Software Security

Adam Board
2005335

1 Context

1.1 Introduction

ScottishGlen is a small organisation within the energy sector. Recently, hacktivists have been making threats towards the organisation via the employees, because the hacktivists were disgruntled from comments made by the CEO of the organisation. ScottishGlen's IT Manager has been tasked with analysing and identifying potential security vulnerabilities within the software the organisation uses for its day-to-day operations. This is because the CEO of ScottishGlen aims to improve the overall resilience and security posture of the organisation when facing cyber attacks.

After discussing with the development team and technical staff about the current software being used by the organisation, it was determined that numerous software systems contained potential security issues. The reason for concern was because the software could contain exploitable vulnerabilities that would allow the hacktivist group to cause reputational damage to the organisation. A critical piece of software being used by HR and Payroll as a database system is PostgreSQL. A database for the HR and payroll system could contain highly sensitive data related to the employees. Additionally, it is common for hackers to target databases for either selling the data or disrupting an organisation's workflow. Should the data integrity of the HR and Payroll system be compromised, ScottishGlen may have to pay a large General Data Protection Regulation (GDPR) fine of up to £17.5 million or 4% of their total annual worldwide turnover, whichever value is higher (Information Comminsioner's Office, 2023). For this reason, PostgreSQL has been chosen as the focus of this report.

1.2 PostgreSQL

PostgreSQL allows for the creation of object-relational databases. This means complex data can be separated into multiple tables that link together through unique identifier fields. PostgreSQL utilises queries to search and modify information within the complex database structure. PostgreSQL began in 1986 at the University of California under Professor Michael Stonebreaker. PostgreSQL is as popular as the widely adopted MySQL due to its extremely feature-rich capabilities and robustness in security (Amazon Web Services, 2024). With PostgreSQL being open source, a lot of independent developers create extensions to improve PostgreSQL. However, even with a level of focus on security and a large community backing, PostgreSQL still contains vulnerabilities which could be exploited by unauthorised attackers.

1.3 Common Vulnerabilities and Exposures Database (CVE)

The CVE database is a repository of known and reported vulnerabilities which are categorised by a multitude of factors such as severity and type of issue. The data base is commonly known as the National Vulnerability database and is maintained by the National Institute of Standards and Technology. Each entry in the database contains a concise description of the vulnerability with links to references and Proof of Concepts. The severity of each vulnerability is calculated using the Common Vulnerability Scoring System (CVSS). CVSS is a qualitative measure of severity based on a numerical scale of 0.0 to 10.0, with 10.0 being the critical vulnerabilities, and 0.0 being the non-severe vulnerabilities. The numerical values are translated into the ranges "Low", "Medium", "High", and "Critical" to ensure organisations handle the vulnerabilities according to their severity (NIST, 2023). For PostgreSQL, the CVE-2023-5869, an integer overflow vulnerability, had been identified as a potential security concern.

1.4 Integer Overflow

An integer overflow vulnerability is a type of arithmetic overflow error that happens when the result of an integer operation does not fit within the allocated memory space. Rather than outputting an error, an unexpected result will occur. While this unexpected result does not typically cause any issues, there are cases in which it can become a concern. Should an integer overflow occur when calculating the length of a buffer, a buffer overflow could happen, which may allow an unauthorised attacker to execute malicious code within a system. A simplified example of how buffer overflow works can be seen in Figure 1.

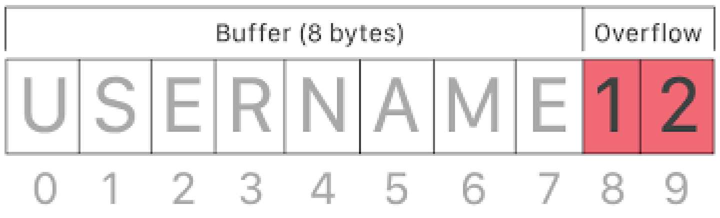

Figure 1 Simplified example of a buffer overflow.

The second case in which an integer overflow becomes a concern is if the overflow happens during a financial calculation. This could result in a negative account becoming positive, or a customer receiving credit rather than paying for a purchase (nidecki, 2023). The second case scenario is especially a concern because the PostgreSQL system is tied to ScottishGlen's payroll system.

Integer Overflow is listed as number 14 in the Common Weakness Enumeration's (CWE) most stubborn weaknesses (CWE, 2023). Although the vulnerability is still a serious concern, in recent years the vulnerability has been on a downward trend in terms of ranking. This can be seen in Figure 2 as CWE-190.

