code analysis tools can be used only in development and deployment; they were present to verify the code quality - nothing less, nothing more, thus they are not present in deployment or maintenance. Checking the quality of the build is a crucial part of the deployment.

Attack Surface Analyzer is a tool that compares the configuration of a machine before and after installation. It is easily thought that it would be only present during deployment; however, actually it can be used also as a health check tool to continuously check that the environment will not change. Microsoft Baseline Security Analyzer can be used in the same manner. There is no use in development for these tools, since they are at their best to verify the quality assurance and production environments.

Nessus, OWASP ZAP and jMeter can be used in continuous security testing perspective to constantly check that no new vulnerabilities are found, and the software runs with the expected performance. OWASP ZAP and jMeter can be useful in the development phase, Nessus most likely will not.

6 PERSONNEL TRAINING

6.1 Plan for educating people

According to Erickson (2008, 5) understanding of programming helps those who exploit, and an understanding of exploitation helps those who program. Hacking is an act where a hacker tries to bypass security in ways programmer never intended the software to be used.

Education is important part of the facing security threats in development. What should be done so that people were willing to use these security controls in their

projects? First they must make to understand the threats that a modern web application will be facing. After that they understand the need and will listen more easily how to mitigate those threats in their projects.

6.2 OWASP TOP 10

From Solita's perspective it is crucial to understand at least the most common threats for web applications. These are mentioned in the OWASP TOP 10 list. Solita has had a long tradition of educating people to know these threats and a continuous action to retrain employees from time to time. During this study it was decided that all new employees are required to go through OWASP TOP 10 training and all personnel should revisit the topic once every two years. This plan will keep these topics on the table and keep Solita's employees alarmed about the ever-changing threats of cyber world.

6.3 CI Security controls in .NET projects

As an another end result of this study a training session was given to .NET developers of Solita about CI security controls in .NET projects to embrace implementing them into projects and to keep developers constantly on guard about cyber threats.

The agenda for the education was following:

- Security model
- Threat modeling
- Hosting and security testing
- Security controls in Continuous integration
- Other tools

In the security model phase confidentiality, integrity, availability, authentication and non-repudiation were introduced alongside onion model of defense in depth.

In threat modeling problem domain was introduced alongside Microsoft Secure development lifecycle. This introduced the personnel how to identify assets, how to model security architecture of the project, how to decompose the application and how to identify, document and rate the threats found.

After thread modeling it was introduced how the onion model of the defense in depth is linked to the projects under study and how hackers plan their attacks against web applications. Also OWASP testing guide 4.0 was introduced.

From the initial theory a transform to actual security controls was made and each of the security controls introduced in this thesis was explained on an equal level that is presented in this thesis.

7 DISCUSSION AND CONCLUSION

7.1 Retrospective questionnaire

After educating the personnel briefly about the security controls studied in this thesis another questionnaire was given to the project teams to reflect the feelings about how the projects would benefit from the new knowledge. The answers to the questionnaire can be found in Appendix 2.

7.2 Main results

The results about the usefulness of different CI tools are presented in the chart below.

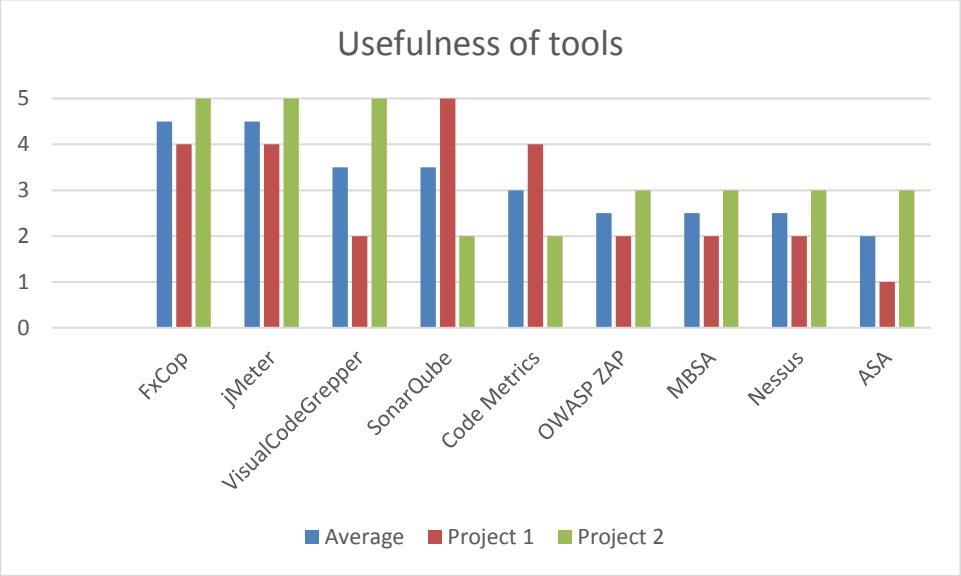

Figure 20 - Usefulness of different tools

It was clear that FxCop was a winner here due to introduction of an easy way to make code quality checks in code. It was a little surprising that the shared first place was taken up by jMeter. After jMeter there were few other static code analysis tools. In the bottom places were all the other tools that especially focused on security concerns. OWASP Zed Attack Proxy, Microsoft Baseline Security Analyzer, Nessus and Microsoft Attack Surface Analyzer got all relatively even scores. The personnel felt that all the tools would give some benefit since the lowest score from zero to five grade was two.

