CMP417 – Engineering Resilient Systems

Secure Software Development

Paul Michael Oates

2001642

23/02/2024

Table of Contents

1. Context

Scottish Glen, a small Scottish company in the energy sector has recently received a series of threats after a hacktivist group took exception to comments made by their CEO, employees have reported to the IT team that they have received threatening messages from this hacktivist group, but no specifics have been mentioned in any of the threating emails.

To combat this, the CEO has expressed his wishes that the company's existing IT infrastructure is reviewed. The HR system is a vital piece of software that enables Scottish Glen to function on a day-to-day basis and if attacked could lead to data loss of sensitive employee information. ASP.NET is used as the web frontend. It is utilised by HR staff to manage employees and their details. Therefore, for the purposes of this report this will be discussed so vulnerabilities can be identified and remediated.

Scottish Glen's HR system is written in Active Server Page (ASP.NET). ASP.NET is a Microsoft product which is described as a "Free. Cross-platform. Open source. A framework for building web apps and services with .NET and C#." (Microsoft, ND) The initial release of ASP.NET was in 2002 and in its modern iterations applications can be written in a variety of .NET languages such as Visual Basic, C#, or F#. (Tyler, 2023)

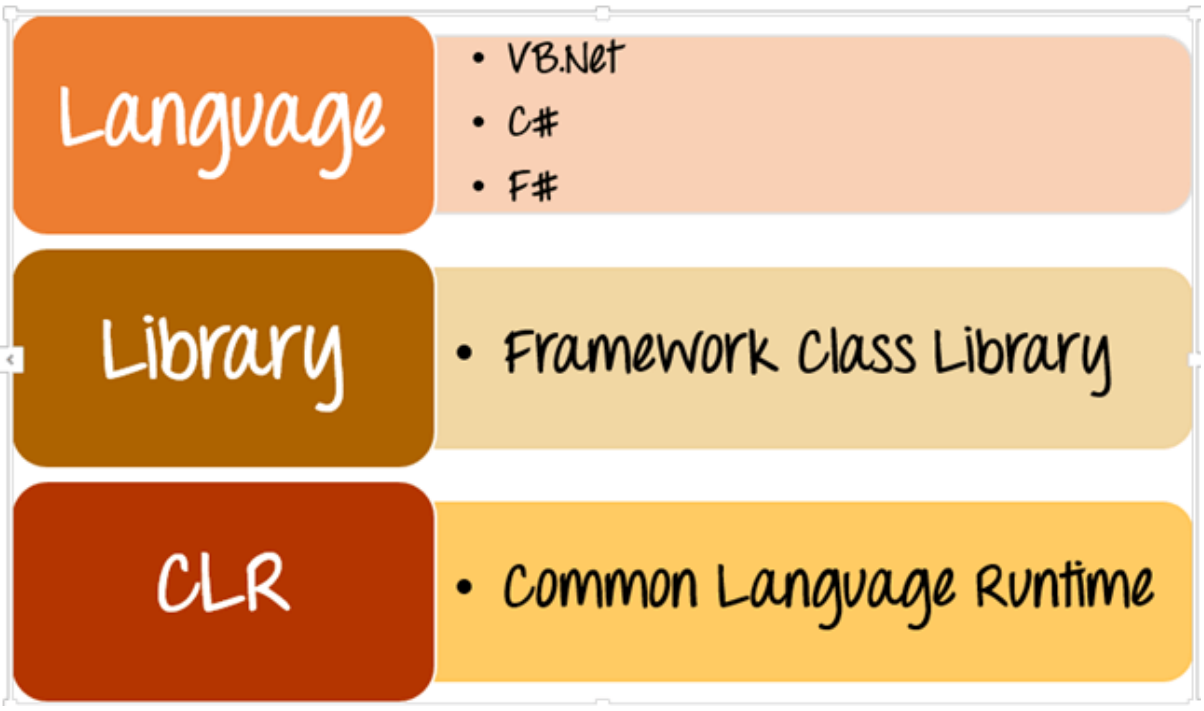

Figure 1-1 ASP.NET Architecture (Tyler, 2023)

Figure 1-1 identifies how the .NET framework is structured. This structure and the wide-open source support of Microsoft's .NET's environment enables developers to build applications in a variety of languages with widely used and supported standard libraries. With regards to ASP.NET it enables developers to separate code and design, state management which allows for objects to be saved, and the caching of popular pages so the website load faster. This makes it a popular service for developers to develop their applications.

