National College of Ireland

Gaps and Improvements in Secure Development – In Practice

MSc Project Report
Programme Name MSCCYBETOP

Nassir Hussain
Student ID: x19223676

School of Computing
National College of Ireland

Supervisor: Dr. Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Nassir Hussain

Student ID: …… x19223676 ……..……

Programme: …… MSCCYBETOP …… **Year:** ……2021…..

Module: …… MSc Research Project/Internship ……..……

Supervisor: ……Dr Imran Khan ……..……

Submission Due Date: ……

Project Title: …… Gaps and Improvements in Secure Development – In Practice ……..……

……

Word Count: **Page 23 Count 5816** ……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Gaps and Improvements in Secure Development – In Practice

Nassir Hussain
X19223676

Abstract

Organisations understand that cybersecurity is a critical issue to address to protect their interests. Data breaches, data leaks and hacks are the events that threaten most organisations. There are challenges to address these issues. Modern Software Development Lifecycles (SDLCs) emphasise the developing of code and getting it deployed to production for business needs. There is little emphasis and importance placed on the security aspect of SDLC. Agile and DevOps teams may or may not address security issues as part of their SDLC. In this research proposal, I will outline the importance of a software security champion being embedded into the SDLC. The criticality of having secure software led SDLC where security should be embedded into whichever flavour of SDLC that is chosen. The inclusion of the security champion into SDLC will help bridge contextual gaps in security frameworks such as OWASP and SSDF.

The literature and underlying research highlights gaps are from practical implementation and theoretical understanding of secure coding. What does this mean? This means that context plays a huge role in the security driven concerns within an organisation. The research will look at the implementation of a specific security toolkit for an organisation using .Net web applications. Context is important is because it refers to architecture, software patterns, software configuration and software implementation understanding. Practical secure development is difficult as it needs understanding across the tenets of software programming, architectural and security practices. While the artefact developed is designed to address security risks in the continuous integration / continuous deployment (CI/CD) pipeline, it can be used as a standalone tool should this be required, however this is less effective as it may not be integrated into the software development life cycle (SDLC). The security toolkit (CLI tool) will address content security policy, X-frame security options, X-Content type options and X-XSS-Protection. The CLI will be built using C# in .Net Core Framework and will allow for security issues to be addressed as part of the SDLC. Automated and building the fixes into the SDLC will be a cost save and will not require a penetration test and subsequent remediation to fix the issues.

Keywords—cyber, cybersecurity, programming, Agile, DevOps, OWASP, SSDF.

1 Introduction

In this report, I will focus on the research in the topic of cybersecurity and software development, this will then extend into the building of an artefact to address security issues within a security focussed organisation.

Cybersecurity incidents and hacks are broad terms that can cover a variety of security policies being bypassed. Cybersecurity incidents take place on physical assets or software assets. In this research proposal, the focus will be on cybersecurity incidents that pertain to software. Software is used by organisations and public bodies to process a wide variety of tasks and business needs. Modern software is the driver of modern work practices and as such software is ubiquitous. The ubiquitous nature of software means that software hacking is seen as a tool for bad actors to extract funds from an organisation, cause reputational damage or to be a nuisance. The Panama papers hack (O'Donovan, et al., 2019) illustrates the impact of a cybersecurity incident, it is a case of reputational loss, data leak and financial impact. A general trend has seen an increasing amount of security hacks and data breaches take place in the last 30 years with no end in sight (Rice, 2007). Recently, the HSE[1] cyber-attack shows that critical technology assets need to be protected, as of writing this, all technology assets have not been restored.

Organisations have a better understanding that secure software is a business need for their organisations (Daley, 2016). There is an understanding that secure software is vital to keep assets, data, and funds safe from bad actors. What is secure software? A simple example of webpage security would be the enforcement of HTTPS (secure web traffic) over the internet, a further case could be made for never allowing unsecured internet traffic to a site (HTTP). How is software kept secure? Software is kept secure by doing manual testing, penetration testing (Shebli & Beheshti, 2018) and code reviews. What are the actions required by those in charge of software to keep it secure? This question has led to multiple frameworks and best practices being created. The frameworks and best practices guide those who implement software and want it to be secure. Secure Software Development Framework (SSDF) is a software development framework that has been created by National Institute of Standards and Technology (NIST), the goals of NIST are to prepare and advise an organisation to develop secure software. SSDF would be considered as an organisation level initiative to help embed secure software practices while building software.

Secure code is in focus for organisations to mitigate and proactively guard against cyber incidents against their assets. Secure coding is seen as best in practice for organisations to implement to reduce the cybersecurity threat from bad actors or hackers. Software developers are primarily concerned with building applications to focus on a business need and the result. Software developers are not trained in understanding the security concerns of applications.

Secure software code development is a complicated and multifaceted undertaken (McGraw & Viega, 2002). It cuts across two concerns: software development and cybersecurity. Modern software development has progressed so much in terms of tooling, support and understanding. The seminal paper "The Errors of Tex" (Knuth, 1989) shows a deep understanding for the process around software development and the pitfalls that can and will

[1] https://www.hse.ie/eng/services/news/media/pressrel/hse-cyber-security-incident.html

be encountered. While this paper does not highlight or indicate issues around secure development, it does exist even if not highlighted. Knuth (1989) illustrates software development and specifically the research of software development is a mature topic whereas cybersecurity research is in its relevant infancy compared to software development. However modern software development books (Graff & van Wyk, 2003) (McGraw & Viega, 2002) , research, and online articles have made vast improvements to improve secure development processes. The biggest crux with secure development is the mastery required across multiple domains: software engineering, software design, software architecture, cybersecurity, and secure software development.

This has led to development of best practices such as OWASP (Anantharaman & Wukkadada, 2020) to assist in this complex process. OWASP provides practical guidelines and uses cases that can be implemented by software developers to fix common security issues. OWASP helps and assists software developers to improve cybersecurity within software applications. In terms of the artefact the OWASP will help guide the use cases, but extra understanding and research is required to understand the entire implication of changes being applied.

The artefact that is implemented as part of the report will be used by an organisation that has modified their software development lifecycle to include elements of cybersecurity throughout it. The organisation uses Agile SDLC lifecycle and does not implement the use of an explicit framework such as SSDF or another NIST framework. The organisation does however use application security specialists that are embedded as part of the SDLC (Oyetoyan, et al., 2016). The artefact developed is part of security toolbox to address gaps, issues and improve security in the SDLC, the main artefact will be embedded into the continuous integration / continuous deployment (CI/CD) pipeline and will work in automated fashion to fix security issues that get raised during penetration testing. The cost associated with identifying and securing these issues in test, User Accepting Testing (UAT) and security testing is several times of remediating issues in the earlier stages of SDLC. The business case and use for this tool is to fix common issues earlier which is cheaper and embedding better security practices in SDLC in an automated manner.

The security toolkit is built in a modular manner that allows for flexibility, extensibility, and reuse. Building in these features allows for future proofing and extending the security toolkit as required when other enhancements and features require. The initial security toolkit will address 7 issues and is a command line interface (CLI) tool.

The practical implementation of this artefact will investigate and look at critical success factors required to improving software security at a practical level and what / how organisations can do build in better practices.

2 Related Work

The research literature can be broken down into the following themes:

2.1 Why does insecure software exist?

It is critical to understand the landscape of software development with respect to cost, comprehension and what secure is. The pragmatic view in the software development

community is that organisations want secure software however Daley (2016) has a unique perspective. Daley proposes that securing software is a cost benefit analysis exercise and may not aligned to the perception that software should be secure. From a cyncical point of view it's understandable that the cost of adding extra security features will cost more and in turn will increase the cost of software. While, there is some merit in this argument we have seen a shift in the industry towards building more secure tools (Zhu, et al., 2014), improved seurity tooling (Graff & van Wyk, 2003) and security design patterns (Bunke, 2015). The other view in terms of insecure software is lack of training (Wu, et al., 2013), ability to teach secure programming (Yu, et al., 2011) or understanding how security fits into current processes (Jaatun, et al., 2017). Secure development is a nuanced topic and requires analysis across multiple areas to understand the requirements of building a practical secure programming environment.

2.2 Frameworks available and best practices

Software development and secure software development should leverage best practices, frameworks, and guidance. The two main areas of note in terms of secure software development are NIST, SSDF and OWASP. NIST leads the way in providing guidance and frameworks across several areas including cybersecurity.

