Project overview for portfolio upload

Project Title:

Secure Software Development – ScottishGlen's Software security case study

Project Description:

The following work is a case study around a company who has had their software systems breached by hacktivist groups. We had to identify a technology which the hacktivists exploited and recommend secure software development practices to prevent further exploitation. This project chose the web front end which was written with ASP.NET. A code review was selected as the secure development practice.

Project Tooling:

- Used C# code with ASP.NET for example

Project processes and techniques:

- Secure code review

Grade: A

Appendices have been removed to avoid plagiarism from future students, however I am happy to provide additional information or discuss the work further if desired.

***Disclaimer**: I do not give permission for this work to be copied, those who do may be liable for plagiarism.*

Table of Contents

1. Introduction

The following report outlines several recommendations made by ScottishGlen's IT manager to help improve the security of the company. These recommendations are considering the recent threat of attack made by a hacktivist group, providing a focus for ScottishGlen to improve their security and mitigate this attack in moving forward.

2. Context

2.1. Technology

The following report will focus on ScottishGlen's web front end which has been written with ASP.NET. ASP.NET is an open-source web framework, created by Microsoft and utilised for building web apps & services (Microsoft, 2022). This framework is widely used across the web, and as a result has created many vulnerabilities that have been reported to Mitres Common Vulnerabilities & Exposures (CVE) database. Therefore, this has led to ScottishGlen's web front end to be vulnerable to Cross-Site Scripting (XSS) such as the vulnerability outlined in CVE-2020-24903

2.2 CVE-2020-24903

CVE-2020-24903 is a XSS vulnerable that exists on ASP.NET software. This vulnerability enables malicious actors to exploit user provided inputs on the system with specially crafted URLs that allow malicious scripts to be executed on a website. This vulnerability will fall under CWE-79: "Improper Neutralization of Input During Web Page Generation". This CWE (Common Weakness Enumeration) details improper - or lack of - validation on the user inputs put in place during the development process. (Mitre, 2023).

3. Recommendations

3.1 Secure Code Review using OWASP guidelines.

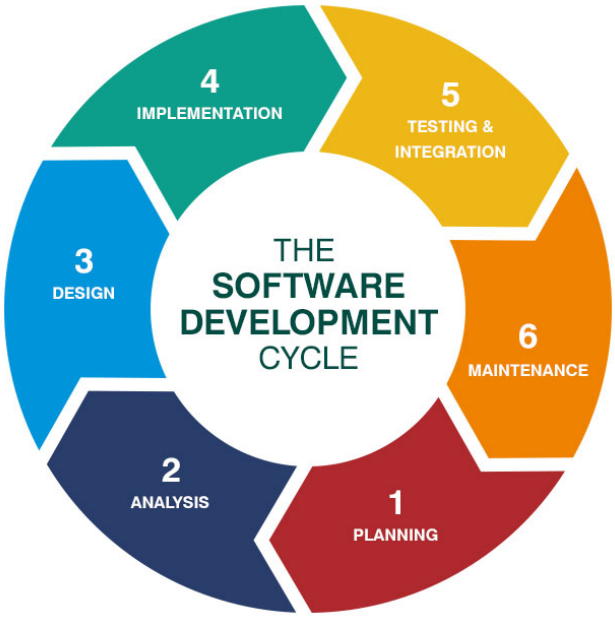
A secure code review can be highlighted as the best secure software development process for ScottishGlen. A secure code review can identify the causes of the XSS vulnerability highlighted by CVE-2020-24903. This review can also provide an effective framework in pre-emptively negating possible vulnerabilities of all types. This secure software development practice is often carried out by a nominated individual within the software development team who is familiar with the code base & tools used. However, this report will outline the necessity for integrating a secure code review across ScottishGlen's development team to ensure its limited resources are fully utilised. This review is built on secure software development best practices that are outlined in reputable resources such as OWASP's secure code review guide (OWASP and Eoin Keary, 2017). These best practices are compared with written code and then determined whether the code will pass or fail the security requirements because of the specification that is standardised with a checklist.