However, due to ASP.NET's age, its popularity and easy access, it is an attractive target for threat actors. There have been several documented vulnerabilities within the ASP.NET framework and its libraries. The United States National Institute of Standards and Technology (NIST) creates a list of vulnerabilities within applications called the National Vulnerability Database (NVD) or CVE database.

1.1. NVD Database Scoring

The National Vulnerability Database or CVE Database is a collection of public recorded vulnerabilities in applications disclosed by members of the security community and numbered by the CVE Numbering Authority (CNA) which includes companies such as Airbus, Siemens and Apple. Vulnerabilities disclosed in the National Vulnerability Database (NVD) are typically accompanied with information regarding the affected software versions, disclosure dates, severity, source, and a brief description is typically provided.

CVEs disclosed are given a code by the CNV which follows the form CVE-AAAA-BBBBB. Where A is the given year and B is a unique identifier of a specific vulnerability. For example, CVE-2023-36038 would be a vulnerability from 2023 and "36038" would be the unique code. This vulnerability was assigned it's CVE number by Microsoft.

Vulnerabilities documented in the NVD are also given a score against the Common Vulnerability Scoring System (CVSS). There are two active versions of the CVSS scoring 2.0 and 3.0.

CVSS v2.0 Ratings		CVSS v3.0 Ratings	
Severity	**Severity Score Range**	**Severity**	**Severity Score Range**
		None*	0.0
Low	0.0-3.9	Low	0.1-3.9
Medium	4.0-6.9	Medium	4.0-6.9
High	7.0-10.0	High	7.0-8.9
		Critical	9.0-10.0

Figure 1-2 CVSS ratings (NIST)

Figure 1-2 identifies the CVSS scoring system. The CVSS 3.0 scoring system investigates three metrics; Base, Temporal, and Environmental to deliver a score between 0 and 10. The Base metric is the exploitability of a vulnerability, metrics such as permissions, complexity, confidentiality all effect this score. The Temporal metric accounts for factors such as maturity of code, remediation method, and confidence in the report. Finally, the Environmental metric is the presence of controls and importance of the compromised information.

Vulnerabilities in ASP.NET are wide ranged and varied in nature however for the purposes of this report the focus will be Denial Of Service (DOS) because of the high scoring and frequency of these vulnerabilities within ASP.NET.

1.2. Denial Of Service (DOS)

There are many different vulnerability classifications within National Vulnerability Database which occur within ASP.NET. Typical vulnerability classifications such as Denial Of Service (DOS), Remote Code Execution (RCE), Insecure Direct Object References (IDOR), Cross Site Scripting (XSS) are all mitigated against in different ways.

Another organisation is the Open Web Application Security Project (OWASP) it describes how to test applications such as websites to mitigate against vulnerabilities such as Denial Of Service (DOS). DOS attacks occur when users are denied access to computer services or resources, usually by overloading the service with requests impeding its ability to function (NCSC, 2020).

The goal of a Denial Of Service attack aims to take down an application, server, or website to make it inoperable to the end users. Methods to perform this exploit vary however they typically involve sending large volumes of data to a service which slows down and eventually stops the service and any other services on the host from performing. (OWASP, 2021) When this occurs it is sometimes possible for malicious actors to perform code injection to the underlying operating system which access sensitive data the service stores. In the case of Scottish Glen this could lead to exposure of names, addresses and any other information the HR program may store.