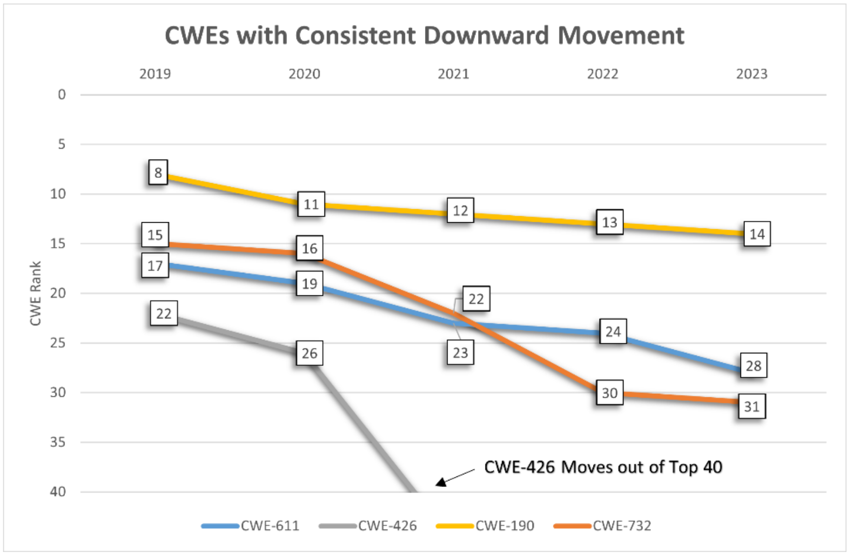

Figure 2 Graph of CWE's downward trend in ranking (CWE, 2023).

2 Recommendation

An integer overflow could cause catastrophic damage to the organisation should this not be caught by the development team. The difficulty of an integer overflow is detecting and debugging it, as it does not typically throw an error when it occurs, but instead throws a wrong result at the end of the operation. To overcome and mitigate the vulnerability, the development team should implement an ongoing dynamic code analysis with "fuzzing". The results should be manually checked to ensure they match up with the expected output for each calculation.

Dynamic analysis, also known as Dynamic Application Security Testing (DAST), is a method of testing an application for exploitable vulnerabilities. DAST is done through running the application with various tools such as a Fuzzer to identify both compile and runtime vulnerabilities. A fuzzer will typically use a dictionary of known malicious inputs to identify if a vulnerability exists inside of an application. The tool can input SQL Queries, Long input strings, large negative and positive numbers, and unexpected input data. Using this type of tool can even identify vulnerabilities stemming from memory allocation issues, which would typically be near impossible to detect (CheckPoint, 2023). The tools are an automated process which can have drawbacks such as an increase in false positives and negatives, and requires code which can work without crashing due to bugs. To resolve false positives and negatives, continuous manual checks should be conducted to ensure the output matches the expected values.

It is recommended that the Fuzzer used for dynamic code analysis is run at all times during development and especially at the production stage, as integer overflows can occur with specific inputs that are rare to come across. Running this kind of test will aid in discovering security faults before they become a fundamental piece of code that is ingrained and hard to replace (HARMANCI, 2023). However, running the Fuzzer at all times will be resource intensive for the organisation because a constant stream of inputs will have to be inserted into the software.

The Fuzzer is a vital tool for dynamic code analysis on software that handles input, like the payroll system. It is important to use the vast range of tools available. The various tools include: Debuggers to analyse the behaviour of the software, Profiliers to collect data on software's performance during runtime, and memory analysis tools to analyse a software's memory usage (Fronty, 2023). Utilising these tools during dynamic code analysis will support the development team in identifying bugs, memory leaks, security faults, and areas of optimisation within the code. Examples of these tools include valgrind, GDB, and Radamsa.

One major advantage of dynamic code analysis is that the tool suite used does not require the code to be available so long as the software is executable. This is useful for security fault detection during the deployment and testing phases, especially when using third party software as it can detect security faults caused by third party sources during testing.

If the development team were to introduce dynamic code analysis into their previous, ongoing, and future projects, they would improve the overall security of ScottishGlen. These improvements could be promoted within the organisation to showcase their commitment to ensuring the systems they use are developed with security and optimisation in mind. Additionally, using the dynamic tools on the development environment version of the software can provide insights into how the application would perform in real scenarios.

3 Implementation

To prove that the implementation of dynamic code analysis would aid in mitigating potential integer overflows, a specific CVE was highlighted: CVE-2023-5869 (Mitre, 2023).

PostgreSQL has a multitude of CVEs related to its software that range from remote code execution to denial-of-service vulnerabilities. CVE-2023-5869 is an integer overflow vulnerability which was found in multiple versions of PostgreSQL. It has a CVSS score of 8.8, which is classified as high in severity rating. during SQL array value modification, an authenticated database user could execute arbitrary code through missing overflow checks. This method of integer overflow would be triggered by a remote user providing specifically crafted data as an input. Executing the arbitrary code would allow an unauthorised user to read and write to the server's memory (NIST, 2024). An unauthorised user viewing or modifying any data within the database would be a breach of data integrity and confidentiality.