3.2 Benefits of a Secure Code Review

3.2.1 Continuous Improvement

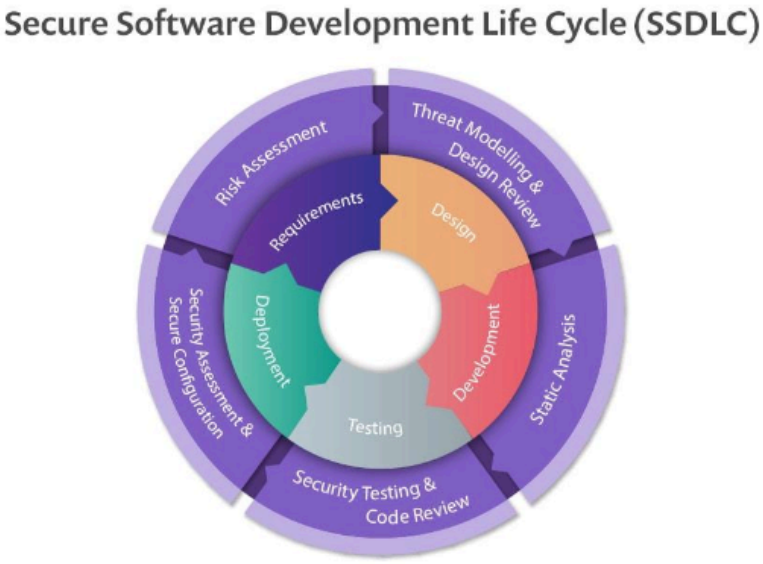
The structured methodology of a code review allows for this process to be integrated seamlessly with the modern, iterative version of the Software Development Life Cycle (SDLC).

An iterative process, such as the SDLC is beneficial when implementing a secure code review, as the process of reviewing a checklist – a method for implementing a secure code review – will allow developers to regularly analyse their work for security faults when allocated to different phases of the SDLC. When comparing this to automated testing, a secure code review will produce more nuanced relevant results. Automated testing works on specific parameters defined by the creator of the tool. These automated tools will often scan for well-known development issues and fail to recognise the entire complexity of the SDLC.



[Figure 1 – Software Development Life Cycle (SDLC)]

Organisations such as Snyk have developed their own interpretation of combining security methods with the SDLC. Synk have termed this the 'Secure Software Development Life Cycle' (SSDLC) (Snyk, 2021) indicating the importance of integrating security at every phase of the SDLC. A secure code review promotes collaboration, opportunity for feedback and most importantly a structured process for secure development that can integrate into ScottishGlen's development process.



[Figure 2 – Secure Software Development Life Cycle (SSDLC). Obtained from https://snyk.io/learn/secure-sdlc/]

3.2.2 Human Involvement

As previously stated, a secure code review will allow for human involvement to play a vital role in securing Scottish Glen's current and future systems. Firstly, context. Context is the process of understanding the requirements & specification that a piece of software hopes to achieve, for example deciding what constitutes as correct business logic within the software. Additionally, individuals can also determine the intent of a piece of software. This means that developers can decipher what parts of the code will perform once executed. Because of this understanding of the individual parts of code, false positives are less likely to affect security testing. False positives are when automated systems fail to pick up on errors of the code. These errors highlighted by automated systems are often syntax or semantic related and do not focus on the larger scaled elements of the software. However, that is not to say human interaction is perfect. Often when developers assess their own teams code, bias can play a role in determining whether a bit of code is useful or required to achieve the overall purpose of the software. To mitigate this, a third party might be tasked with the code review instead of an individual from the team, should resources allow this within ScottishGlen/

3.2.3 Tackling the customised systems used by ScottishGlen

As ScottishGlen's systems are all customised to some degree, it is important to maintain a fundamental level of security across all systems. This is done through enforcing a consistent methodology around secure development. This can be achieved with a secure code review. A secure code review can be tailored to ScottishGlen's system requirements both externally and internally. For instance, if ScottishGlen implement a new library, develop a large codebase, or use some new framework that combines multiple languages (such as their use of ASP.NET), standardised processes can be developed for checking the security of code.

This can be highly beneficial over automated tools that carry out methods such as static & dynamic analysis. These automated tools are built for generic & popular languages & libraries and as a result can sometimes only work on standard configurations of code bases or frameworks. This is a particularly important issue to consider for ScottishGlen. Scottish Glen's web frontend is written in ASP.NET. As a framework for web frontend, it is possible for developers to build their own libraries on ASP.NET, or in fact pull from a variety of different

libraries to help achieve their task (Microsoft, 2024). Furthermore, ensuring that the individual or team who has carried out the code review is able to give customised and relevant feedback to the rest of the team about the code is crucial in implementing the right security strategies.

3.3 Potential roadblocks

3.3.1 Lack of security expertise

ScottishGlen does not have employees who are solely responsible for security. Because of this lack of knowledge within the company, it is important that a seamless and easy to follow implementation of a secure code review is carried out to ensure that ScottishGlen has an efficient strategy, both from an employee and technical perspective to implement the review correctly. This can be achieved by utilising training resources to ensure everyone in ScottishGlen is up to date on security measures.