One example of a Denial Of Service attack is The Mirai Dyn DDOS Attack in 2016. In this attack multiple DOS attacks called a Distributed Denial Of Service attack (DDOS) flooded the Dyn DNS provider with 1 terabyte per second of network traffic which resulted in websites such as Reddit, PayPal and GitHub to be inaccessible for a number of hours. It is worth noting that DOS or DDOS as a service providers could be paid for by Hacktivists or ran on their own machines to target the public facing resources of Scottish Glen.

2. Recommendation

2.1. Fuzzing

The report recommends implementing secure coding practices to mitigate against vulnerabilities. One method that could be incorporated is Fuzzing specifically Network Fuzzing. Network Fuzzing would increase the likelihood to be able to catch vulnerabilities and crashes that occur on the webpage. By implementing this secure coding practice within a Continuous Integration/ Continuous Development CI/CD pipeline would enable the development team to detect for vulnerabilities including DOS attacks prior to pushing the application into a production environment.

Fuzzing is a Black Box testing technique that injects semi random values into a program to see how the application reacts and logs any suspicious responses or crashes. Malicious values such as tags “<>” or brackets “()” are typically included alongside completely random ASCII or raw binary to see how the application reacts to the input. (OWASP, 2021) When an application crashes by fuzzing it gives the developer the input that caused the crash which in turn enables an opportunity to quickly understand what caused the crash and where in the code the crash

happened. This can then be mitigated against. Fuzzing should be incorporated with other traditional debugging techniques such as code review.

2.2. Code Review incorporating Fuzzing.

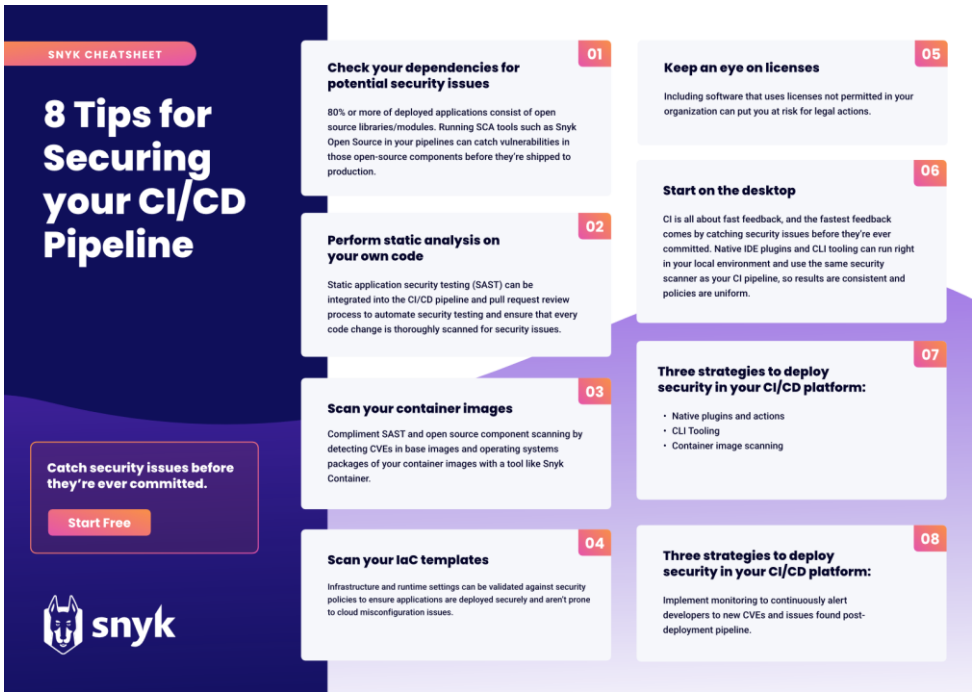

Figure 2-1 CI/CD Pipeline advice (SNYK, ND)

Figure 2-1 identifies SNYK's high level advice for secure coding practices, one suggestion made is Static Analysis or Code Review. Code Review is when the team of developers would collaborate and manually analyse the code on a weekly basis. Incorporating the crashes that fuzzing caused into the discussion would enable the developers to discuss why the application is crashing and where in the code this is occurring.