SSDF is designed and developed by NIST and is used to govern software security in an organisation (Dodson, et al., 2020). As SSDF has a role in organisation governance, the practicalities from a secure software perspective are to have checks in place, provide high level guidance for implementing security in software and instructing an organisation to have a plan in place. While these are useful from an organisational perspective and to get funding at program / project level from a practical perspective it does not assist the developers in their day to day to work but ultimately, it is not meant to.

OWASP was designed and developed to provide practical solutions to problems, and it does this well for a specific set of use cases (Devi & Kumar, 2020). OWASP provides a set of use cases but if not understood with sufficient knowledge by software security developer they may be implemented poorly or missed (Wang, 2018). Zhu (et al., 2014) look at the use of OWASP to build tools (static analysis real-time) feedback to developers and assist to fix security issues, this will work in scenarios where the solution is perscriptive and the developer understands the issue. The issue with this approach is that you require the developer to understand this could be a false positive or how to use the tool to fix the issue. Static analysis may not be able to identify the issue, this is the biggest issue with static analysis tools they are only as good as well as those who developed / designed them. Use of these tools also allows for delegation of security fixes to tools identifying issues as opposed to systematic checks in SDLC. OWASP is used by Anantharaman & Wukkadada (2020) and Devi & Kumar (2020) in a different way, by using OWASP tools to identify issues and as testing tools, while this is commendable to find issues, this won't fix issues. Developers are unlikely to perform the secure testing of items as it is usually out of their remit. Are testers versed well enough in these tools? The ideal scenario would be for developers to be able to identify or fix issues before they arise or a specialist on team does this work.

The other criticisms that can be laid against SSDF and OWASP, is the lack of integration of either with SDLC such as DevOps and Agile. How does SSDF integrate with Agile development? Where does OWASP fit in with Agile?

2.3 SDLC (DevOps / Agile) usage

Every software development project uses a SDLC methodology, historically waterfall was used, DevOps / Agile are SDLC frameworks that are in vogue. For secure software to be implemented, we need to understand what SDLC is in use and what provisions are in place for the security elements. DevOps (Leite, et al., 2019) outlined in figure 1 below, illustrates a loop and feedback mechanism between development phases and operation phases too. This approach to software development allows closer coordination between the implementers of software and the users (operations (Ops)). The idea of closer relationships between developers and users will produce better software that is more aligned with business needs.

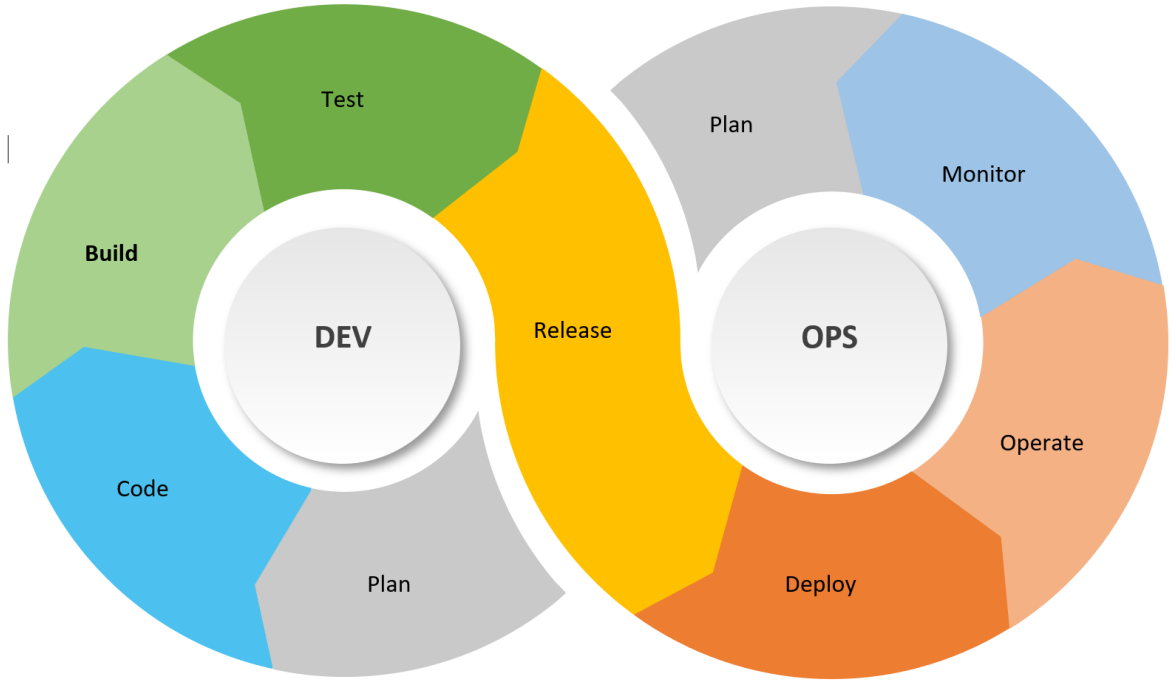

Figure 1: DevOps

Agile development (Rauf & AlGhafees, 2015) as outlined in figure 2 below follows a similar cyclic process as DevOps. Agile is based around the concept of having shorter development cycles of 1-4 weeks with delivery of functionality, this prioritises business needs. The feedback loop mechanism also helps to prioritise critical items but again this would be business needs as opposed to security needs in general.

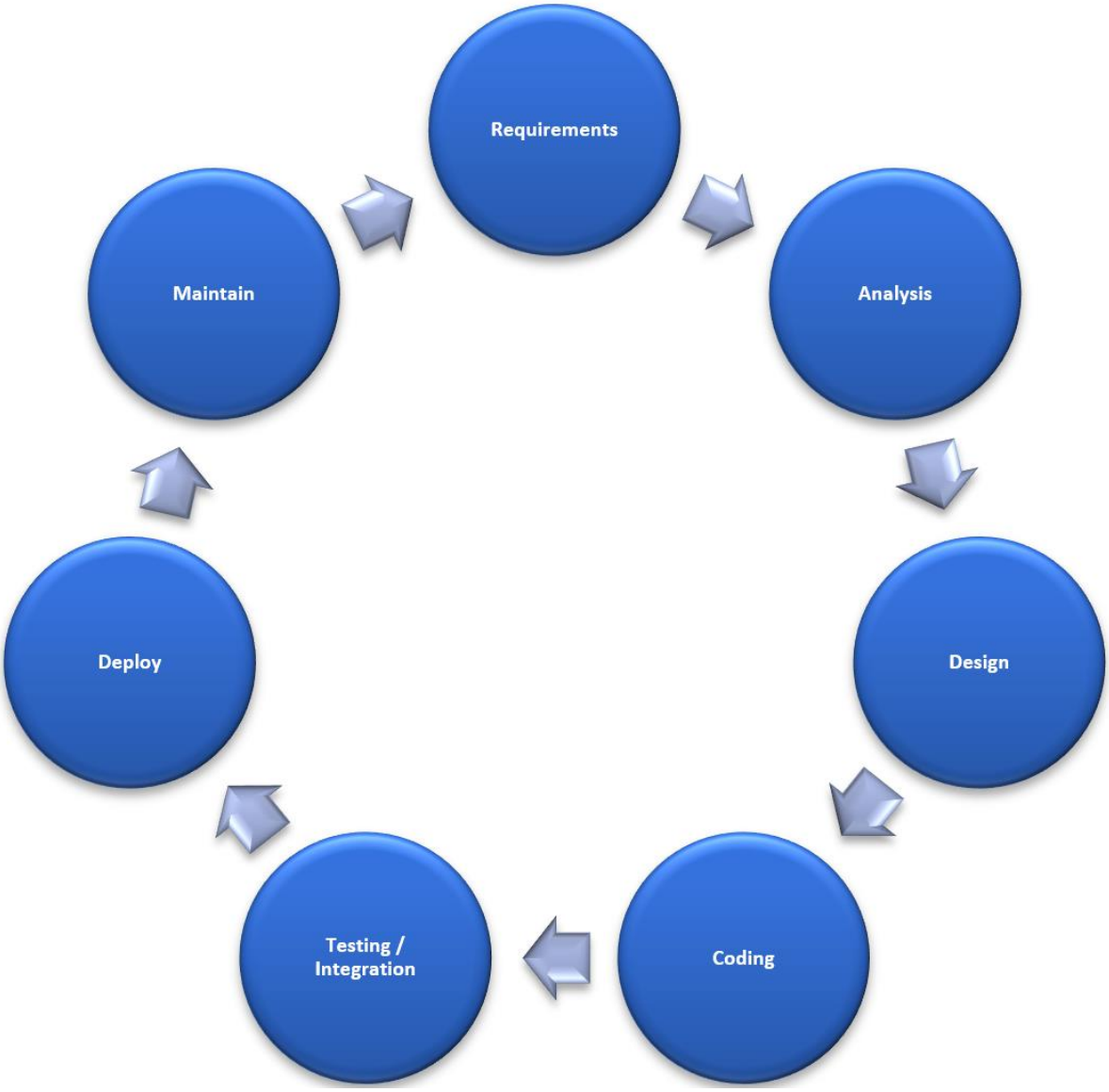

Figure 2: Agile

Both Agile and DevOps are specifically designed for building and deploying features for business needs in a rapid and predictable manner, neither explicitly factor in security or security testing. Testing in both refers to business which may or may not include security. Software development teams are building software for a business need unless they are developing a security product, where security is to the fore of development concern. Why am I discussing the use of SDLC frameworks in the context of secure development, secure development that will take place in a SDLC framework? Well, it must be understood and taken into consideration. Organisations need to make decisions how to customise and embed secure development into a SDLC. This leads to customisation of frameworks.