3.3.2 Small team with limited resources

It is important to note that there still may be some difficulty in implementing a secure code review into such a small team. Often, when implementing a secure code review, it is important to have a designated reviewer who can designate their time and company resources to analysing the code for faults. This is not as feasible in smaller teams, where it is important for each employee to be working collectively on whatever project is the company's priority. Therefore, automated security tools such as static analysis could be seen as a better option over a manual code review. However, automated tools can often produce reports that are not easy to translate if the developers do not have a background in security. The time it takes to decipher and act upon these reports can negate the time saved on automated processes, strengthening the argument for secure code reviews. Additionally, if ScottishGlen implement the processes of a secure code review across each phase of the SDLC, security principles can be implemented at every level, allowing for a secure code review to be done more efficiently.

3.4 Foundation for further security

By implementing a secure code review, processes can be put into place on daily bases, and ultimately becoming part of the way ScottishGlen develop their software going forward. Through several of the strengths which were outlined previously in the report, developers in ScottishGlen are now able to go further with their security procedures and have a structured foundation for utilising some of the automatic tools that can help detect vulnerabilities. It is an optimal time for these tools to be introduced as the code review will provide a foundation for developers. This combining of approaches is only possible through a foundation built through secure code reviews and will be an aim for ScottishGlen.

4. Implementation

4.1 Cross-Site Scripting (XSS)

Cross-Site scripting is a common web application vulnerability which hackers such as those attacking ScottishGlen, can input malicious scripts. This is possible if incorrect practices have been followed in the development phase of software development. These scripts, once successful can allow for hackers to intercept user input, stealing sensitive date or hijacking user sessions. This is particularly dangerous when systems containing critical information have been breached such as Scottish Glen's web front end that manages users.

4.2 Input Validation

Understanding how to combat security faults that lead to XSS are pivotal for companies, including ScottishGlen, to identify in the development process. Identifying security faults such as input validation before shipping vulnerable software Is essential to not waste time in the development process with preventative maintenance (Braz *et al.,* 2021).

Input validation is the process of testing inputs received from the user to ensure that the inputted data meets a certain acceptable criterion. What is deemed acceptable is often dictated by the software developers and will be in relation to what data the system can reliably manage. When a system does not include input validation, this system can then often be

manipulated and be used by malicious actors such as the hacktivist group targeting ScottishGlen. Scottish Glen have web front end is used by HR staff to manage users and their details. Therefore, it can be assumed that CRUD (Create, Read, Update, Delete) operations will be prevalent on this system. These types of operations send data back and forth between user inputs & the system, therefore no input validation allows for this system to be open to XSS.

4.3 Education

Understanding the causes behind vulnerabilities such as Cross-Site Scripting is crucial in developing preventative measures. This education can be easily obtained for development teams of all sizes. For example, OWASP, alongside their code review guide, have a designated top 10 list of the most popular web application security risks (OWASP, 2021) .This top 10 is highly valuable and is the basis for the OWASP code review guide 2.0 which is the foundation of a suggested secure code review. Other organisations such as Snyk (Gee, *et al,* 2020*)* or MITRE also have their own tailored information on how to perform a secure code review (Buttner *et al.,* 2020). Therefore, utilising a knowledge base by one of these organisations, will allow developers to be informed of common vulnerabilities and help combat the faults before they become a vulnerability. Standardising this information into a reliable methodology is how a code review can be created, one way of doing so is in the form of a checklist.

4.4 Standardising the code review; implementing a checklist

It can be easy for Scottish Glen's software development team to believe they are implementing the correct practices for secure code review. However, until they are given the ability to validate the security of their code with a structured process, faults will fall through and lead to vulnerabilities (Gonçalves *et al.,* 2020). This is not the fault of the developers, they can often be under stress, with considerations in other areas out with security that are enforced by management. Therefore, it is important for ScottishGlen to formalise their own secure code review. This can be done by introducing a checklist with areas to validate before moving on. This methodology is outlined by OWASP and can be adapted for any vulnerability or situation, regardless of size. When analysing code for potential errors leading to cross site scripting such as input validation, there are several parameters to identify and correct.

Derived from the checklist provided by the OWASP code review guide 2.0, Figure 3 highlights several input validation errors that could leave Scottish Glen's web front end open to a XSS attack. The following section demonstrates this checklist in practice, showing the value of a code review with a standardised process.