In relation to the specific ASP.NET application network packet fuzzing would enable the IT team to see how the application responds to a large volume of data as well as specially crafted malformed data. (OWASP, 2023) As put by Jun Li et al "Fuzzing is currently the most effective and efficient vulnerability discovery solution" (Li, 2018). Organisations such as Google have developed and published their own fuzzing tools such as ClusterFuzz which they employ against their own Codebase. According to Google ClusterFuzz has detected around twenty-seven thousand bugs in Google products such as Chrome. (Google, 2024)

However, one issue with Fuzzing an application is the considerable number of false positives it can identify (Hill, ND). This could lead to a large amount of wasted time at a company as small as Scottish Glen. This risk is compounded by developers becoming over reliant in this practice and potentially missing or disregarding vulnerabilities as the Fuzzer did not crash the

application. Fuzzing should be used as one tool of many that the development team implements as discussed in Figure 2-1. Such as container image scanning.

2.3. ISA / IEC 62443-4-1

ISA / IEC 62443-4-1 is an Operational Technology (OT)/ Industrial Control System (ICS) standard for automation and control systems for the secure product development lifecycle (SDLC). It includes requirements for Static Code analysis and Malformed Input Testing (Fuzzing) against devices such as Programable Logic Controllers (PLC) to mitigate vulnerabilities (FORTA - Beyond Security, ND). This Standard aims to enable security to be implemented throughout the SDLC to prevent vulnerabilities including Denial Of Service (DOS) attacks.

3. Implementation

The most recent version of the .NET framework is 8.0.2 and is supported (performance enhancements and vulnerability patching) till November 2026. This stable build should be the version that Scottish Glen uses throughout their codebase. The developers can develop the application in Linux, MacOS, and Windows.

3.1. CVE-2023-36038

An unconventional DOS attack is CVE-2023-36038 this vulnerability is present in ASP.NET Core version 8.0 with a CVSS 3 score of 7.5 (High) (NIST, 2023). The attack occurs when a malicious HTTP POST request is sent to a ASP.NET page with the Content-Length set to a large value such as 10000000000 and Content Type set to multipart/form data segment. (CVE CyberSecurity Database News, 2023) By using a tool like Burp Suite a security researcher or even a malicious attacker can forward a specially crafted packet that meets these requirements.

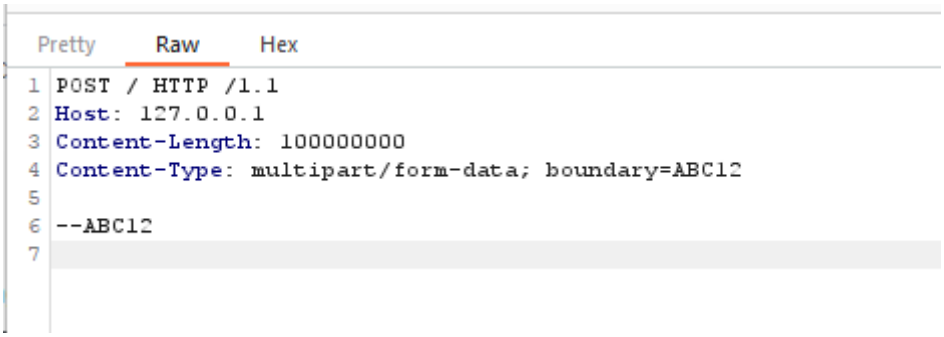

Figure 3-1 Network Packet Data

Figure 3-1 identifies the modified packet within Burp Suite, the single packet instructs the webserver to provision a large amount to space of the servers memory and wait for several packets of data, however no further packets are sent. Ultimately this will starve the sever of its available resources making the server slow or potentially completely unresponsive to legitimate end users.

3.2. Fuzzing and Mitigation

CVE-2023-36038 is a vulnerability that enables a hacker through a single packet to take down a webserver, regarding Scottish Glen this would prove a genuine issue as it could potentially result in HR being unable to provide their essential functions. To mitigate this vulnerability the development team can add resilience to their software development CI/CD pipeline by employing network Fuzzing prior to pushing an application to production.