2.4 Customise SDLC

Having looked at frameworks and investigated their role in software development, we need to understand the customisation of them. As Agile is used by the organisation, I will investigate

the customising of Agile to fit secure development requirements. Bartsch (2011) investigated the subject, the outcomes were the inclusion of stakeholders to get buy in to allow for security to be factored in Agile (in particular during sprints and epics). The study was limited and didn't lead to any firm conclusions about practical implementations but outlined that secure development is possible but more work is required. ben Othmane, et al. (2014) does a deeper dive into the extension of Agile and while it's possible it may not scale as well as the generic Agile model, again this means extra cost for an organisation to implement secure development in terms of their SDLC. Security in Agile would have to be factored into every process, in essence which would also increase the SDLC complextiy as well. The research also suggests that a specialist role is required to implement secure development.

2.5 Specialised Role (Security Champion)

Secure development research has identified that secure development should be undertaken by a specialist role. Wang (2018) highlights the importance of an experience in pair-programming. Secure development needs to be viewed as specialised task that requires deep understanding. Certain organisations train and develop software developers into a security champion which is a specialised role to help fix security issues. The software developer is already embedded in the day to day work of the developing software for an organisaton, the proposal that a certain trained developer take additional roles in this case security makes sense from a pratical standpoint. The developer has domain knowledge, product knoweledge and may understand the architecture too. This means that a critical understanding of security development is required. For the individual this means training, upskilling and better understanding of software development. For the organisation this means an improved workforce that have skills to remediate and fix their own security issues. This however does not exclude the need of external in certain instances where the organisation is lacking from a security standpoint. Well rounded developers with the ability to fix and remediate issues is also a costing as it occurs in the earlier stages of SDLC.

2.6 Teaching and Training

Throughout the research, the themes of training and teaching are repeated. It is clear from the research and in particular Yu, et al., (2011) which outlines that teaching security code should be taking as serious as teaching software development, they consider that teaching and lecturing in college should approach security development akin to programinng 101. While this approach does have merits, the study doesn't have longitudinal results to back up or quantify the improvement to a level that would considered efficient. Their approach shouldn't be disregarded on this basis though. Gasiba & Lechner (2019) illustrate that training, awareness and teaching are all required to improve the quality of code and the aspects of security, however, there isn't a definitive framework or guidance to implement this. It needs to be considered and implemented. ben Othmane, et al. (2014) does a comparative study between two organisations and finds that training is required across all people in the SDLC, it's important that there is awareness amongst an organisation of the criticality of security. The study also highlights that tools like static analysis can assist but on their own may not perform

as expected, teaching and training are required and is essential, there are no shortcuts to security implementation.

2.7 Takeaways

The points above illustrate the multifaceted nature of embedding security in an SDLC; there is not one simple solution or approach that can be applied due to the varied and agile nature of SDLCs in use. There is no silver bullet or singular framework that will fix all your security woes, security is nuanced even before we delve into out scope items such as architectural design and architectural flaws. Why is software insecure? Software is insecure because it is complex and requires multiple items to be addressed from an organisational standpoint all the way to developers implementing code. Frameworks like SSDF provide little practical guidance to those who develop code. OWASP provides practical guidance but may be ineffective without sufficient context and understanding. Agile/DevOps requires customisation and may no longer scale with security embedded. The sufficient skills and understanding may be lacking, and the specialised role of security developers may not be understood or exist an organisation. The multitude of items need to be addressed before secure code is achievable, this highly complex and may not be easy to implement.

3 Research Methodology

This topic was derived from addressing use cases that were encountered in an industry scenario. Modern SDLC embrace rapid development and deployment via Agile or Scrum. The industry scenario occurs after the deployment of web application to a User Accepting Testing (UAT) environment, at this point a penetration test scan occurs. The results and scheduling can take several weeks to several months depending on the penetration test and getting access to the environment. Typically, the penetration may take several months after the sprint it was created in. This has several impacts; security issues may not be identified for several months after issues occur. The web application may be vulnerable until issues are identified and then remediated. We have rapid development and deployment, but security seems to be less rapid.

Over a period, it was noticed that common security issues were not fixed in an automated fashion, despite using CI/CD toolset. CI/CD is identified as a key area that should be used to help build in security features. In section 6 of this report, I outline the 7 evaluation items and the approach to fixing them. SDLC uses CI/CD, and the fixes will be a part of this critical process in SDLC.

After completing a literature review in section 2, the approach in conjunction with industry was to address the specifics issues that affect the company. This means providing a systematic, automated approach that will fit into the current SDLC process using by the organisation. The literature review does have some practical uses of the research and I was guided by it to build a proof of concept that addresses specific security issues to assist the needs of organisation. There are two reasons to address this financial and to improve security.

Systematic and automated approach is key to the deliverable, the organisation wants to automate security and ensure it is implemented. Modern SDLC use tools such as CI/CD to build and deploy applications, embedding the artefact into this process makes the security updates form part of the in-place build and deployment processes that are used. This approach allows any SDLC to leverage the use of the artefact once the SDLC uses automated

builds and automated deployments. This removes the need for manual checks or code not the security framework features not implemented as part of SDLC. It is imperative that security fixes are a part of SDLC that can't be bypassed or else this defeats the security purposes. The CLI as outlined in section 4 will be implemented in automated and systematic approach. This approach allows for this tool to be rolled out organisation wide and in other business units. Security hardening is vital for any critical systems that are built and maintained by developers. Building in security focused processes helps to drive the team to build security as part of processes being used.

The report proposal illustrates the key elements of the literature review, the background the approach, my tool, the reasons for it being built and the parts to be considered. A CLI tool was made to address issues in the organisation and based on the research undertaken to fit the needs of the organisation. A CLI tool makes sense for the organisation as it can be integrated with ease into the processes that exist in the organisation. The use of CLI tool allows for it to be transported to different environments such as local developer machines and to build servers as well. This portability is a benefit as the CLI artefact and can be used in any environment that is required. It is built in .Net Core code which means the tool is compatible with any machine that has .Net Core code installed, that is the only pre-requisite. In the following specification section, I'll illustrate where the CLI tool will fit into the CI/CD process of an organisation without changing their processes.

4 Design Specification

Current Model

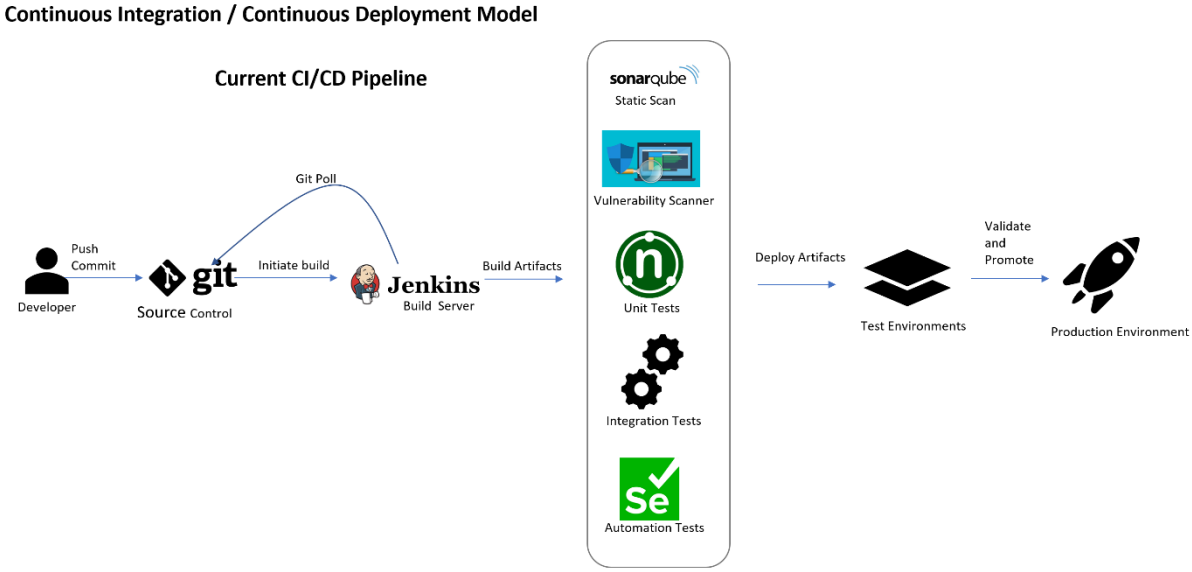

Figure 3: CI/CD Pipeline

The current Continuous Integration (CI) / Continuous Deployment (CD) model is outlined in the diagram above. The sequential flow describes from developer commit all the way to deployment to test environments and to production.