This was positive feedback since it seems that although some tools were rated above others still all the tools were felt to be useful in some points.

Based on open questions there are few things that have to be done more in the projects. Secure development lifecycle has to be brought earlier to projects to make sure that security is thought of in all stages of development, not only from the development to maintenance. Especially threat modeling would be a good addition for developers to truly understand what the software is protected from.

Main result of the study was that having a different kind of security controls in continuous integration is a good idea since they test different things. Software developers in the example projects were more likely to welcome performance and code quality oriented tools than actual security scanners. It is most likely operations that

are more interested in security scanners than software developers. This is where DevOps is needed to build the bridge between operations and development so that there will be no gap in this area at the continuous integration.

7.3 Critics

The first questionnaire was created to be numerical to get as much attention as possible from the projects with the principle “fast to response”. Nevertheless, the amount of responses was depressing. Assumptions about answers were overestimated and a different more detailed approach might have been more generous from the study perspective.

Another fact that might have been better was the timing. It would have been better to make the projects implement all the CI controls – not just the questionnaire with their feelings about them. Lack of time in projects made this hard and it was decided to just educate the personnel and make the questionnaire based on the training material and the education.

Focusing on the Microsoft environment and tools that are as native as possible left out plenty of tools that might or might not have been useful. There were also plenty of commercial products that were left out for the same reasons.

Another issue that was not perfect is that the ReSharper command line tools were added afterwards, and thus they are not included in the results, questionnaire and many of the tables above. It was felt to be useful information still, and thus it was added later on. Nevertheless, it was preliminarily integrated also in SonarQube study.

7.4 Proposal for further research

In the further research there are multiple ways to further study the subject. By leaving Microsoft environment and taking into account commercial products there would be a huge deal of security controls to study. Many of the security products are for Linux environments, and actually there is even a Linux distribution named Kali that focuses on delivering security testing tools to security testers.

More of web application scanners

There are plenty of web application security scanners out there. Many interesting ones were left out. Burp suite and Acunetix, for example, had a costly licensing model. Tools like w3af, Nikto, Wikto and Skipfish might have been eligible but were left out for schedule reasons. OWASP ZAP were the only one present from this category and a further analysis could be conducted with different web application scanners although many of these have been designed to be from Linux machines. Skipfish, for example, needs Cygwin in Windows to work and administrators might not be happy to contaminate their build server with tools like that.

More of vulnerability scanners

Vulnerability scanners like Nessus, QualysGuard, OpenVAS, Nexpose and Nmap are at their best at the network and host level scanning. Since in this scenario that part was nearly entirely left out these tools got only next to nothing attention. There might be more tools for continuous security than continuous integration or continuous delivery. Nevertheless in different scenarios these tools can help out.

Mittn and Gauntlt

Mittn and Gauntlt are both behavior-driven development (BDD) process tools. BDD has emerged from test-driven development (TDD). This whole approach BDD approach would be an interesting study subject as well as its comparison to other approaches.

Configuration analysis with Microsoft Web Application Configuration Analyzer

The tool looked promising; however, lacked a command line tool. It might be possible to integrate this to CI build by either extracting logic from DLLs or by implementing a graphical user interface robot.

REFERENCES

Campbell, G. A. and Papapetrou, P. P. 2013. SonarQube in Action. 1st ed. Greenwich: Manning publications Co.

Duvall, P. M., Matyas, S. and Glover, A. 2007. Continuous integration: improving software quality and reducing risks. Crawsfordsville: R R Donnelley

Erickson, J. 2008. Hacking: the art of exploitation. 2nd ed. San Francisco: No Starch Press

Loukides, M. 2012. What is DevOps? Infrastructure as Code. O'Reilly Media

Meucci, M. and Muller, A. 2014. OWASP Testing guide 4.0

Smart, J. F. 2011. Jenkins: The Definitive Guide, First edition, Sebastopol: O´Reilly Media

Stuttard, D. and Pinto, M. 2011. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition. Indianapolis: John Wiley & Sons.