One tool that could be used is “wfuzz” which Fuzzes all input boxes on a given application. A command such as:

“`Wfuzz -w wordlist/general/common.txt --hc 404 http://TESTWEB.NET/FUZZ`”

By Fuzzing the application enables it to crash in a manner that allows the developers to understand what went wrong and how to mitigate it. However, this would only give the developers an indication of an issue and not the solution and other secure software development methods would need to be deployed such as a code review to discuss the solution to the crash.

By employing resilience in critical systems makes DOS vulnerabilities harder to deploy by the Hacktivists. One solution that would have prevented this vulnerability prior to the CVE being disclosed was limiting content length “`options.MaxRequestBodySize = LIMIT;”`. Adding further resilience by employing a load balancing cluster in a round robin fashion would assist in mitigating this vulnerability as separate requests would go to separate servers.

Finaly adherence to proper modern coding standards set by The Open Web Application Security Project (OWASP) enables developers to keep abreast of modern secure coding standards. As stated in the standard ISA / IEC 62443-4-1 aims to prevent CVEs such as CVE-2023-36038 from occurring in Operational Technology (OT). However, Scottish Glen could also apply this standard to its IT infrastructure with the aim to mitigate similar vulnerabilities.

In conclusion, if the software engineers at Scottish Glen follow secure coding practices and implemented fuzzing into the CI/CD pipeline the chances of CVE-2023-36038 being detected prior to the production push would have been greatly increased.

4. References

CVE CyberSecurity Database News, 2023. *CVE-2023-36038: A Deep Dive into the ASP.NET Core Denial of Service Vulnerability and How to Mitigate It*. [Online]
Available at: https://www.cve.news/cve-2023-36038/
[Accessed 21 02 2024].

FORTA - Beyond Security, ND. *How to Use SAST and DAST to Meet ISA/IEC 62443 Compliance*. [Online]
Available at: https://www.beyondsecurity.com/blog/isa-iec-62443-security-testing
[Accessed 25 02 2024].

Google, 2024. *ClusterFuzz*. [Online]
Available at: https://github.com/google/clusterfuzz
[Accessed 21 02 2024].

Hill, M., ND. *What is Fuzzing in Cyber Security: A Game Changer*. [Online]
Available at: https://cyberexperts.com/what-is-fuzzing-in-cyber-security/
[Accessed 21 02 2023].

Li, J. Z. B. &. Z. C., 2018. *Fuzzing: a survey*. [Online]
Available at: https://cybersecurity.springeropen.com/articles/10.1186/s42400-018-0002-y#citeas
[Accessed 21 02 2024].

Microsoft, ND. *ASP.NET Core*. [Online]
Available at: https://dotnet.microsoft.com/en-us/apps/aspnet
[Accessed 21 02 2024].

NCSC, 2020. *Denial of Service DoS*. [Online]
Available at: https://www.ncsc.gov.uk/files/Preparing%20for%20DoS%20(Denial%20of%20Service)%20attacks_V2.pdf
[Accessed 21 02 2024].

NIST, 2023. *CVE-2023-36038 Detail*. [Online]
Available at: https://nvd.nist.gov/vuln/detail/CVE-2023-36038
[Accessed 21 02 2024].

OWASP, 2021. *Denial of Service*. [Online]
Available at: https://owasp.org/www-community/attacks/Denial_of_Service
[Accessed 21 02 2024].

OWASP, 2023. *Fuzzing*. [Online]
Available at: https://owasp.org/www-community/Fuzzing
[Accessed 21 02 2024].

SNYK, ND. *8 Tips for Securing Your CI/CD Pipeline*. [Online]
Available at: https://go.snyk.io/cicd-security-cheatsheet-dwn-typ.html
[Accessed 23 02 2023].

Tyler, C., 2023. *What is ASP.NET? and it's ARCHITECTURE*. [Online]
Available at: https://www.guru99.com/what-is-asp-dot-net.html
[Accessed 09 Febuary 2024].