1. Developer commits to source control (git repository), the git repository is a central storage for all code that developers have access to. The code can be modified on a local

developer machine and pushed to the server via a review mechanism known as pull requests where the code is peer reviewed.

2. This commit will initiate a build once the pull request in git has been approved and the commit gets integrated into the master build of git repository.
3. A build server using a tool such as Jenkins will clean, build, compile code and publish artefacts. Jenkins can be considered the orchestration tool to manage the CI/CD process.
4. Artefacts will be scanned by sonarqube (static scan) and vulnerability scan in parallel. Static scans perform a scan to undercover security defects against known issues and highlights them. A vulnerability scan will identify issues with insecure components or third part library.
5. Other tasks such as unit tests, integration tests and automation tests (Selenium) will take place. Whilst these tasks may not necessarily perform a security function, this is to illustrate the completeness of the scan.
6. Once the scans have been completed the artefacts get deployed to the test environment. In the current process, we push to testing without any additional checks for our specific security needs.
7. Validation and User Accepting Testing (UAT) takes place and then deployed to production. A penetration test is completed as part of UAT; however, it may take several weeks to complete and impact the deployment to production. This is late in the process.

The steps for the CI/CD are linear until the point of scanning artefacts, multiple scans in run parallel to reduce the overall duration of the build process. This is a setup that is used by this specific organisation but can customised to the suit the needs of any business. CI/CD processes vary by organisation but is critical in terms of their build and scans to be performed. Automated CI/CD provides consistency which is critical in building software. Best fit in one environment may not be in another. Additional tools can be added for dynamic scanning or any other needs. I have illustrated the current tools being used as I'm improving this process.

The proposed model shows the addition of custom security tools step which is a command line interface (CLI) tool. The is additional step will be the artefact that will be built and outlined in this report. The tool is used after the build artefacts step as it will need the compiled, build and published artefacts before it can be used. The CI/CD model has this step before any other scans, again the preceding needs to finish before scans can run. While the CLI is being used as part of CI/CD the CLI can be used a standalone tool as well. Standalone use allows for other use cases on developer's machines to be catered for other unseen use cases that I have may have not identified. The main use case is to build the extra security features as part of the build process to embed this extra step into the Software Development Lifecycle (SDLC).

Proposed Model

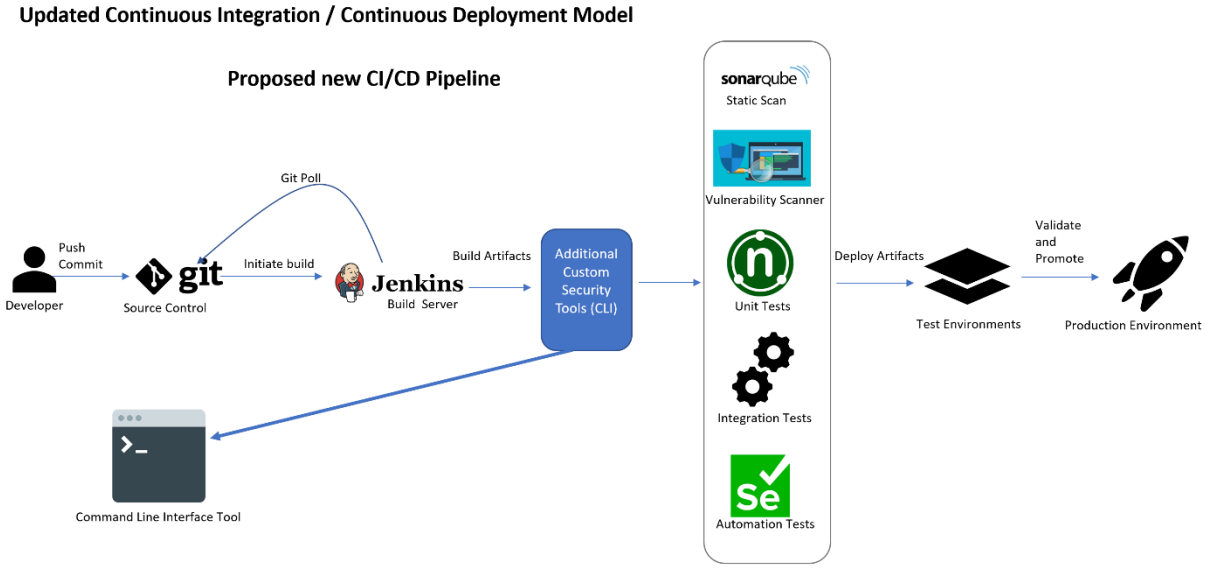

Figure 4: CI/CD Pipeline

CLI

The CLI is a windows executable built using .Net Core 3.1 which is build using Microsoft Visual Studio. The tool being a CLI allows for the flexibility of parametrised executable to change or modify execution at runtime, this allows for overriding default behaviour, path(s) to search and other items that will be outlined in configuration manual.

The CLI is tool is step occurs after the artefacts are built and before the UAT and subsequent scanning takes place. Artefacts will be deployed for testing and need to be checked by the scans we have in place which will be a penetration test which will occur UAT phase. This is ideal point to inject the CLI tool in the CI/CD process plus it will illicit feedback earlier in the SDLC process.

The limitation of my approach is that model and artefact has been built and designed for a specific organisation, the remit was to address issues within the organisation to improve security. The SDLC being used is Agile and the toolset being used is Microsoft .Net web application frameworks. The CLI tool will only address Microsoft .Net security issues and is not web framework agnostic. It does leverage the findings of OWASP and penetration tests to fix issues but they CLI tool will only work on Microsoft web applications which is the domain of interest. The CLI tool has no configuration editor or UI tool to assist with customising the fixes as they may be a requirement in the future, another update would be required to address this. It is design to handle a specific use case and not every use case on every platform where it could exist. The fixes are limited in scope.