Cloud Security Alliance The Notorious Nine: Cloud computing Top Threats in 2013. Accessed on (2015, November, 9). Retrieved from https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

Lehtonen, T., Suonsyrjä, S., Kilamo, T. and Mikkonen, T. Defining Metrics for Continuous Delivery and Deployment Pipeline [Abstract]

OWASP Top 10 – 2013. Accessed on (2015, November, 4). Retrieved from http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

How To : Use the Microsoft Baseline Security Analyzer. Accessed on (2015, November, 3). Retrieved from https://msdn.microsoft.com/en-us/library/ff647642.aspx

APPENDICES

APPENDIX 1 – Continuous security quoestionare

Question	Project 1	Project 2
Security overall questions (0-5)		
Has there been security testing in project? (0 = no, 5 = deeply tested by various parties)	0	3
Is there guide for secure development for developers (0 = no, 5 = well documented guide)	0	0
Does all the project developers know OWASP TOP 10 vulnerabilities (0 = no, 5 deep understanding)	1	1
Does all the project developers know counter measures for OWASP TOP 10 vulnerabilities (0 = no, 5 = deep understanding and experience of protecting software)	1	1
Web application security testing questions (y/n)		
Has there been any web application security scanning software in use? (if there has been then what was it?)	n	y
Has the scanning results been studied together?	n	n
Is there any web application security scanning in CI pipeline?	n	n
Has there been any static code analyses in project? (if there has been, then what was it?)	n	y
Is there any static code analysis in CI pipeline?	n	n
Security testing questions (y/n)		
Are the servers on our responsibility?	n	n
Has there been any hardening on our servers?	n/a	n/a
Has there been any security scanning that has verified the hardening?	n/a	n/a
Is security scanning done frequently?	n/a	n/a
Overview questions (0-5)		
How strict requirements there are for security? (0 = internal homepage on my own computer, 5 = bank system)	1	4
What would be grade for projects security? (0 = no security testing, 5 = we know our threats and there are security controls to mitigate them)	3	1
What would be grade for projects security testing automation level? (0 = there is none, 5 = I wouldn't add anything)	0	1
How well project security fits into customers security policy (0 = don't know, 5 = we have went this through and there is no loose ends)	0	4
Feel questions (0-5)		

How much security testing should be considered in your project? (0 = no need, 5 = we are under cyber-attack)	3	4
Would you like to have security controls in CI pipeline in your project (0 = no, 5 = we are under cyber-attack)	0	3

APPENDIX 2 – Retrospective quoestionare

Question	Project 1	Project 2
Security control questions (0-5)		
How useful would FxCop be in CI pipeline in your project?	4	5
How useful would VCG be in CI pipeline in your project?	2	5
How useful would SonarQube be in CI pipeline in your project?	5	2
How useful would Code Metrics be in CI pipeline in your project?	4	2
How useful would OWASP ZAP be in CI pipeline in your project?	2	3
How useful would MBSA be in CI pipeline in your project?	2	3
How useful would ASA be in CI pipeline in your project?	1	3
How useful would Nessus be in CI pipeline in your project?	2	3
How useful would jMeter be in CI pipeline in your project?	4	5
Education questions (0-5)		
Did you learn new things in thesis presentation?	4	1
Do you understand threat modeling?	4	4
How useful it is to have constant OWASP TOP 10 education?	4	5
Do you understand defense in depth?	0	3
How important you think is to implement these controls to CI?	2	3

Open questions – Project 1

What was useful in the education materia?

Compiling and explaining the different testing tools was very useful. Security threat model isn't anything that special, but good to keep in mind and refresh. Visualizing it is good.

How did the material change your thinking about security testing?

That we should do it more, if we can convince the customer to pay for it 🙂

How would you change security testing in your current project?

I think our assets should analyzed. We should at least try FxCop and those code complexity tools (even if they're not very much related to security IMO)

How did the material change your thinking about threats your project is meeting?

That with some small efforts there can be significant benefit.

What would you do differently if you were to start a new project now?

Depends on the project really. All of these tools can already be used in existing project.

What would you like to know more about?

You said SonarQube is pretty complicated to configure, so I hope there will be more tutorial about this.

Open questions – Project 2

What was useful in the education material?

It provided valuable training to Solita, which is also incidentally the first part of executing what is the Microsoft Security Development Lifecycle. https://www.microsoft.com/en-us/sdl/

I would like to see Solita establish a baseline secure way of software development, so that security isn't first thought about when the software is already halfway done.

I would follow the Microsoft SDL process closely to achieve this, and I think the presentation looks to be a good starting point towards that goal.

How did the material change your thinking about security testing?

It didn’t, I was already familiar with the concepts provided in the material and have experience with many of the tools mentioned. It reminded me that we are so incredibly lackluster in this aspect as a team.

How would you change security testing in your current project?

I would like to have established a CI environment which I could trust to check my implementation for most of the common points of security and code analysis. I would also have liked to see a clear set of requirements in terms of security for the project and an analysis on what are the most security critical aspects of the application.

How did the material change your thinking about threats your project is meeting?

I wouldn’t say it changed much, but more reminded me that customers never want to pay for security – it is something that must be provided by us to every project de facto, and it should be part of the original price. It isn’t an afterthought but it must be built in. The customer’s business doesn’t rely on it, but ours does because if anything goes wrong – we will be blamed and will stand to lose the most if that happens.

What would you do differently if you were told to start a new project now?

Keeping strictly on the topic of security, I would make sure to actively have a set of tools to test the solution in the CI pipeline. I would also be louder about the security implications of architectural decisions that were made incorrectly or with callous disregard to security.

What would you like to know more about?

Attack vectors are always useful to know about, because they actively make a programmer more able to see the implications of their implementation.