	CATEGORY	DESCRIPTION	PASS	FAIL
1	Business Logic and Design	Does the design maintain any exclusion list for parameters or features from being validated?		
2	Business Logic and Design	Does the centralized validation check block all the special characters?		
3	Business Logic and Design	Does the centralized validation get applied to all requests and all the inputs?		
4	Imput Validation	Are all the untrusted inputs validated? Input data is constrained and validated for type, length, format, and range.		
5	Business Logic and Design	Does are there any special kind of request skipped from validation?		
6	Data Management	Is output that contains untrusted data supplied input have the correct type of encoding (URL encoding, HTML encoding)?		

[Figure 3 – Checklist derived from master checklist on OWASP code review guide]

4.5 Demonstrating the value of following a secure code review

To validate the effectiveness of the checklist created for the secure code review, source code will be analysed for potential faults regarding its input validation. Figure 4 shows source code written in C# for an ASP.NET app. The source code below has several input validation errors and can be seen from the comments on the code alone. Moreover, when analysing the source code in relation to the checklist, its omission of input validation is more prominent .

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;

namespace VulnerableApp.Controllers
{
    [Route("api/[controller]")]
    public class DataController : Controller
    {
        // POST api/data
        [HttpPost]
        public void Post([FromBody]string value)
        {
            //No validation on the input
        }
    }
}
```

[Figure 4 – Source code taken from: *https://docs.fluidattacks.com/criteria/fixes/csharp/186/*]

4.6 Checklist comparison

The checklist can be used as a guide for Scottish Glen when implementing a secure code review on the code snippet highlighted in figure 5. Even if one section of the checklist fails, there is reason for revision of the original code. The summary of the failures is as follows outline in figure 5.

	CATEGORY	DESCRIPTION	PASS	FAIL
1	Business Logic and Design	Does the design maintain any exclusion list for parameters or features from being validated?		✓
2	Business Logic and Design	Does the centralized validation check block all the special characters?		✓
3	Business Logic and Design	Does the centralized validation get applied to all requests and all the inputs?		✓
4	Imput Validation	Are all the untrusted inputs validated? Input data is constrained and validated for type, length, format, and range.		✓
5	Business Logic and Design	Does are there any special kind of request skipped from validation?		✓
6	Data Management	Is output that contains untrusted data supplied input have the correct type of encoding (URL encoding, HTML encoding)?		N/A*

* Applicable to the code snippet as it is valid, however its not applicable to input validation & XSS

[Figure 5 – Checklist Complete]

Summary

Implementing a secure code review which follow OWASP guidelines can bring several benefits to ScottishGlen. Firstly, it will allow Software Developers to identify current vulnerabilities related to CVE-2020-24903 and fix these vulnerabilities using a standardised checklist to identify missing input validation. This structured secure code review can help prevent any further vulnerabilities related to XSS and can be transferred to other vulnerabilities & deliver a secure software development process overall.

References

Buttner, A., Piazza, R., Purohit, R. and Summers, A. (2020) 'A Secure Code Review Retrospective', *2020 IEEE Secure Development (SecDev),* , pp. 31-32. doi: 10.1109/SecDev45635.2020.00020 https://ieeexplore.ieee.org/document/9230284

Gonçalves, P.W., Fregnan, E., Baum, T., Schneider, K. and Bacchelli, A. (2020) 'Do Explicit Review Strategies Improve Code Review Performance?', *2020 IEEE/ACM 17th International Conference on Mining Software Repositories (MSR),* , pp. 606-610. doi: 10.1145/3379597.3387509 https://ieeexplore.ieee.org/document/10148688

Microsoft (2022) *ASP.NET overview.* Available at: https://learn.microsoft.com/en-us/aspnet/overview (Accessed: Feb 27, 2024)

ASP.NET Core | Open-source web framework for .NET. (2024) Available at: https://dotnet.microsoft.com/en-us/apps/aspnet (Accessed: Feb 27, 2024)

Mitre (2023) *CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')*
. Available at: https://cwe.mitre.org/data/definitions/79.html (Accessed: 27th February, 2024)

OWASP (2021) *OWASP top 10 Web Application Security Risks.* Available at: https://owasp.org/www-project-top-ten/ (Accessed: 27th February, 2024)

OWASP and Eoin Keary (2017) *OWASP Code Review Guide 2.0.* Available at: https://owasp.org/www-project-code-review-guide/ (Accessed: 27th February, 2024)

Snyk *Secure Software Development Lifecycle (SSDLC).* Available at: https://snyk.io/learn/secure-sdlc/ (Accessed: 27th February, 2024)

Snyk, Brian Vermeer and Trisha Gee (2020) *Secure code review: 8 security code review best practices.* Available at: https://snyk.io/blog/secure-code-review/ (Accessed: 27th February, 2024)