5 Implementation

Implementation Overview

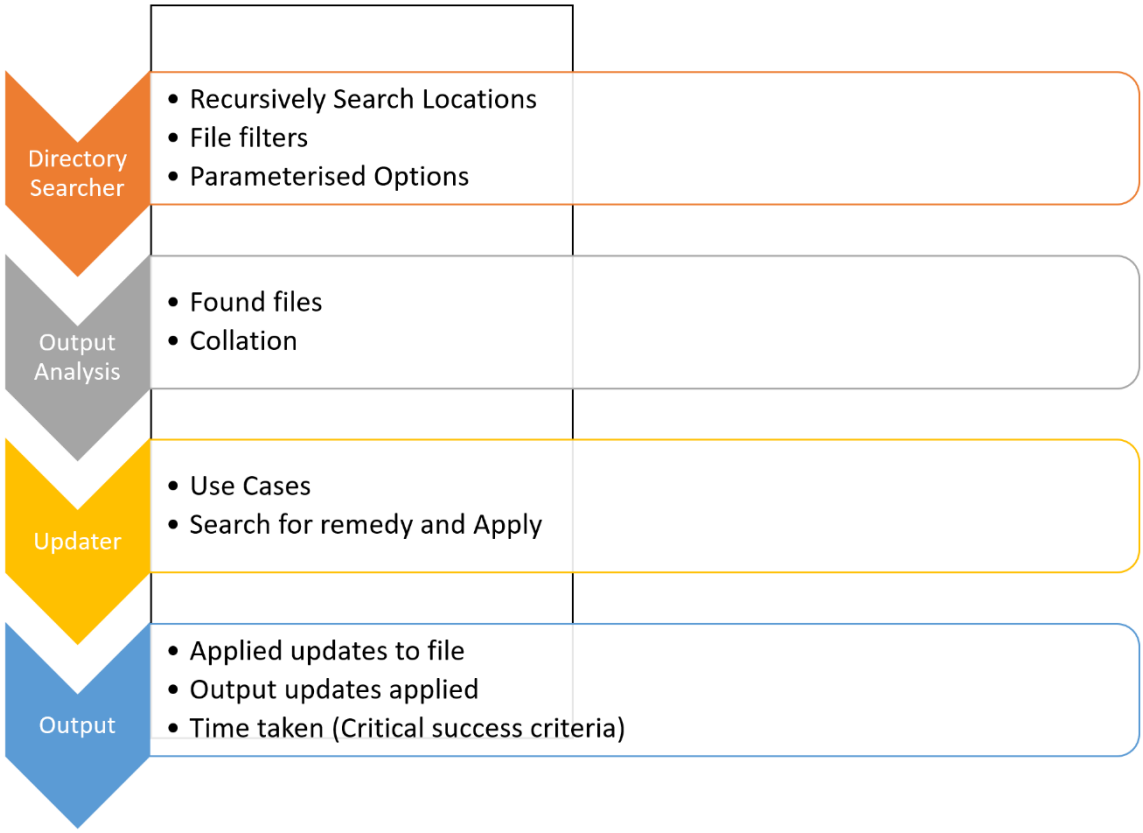

Figure 5: Artefact Overview

Proposed

Microsoft .Net Framework Web Applications

The CLI tool is used to analyse source code directory on a recursive basis to find any web.config files that exist that are hosting web applications. It will scan the web.config to identify if the security controls around content security policy, X-Frame options, X-Content-Type-Options and X-XSS-Protection are implement in a secure manner that respect the policies according to both best in practice as well as organisational policies. If the security policies are not in place, the CLI tool will transform the configs to align with policy used by developer.

Proposed but out of scope
Microsoft .Net Core Web Applications

While the primary focus will be to fix web applications using .Net Framework, the next logical step would be for a tool to implement the same features above but for .Net Core web applications which is the latest framework from Microsoft.

Use Cases

Use Case 1 – File Searcher

The file searcher use case is a critical component in my artefact, it will be built in a modular fashion that will allow reuse and flexibility of use. The reason for modular design allows for the file searcher to be used in this artefact but can be extended in the future. The flexibility is critical as I need to have the ability to search for multiple files and not just hardcoded files to allow for this flexibility feature. Flexibility will allow for build once but re-use multiple times in many future applications or yet unforeseen applications. The apply rule's function will be further broken down by use case 2 Rules engine. The rest of use case requires the ability to update the files to ensure the security features required are applied. For initial proof of concept (POC) and design I have used 7 updates to prove out my feature set.

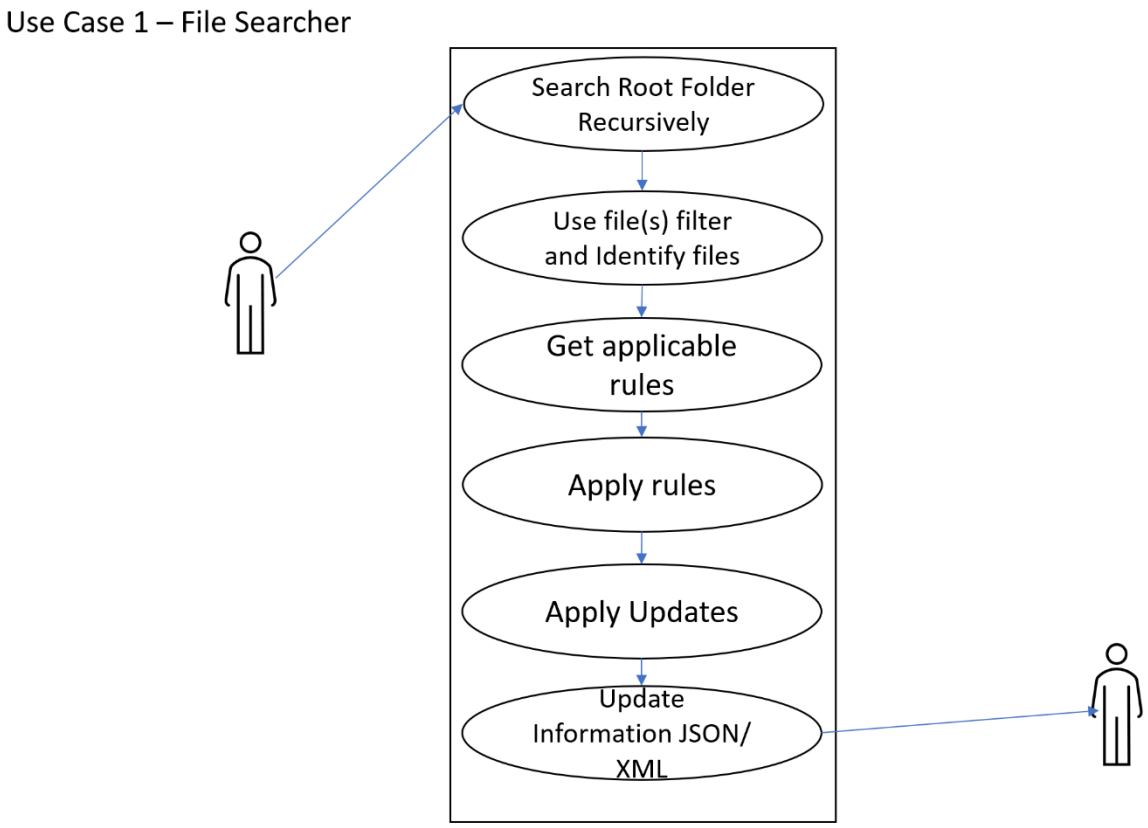

Figure 6: Use Case 1 – File Searcher

Use Case 2 – Rules Engine

The rule engine is the other critical feature as it allows for multiple use cases to be catered for and allows for the ability to reuse and extend the code for feature scenarios as well as provide a flexible and rich feature set for my initial artefact development. A rudimentary rule matching and verification has been implemented as this an Agile based development where

the tool will be production ready with less viable product in 2-3 sprints. Modularity and flexibility are critical for
design and implementation considerations as this product / toolset will grow as more scenarios are encountered. I have focussed on 7 scenarios for initial POC and implementation. The rules engine allows for the catering as many rules as required, I have addressed 7 issues, but this could be as many as the developer builds.