To mitigate this vulnerability, the official recommendation is to apply updates to systems affected by this vulnerability as soon as possible. The current versions of PostgreSQL that have been updated to mitigate this issue are 16.1, 15.5, 14.10, 13.13, 12.17, and 11.22 (Red Hat Bugzilla, 2024). If the software had been tested with dynamic code analysis before being used in a live deployment, the specifically crafted data could have been flagged by a Fuzzer and discovered to be a vulnerability before it could have affected organisations in real world scenarios.

CVE-2023-5869 was found to be very similar to a previous integer overflow vulnerability, CVE-2021-32027 (PostgreSQL, 2021). The 2021 integer overflow vulnerability was also a security fault caused by missing overflow checks during SQL array value modification. Fixes which were used for CVE-2021-32027 also mitigated further damage being caused with the 2023 integer overflow vulnerability (PostgreSQL, 2023).

Using DAST would allow the development team to catch the specifically crafted data before it can become a security failure for ScottishGlen. Using the Fuzzer, the development team could create variants of the crafted data from CVE-2023-5869 to test for new integer overflow possibilities and catch them before a malicious hacker (such as the hacktivists) can discover them. This is especially important as new variants of specifically crafted data could keep appearing even in newly patched versions of PostgreSQL. As mentioned previously, CWE claims integer overflow as the 14th most stubborn weakness due to its constant reoccurrence within software even after patches have been applied.

4 Conclusion

In conclusion, this report detailed the potential improvements to security ScottishGlen could make by implementing continuous dynamic code analysis. With focus on CVE-2023-5869 and CVE-2021-32027 both of which were found in PostgreSQL, the advantages of implementing a dynamic code analysis were explained to aid in preventing and mitigating security faults before they are found by a malicious hacker. Dynamic code analysis was also effective in improving the optimisation of the code as well as preventing security faults. This will overall aid the organisation in improving their security posture.

5 References

Amazon Web Services, 2024. *What is PostgreSQL.* [Online] Available at: https://aws.amazon.com/rds/postgresql/what-is-postgresql/ [Accessed 2 February 2024].

CheckPoint, 2023. *What is Dynamic Code Analysis?.* [Online] Available at: https://www.checkpoint.com/cyber-hub/cloud-security/what-is-dynamic-code-analysis/
[Accessed 4 February 2024].

CWE, 2023. *2023 Stubborn Weaknesses.* [Online] Available at: https://cwe.mitre.org/top25/archive/2023/2023_stubborn_weaknesses.html [Accessed 3 February 2024].

CWE, 2023. *Trends in Real-World CWEs: 2019 to 2023.* [Online] Available at: https://cwe.mitre.org/top25/archive/2023/2023_trends.html [Accessed 3 February 2024].

Fronty, 2023. *What is Dynamic Code? Best tools for Dynamic Code Analysis.* [Online] Available at: https://fronty.com/what-is-dynamic-code-best-tools-for-dynamic-code-analysis/ [Accessed 16 February 2024].

HARMANCI, M. F., 2023. *What is Dynamic Code Analysis? How does it work?.* [Online] Available at: https://virgosol.com/en/blog/detail/what-is-dynamic-code-analysis-how-does-it-work [Accessed 16 February 2024].

Information Comminsioner's Office, 2023. *Penalties.* [Online] Available at: https://ico.org.uk/for-organisations/law-enforcement/guide-to-le-processing/penalties/#:~:text=The%20higher%20maximum%20amount%2C%20is,financial%20year%2C%20whichever%20is%20higher.
[Accessed 2 February 2024].

Mitre, 2023. *CVE-2023-5869.* [Online] Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-5869 [Accessed 17 February 2024].

nidecki, T. A., 2023. *What is Integer Overflow.* [Online] Available at: https://www.acunetix.com/blog/web-security-zone/what-is-integer-overflow/ [Accessed 3 February 2024].

NIST, 2023. *Vulnerability Detail Pages.* [Online] Available at: https://nvd.nist.gov/vuln/vulnerability-detail-pages [Accessed 2 February 2024].

NIST, 2024. [Online] Available at: https://nvd.nist.gov/vuln/detail/CVE-2023-5869 [Accessed 18 February 2024].

PostgreSQL, 2021. *CVE-2021-32027.* [Online] Available at: https://www.postgresql.org/support/security/CVE-2021-32027/ [Accessed 18 February 2024].

PostgreSQL, 2023. *PostgreSQL 16.1, 15.5, 14.10, 13.13, 12.17, and 11.22 Released!.* [Online] Available at: https://www.postgresql.org/about/news/postgresql-161-155-1410-1313-1217-and-1122-released-2749/
[Accessed 18 February 2024].

Red Hat Bugzilla, 2024. *Red Hat Bugzilla - Bug 2247169.* [Online] Available at: https://bugzilla.redhat.com/show_bug.cgi?id=2247169
[Accessed 18 February 2024].