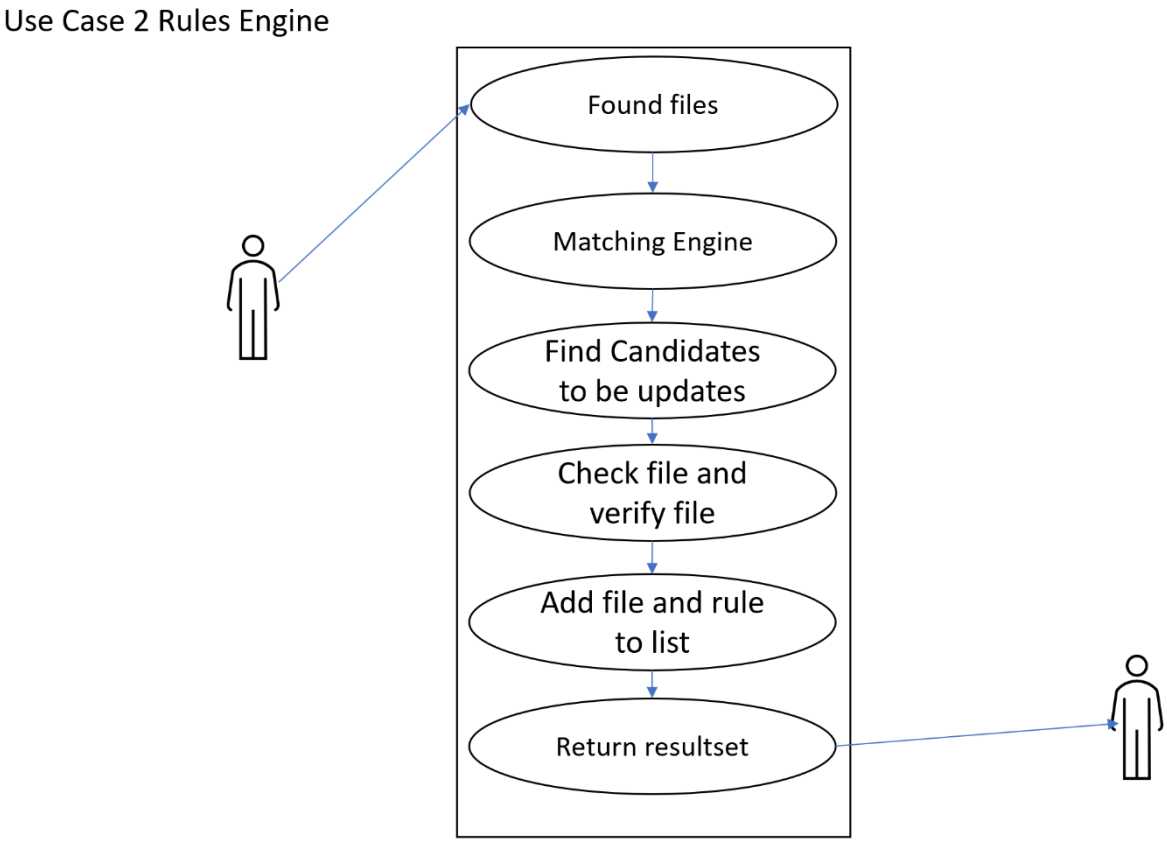

Figure 7: Use Case 2 – Rules Engine

6 Evaluation

The evaluation is based on the implementation of 7 security fixes outlined below. The 7 scenarios as outlined previously at a high level but will be expanded in this section. The evaluation will base the implementation and use of the POC into a full fledging production CLI tool. Acceptance will be taken developers and security developers in the firm. The report and configuration will be used to support and build test cases for tool usage.

6.1 X-Frame Options

We use the SAMEORIGIN policy for the X-Frame Options to allow only embedding iFrames on the domain itself and not by hosted third parties. This protects against clickjacking attacks through iFrames. [2]

<add name="X-Frame-Options" value="SAMEORIGIN" />

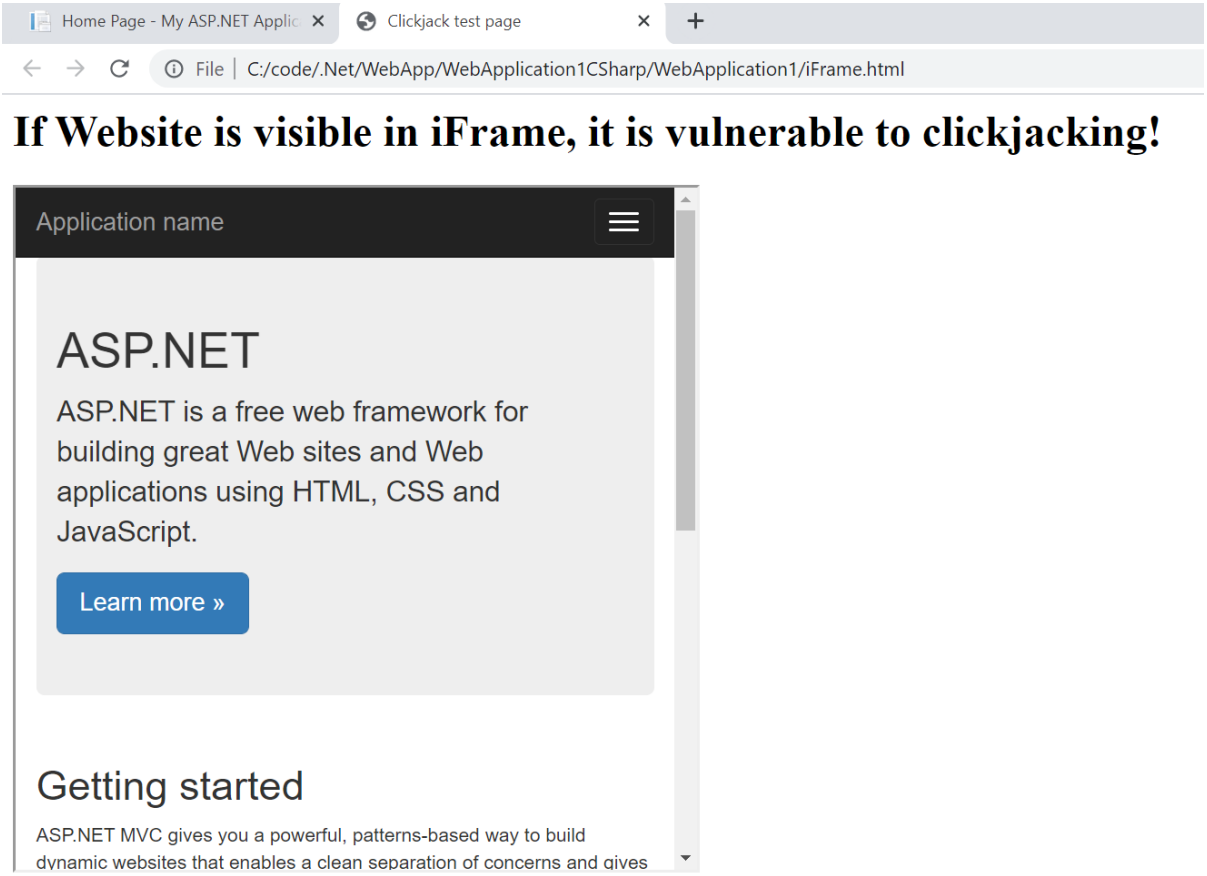

Figure 8: iFrame exploit demonstration

[2] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Figure 9: iFrame exploit disabled

6.2 Content-Security-Policy

The content security policy setting should adhere to organisation policy which ensures that content should only be loaded. This includes load content from the same domain and not cross domain for instance.[3]

<add name="Content-Security-Policy" value="default=src 'self' style-src 'self' 'unsafe-inline'" />

[3] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/default-src

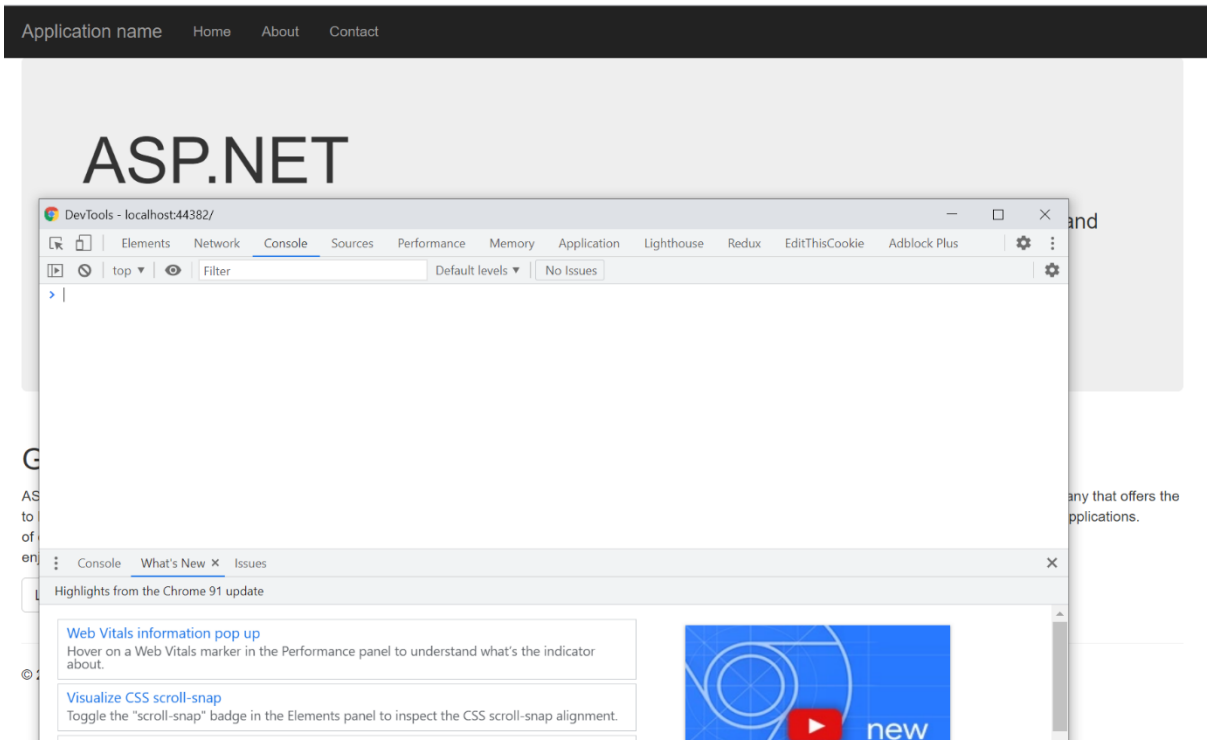

Figure 10: Pre-patching vulnerability– Ability to reference files from insecure locations

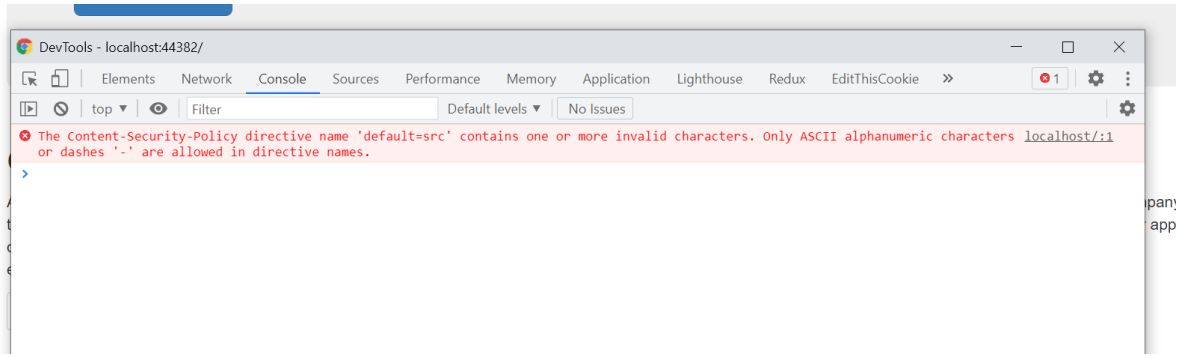

Figure 11: Remove ability to reference files from insecure locations

X-XSS-Protection

XSS Protection should be used to protect against XSS attacks, in this case, the use of 1; mode=block to prevent rendering of the attack.[4]

<add name="X-XSS-Protection" value="1; mode=block" />

6.3 X-Content-Type-Options

Enforce web browsers not to interpret content type that differs to though those directed by the server. To prevent possible attacks such as MIME confusion.[5]

[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection
[5] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

<add name="X-Content-Type-Options" value="nosniff" />

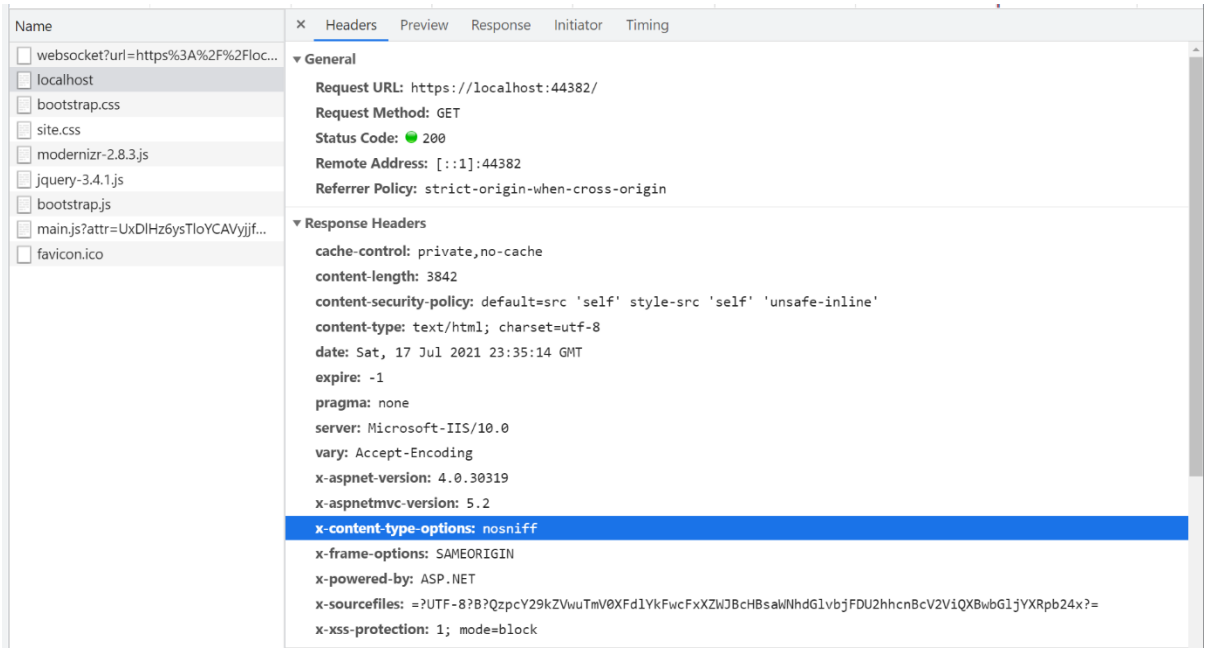

Figure 12: Prevent Mime confusion attacks

6.4 Cache-Control

Allows for web browsers to not to maintain sensitive data.[6]

<add name="Cache-Control" value="no-cache" />

6.5 Pragma

Allows for web browsers to not to maintain sensitive data. [7]

<add name="Pragma" value="none" />

6.6 Expires

Enforcing staleness on data to avoid caching of sensitive data. [8]

<add name="Expire" value="-1" />

[6] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control
[7] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Pragma
[8] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expires

Figure 13: Caching, expires and pragma not enforced

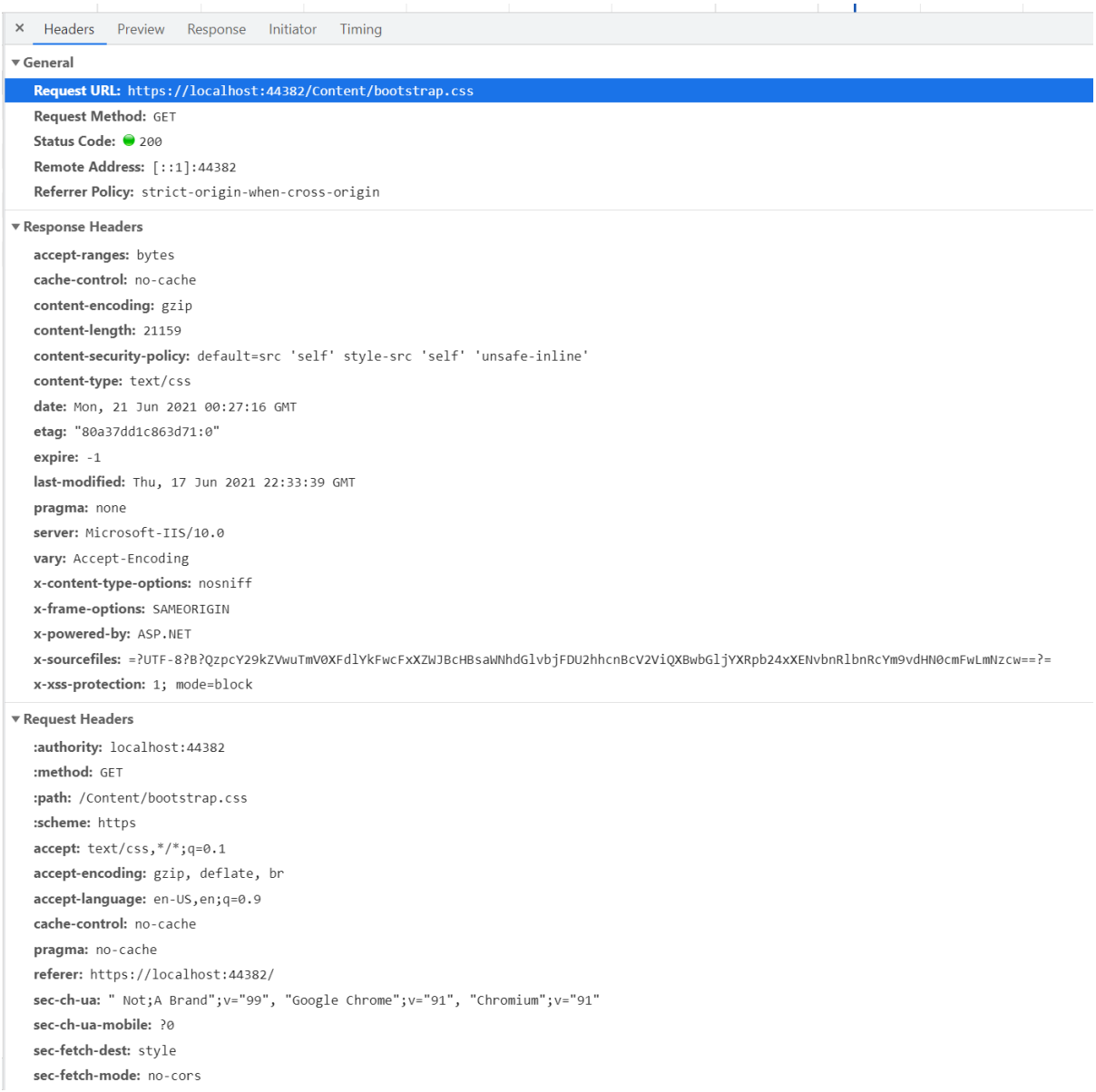

Figure 14: post-patching vulnerability – Enable expire, no-cache and pragma

The seven items were investigated before the experiments took place and after the CLI tool was run. Subsequent penetration tests will take place to confirm to risk teams that vulnerabilities do not exist.

7 Conclusion and Future Work

There is a vast amount of research in the field of cyber security that relates to secure programming as well. The initial outline was to try to understand why insecure software exists. The research points to the following items: return on investment, financial, training, lack of understanding and lack of secure development processes. This led to further investigations of SDLC and their defects and possible mechanisms to improve them. It is important that there is no one fits all security model that will fit an organisation, secure development may need to be customised and integrated into the SDLC being used, I provide two approaches for this at a high level. The CLI tool helps to address security issues in an automated manner in the early

stages of SDLC where remediating issues is cheaper. Automation is built into CI/CD which ensures security is embedded in the SDLC. Cost reduction as the remediations is automated, there is no lead time from penetration test findings and then the issues being addressed. The CLI can be used for any Microsoft .Net applications. Building tools like this will also build a culture whereby developers will actively address security issues and build securer products.

The future use of CLI will be a wider rollout to other business units in the organisation and to be rolled at a strategic organisation level. The organisation has placed a huge emphasis on security and automation of security defects. This first iteration of CLI is to address 7 specific cases but the rationale was to design and implement a tool from proof of concept to real world use. The efficacy of tool has been demonstrated to local stakeholders which will lead to other investment and expansion of toolset to address other security issues. As software frameworks are constantly being updated, the next use case will be to address .Net Core web applications. The modular approach of the CLI allows for it be extended without an overhaul of the application. The future development will be integrated in the Agile methodology used in the organisation which will help build the most urgent features.

References

Anantharaman, N. & Wukkadada, B., 2020. *Identifying The Usage Of Known Vulnerabilities Components Based On OWASP A9.* Pune, 2020 International Conference on Emerging Smart Computing and Informatics (ESCI), pp. 88-91.

Bartsch, S., 2011. *Practitioners' Perspectives on Security in Agile Development.* Vienna, 2011 Sixth International Conference on Availability, Reliability and Security.

ben Othmane, L., Angin, P., Weffers, H. & Bhargava, B., 2014. Extending the Agile Development Process to Develop Acceptably Secure Software. *IEEE Transactions on Dependable and Secure Computing,* 11(6), pp. 497-509.

Bunke, M., 2015. *Software-Security Patterns: Degree of Maturity.* New York, Association for Computing Machinery.

Buttner, A., Piazza, R., Purohit, R. & Summers, A., 2020. *A Secure Code Review Retrospective.* Atlanta, 2020 IEEE Secure Development (SecDev).

Daley, J., 2016. Insecure Software Is Eating the World: Promoting Cybersecurity in an Age of Ubiquitous Software-Embedded Systems. *Stanford Technology Law Review (STLR),* 19(3), pp. 533-546.

Devi, R. S. & Kumar, M. M., 2020. *Testing for Security Weakness of Web Applications.* Tirunelveli, 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184).

Dodson, D., Souppaya, M. & Scarfone, K., 2020. *Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF),* s.l.: National Institute of Standards and Technology.

Gasiba, T. & Lechner, U., 2019. *Raising Secure Coding Awareness for Software.* Jeju Island, 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW).

Graff, M. G. & van Wyk, K. R., 2003. *Secure Coding: Principles and Practices.* 1st ed. Califronia: O'Reilly Media, Inc..

Jaatun, M. G., Cruzes, . D. S. & Luna, J., 2017. *DevOps for Better Software Security in the Cloud Invited Paper.* New York, Association for Computing Machinery.

Knuth, D. E., 1989. The errors of TEX. *Software—Practice & Experience,* 19(7), p. 607–685.
Leite, L., Rocha, C., Milojicic, D. & Meirelles, P., 2019. A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys,* 52(6).

Łukasiewicz, K. & Cygański, S., 2019. *Security-oriented agile approach with AgileSafe and OWASP ASVS.* Leipzig, 2019 Federated Conference on Computer Science and Information Systems (FedCSIS).

Mburano, B. & Si, W., 2018. *Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark.* Sydney, 2018 26th International Conference on Systems Engineering (ICSEng).

McGraw, G. & Viega, J., 2002. *Building Secure Software: How to Avoid Security Problems the Right Way.* 1st ed. Boston : Addison-Wesley.

Nanisura Damanik, V. N. & Sunarin, . S. U., 2020. *Secure Code Recommendation Based on Code.* Depok, 2020 International Workshop on Big Data and Information Security (IWBIS).

O'Donovan, J., Wagner, H. F. & Zeume, S., 2019. The Value of Offshore Secrets: Evidence from the Panama Papers. *The Review of Financial Studies,* 32(11), pp. 4117-4155.

Oyetoyan, T. D., Cruzes, D. S. & Jaatun, M. G., 2016. *An Empirical Study on the Relationship between Software Security Skills, Usage and Training Needs in Agile Settings.* Salzburg, 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 548-555.

Qian, K., Parizi, R. M. & Lo, D., 2018. *OWASP Risk Analysis Driven Security Requirements Specification for Secure Android Mobile Software Development.* Kaohsiung, 2018 IEEE Conference on Dependable and Secure Computing (DSC).

Rasalam, J. & Elson, R. J., 2019. Cybersecurity and Management's Ethical Responsibilities: The Case of Equifax and Uber. *Global Journal of Business Pedagogy (GJBP),* 3(3), pp. 8-15.

Rauf, A. & AlGhafees, M., 2015. *Gap Analysis between State of Practice & State of Art.* Washington, 2015 Agile Conference.

Rice, D., 2007. Geekonomics: The Real Cost of Insecure Software. In: *Geekonomics: The Real Cost of Insecure Software.* s.l.:Addison-Wesley Professional.

Rigby, D. K., Sutherland, J. & Takeuchi, H., 2016. embracing agile. *Harvard Business Review*, May, pp. 40-50.
Shebli, H. M. Z. A. & Beheshti, B. D., 2018. *A study on penetration testing process and tools.* New York, 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT).

Sφhoe, H., Jaatun, M. G. & Boyd, C., 2018. *OWASP Top 10 - Do Startups Care?.* Glasgow, IEEE, p. 1.
Tabassum, M., Watson, S. & Lipford, H. R., 2018. *Evaluating Two Methods for Integrating Secure Programming.* Baltimore, Conference: the 49th ACM Technical Symposium.

Tisdale, S. . M. & Morris, R., 2015. CYBERSECURITY: CHALLENGES FROM A SYSTEMS, COMPLEXITY,. *Issues in Information Systems,* 16(3), pp. 191-198.

Wang, Z., 2018. *The Impact ofExpertise on Pair Programming productivity in a Serum team: A Multi-Agent Simulation.* Beijing, 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS).

Wu, P.-H., Hwang, G.-J. & Tsai, W.-H., 2013. An Expert System-based Context-Aware Ubiquitous Learning Approach for Conducting Science Learning Activities. *Journal of Educational Technology & Society,* 16(4), pp. 217-230.

Yu, H., Jones, N., Bullock, G. & Yuan, X. Y., 2011. *Teaching secure software engineering: Writing secure code.* Moscow, 2011 7th Central and Eastern European Software Engineering Conference (CEE-SECR).

Zhu, J., Xie, J., Lipford, H. R. & Chu, B., 2014. Supporting secure programming in web applications through interactive static analysis,. *Journal of Advanced Research,,* 5(4), pp. 449-462.