Software Security Report Concerning ScottishGlen

Isaac Basque-Rice

BSc. (Hons.) Ethical Hacking
Abertay University
Dundee, United Kingdom
1901124@abertay.ac.uk

14th March, 2023
1991 Words

Contents

List of Figures

List of Acronyms

AD	Active Directory
CA	Certificate Authority
CERT	Computer Emergency Response Team
CNA	CVE Numbering Authority
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DN	Distinguished Name
DoS	Denial of Service
EIP	Extended Instruction Pointer
KDC	Key Distribution Center
MIT	Massachusetts Institute of Technology
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OWASP	Open Worldwide Application Security Project
SEI	Software Engineering Institute

1 Context

The energy company ScottishGlen is a small organisation within the energy sector. In response to a recent series of threats made against the organisation, via its employees, by a hacktivist group displeased by comments made by the CEO, the IT manager for ScottishGlen has been tasked with analysing the tech stack being used by the organisation from a security perspective. This is in order to mitigate, or ideally prevent, the hacktivist group from successfully carrying out their attack on the network.

ScottishGlen makes use of the Kerberos network authentication system, which has the potential to be exploited at some point in the future by a malicious actor and/or hacktivist group displeased with the organisation.

Kerberos is a popular system developed by the Massachusetts Institute of Technology (MIT) and implemented in client/server applications to allow users to authenticate their identity, normally for cases including email servers, file servers, Active Directory (AD), and other areas such as this (Katz 2021). Kerberos is "the default authorization technology used by Microsoft Windows" and additionally has functionality in Apple OS, UNIX, Linux, and FreeBSD (Buckbee 2020).

Naturally, due to the popularity of this system and its proximity to possibly valuable credentials, there are a significant number of security researchers analysing Kerberos, which in turn results in a considerable amount of vulnerabilities that can be found in the Common Vulnerabilities and Exposures (CVE) database. This section of the report will go into detail about a select number of these vulnerabilities that share a common classification, specifically Overflow vulnerabilities.

1.1 CVE Database

The CVE database (also called the National Vulnerability Database (NVD)) is a database maintained by the US Government's National Institute of Standards and Technology (NIST). It is a repository of reported known vulnerabilities categorised (in part) by classification, i.e. what kind of issue it is, and its severity, which will be touched upon in a following paragraph. The database itself also provides a brief description of the vulnerability in question as well as a number of references, links to Proofs of Concept, technical advisory documents, and other important information crucial to a full understanding of the issue in question.

Each CVE is issued by a trusted CVE Numbering Authority (CNA), such as IBM, Microsoft, Red Hat, and so on, and is given a unique identifier in the format `CVE-YYYY-XXXX`, where Y is the year the vulnerability was reported, and X is a further specific identifier.

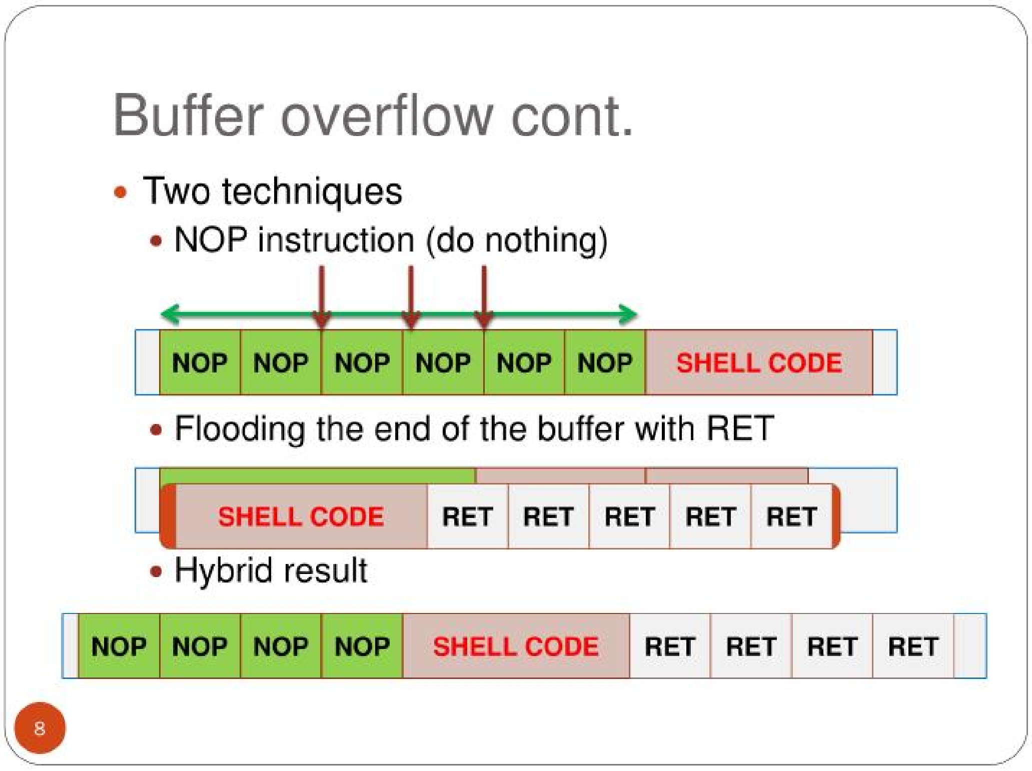
The CVE database makes use of the Common Vulnerability Scoring System (CVSS), which is a method of evaluating the severity of a vulnerability. This is a numerical scoring system represented on a scale of 0.0 to 10.0, where 10.0 is the most critical vulnerability. This system can also be represented in a more qualitative manner, where specific value ranges can be translated into either 'low', 'medium', 'high', or 'critical' "to help organizations properly assess and prioritize their vulnerability management processes." (Forum of Incident Response and Security Teams, Inc. n.d.).

1.2 Overflow Vulnerabilities

There are many different classifications of vulnerability, such as Injection, Man-in-the-Middle, Code Execution, Denial of Service (DoS), and so on. All of these vulnerabilities are exploited in different ways and therefore must be mitigated in different ways. As of the writing of this document, Kerberos has been the subject of a total of 302 vulnerabilities, of which 34 were Overflow vulnerabilities (MITRE n.d.).

An Overflow vulnerability, also called Buffer Overflows or Buffer Overruns, is a classification of vulnerability caused by an error in development. They occur when a 'buffer', which is a space in memory that persists for a relatively short period of time as a temporary storage area, holds data as it is being transferred from one location to another to perform an action (Christensson 2006). Most people are familiar with the concept of a buffer from streaming platforms such as YouTube or Spotify, where the services pre-load a section of the streaming media into a buffer to prevent minor downtime on a network from interrupting the viewing or listening experience.

Many computer programs make use of buffers in the same way, however, unlike media streaming platforms, the buffer is of finite size and can be overloaded. When a buffer is allocated more data than it is able to hold, or when data is placed past the buffer (OWASP Foundation n.d.). This can allow a malicious actor to insert their own 'shellcode', which in turn could result in arbitrary code execution on the device in which the vulnerable software is hosted. Figure 1 shows a high-level overview of how this is done.



Buffer overflow cont.

- Two techniques
  - NOP instruction (do nothing)
  - Flooding the end of the buffer with RET
  - Hybrid result

NOP NOP NOP NOP NOP NOP SHELL CODE
SHELL CODE RET RET RET RET RET
NOP NOP NOP NOP SHELL CODE RET RET RET RET
8

Figure 1: A slide showing how a buffer overflow works and how it can be used to execute malicious shellcode (Nickle@NSC 2014)

When an overflow has been performed the program in question crashes. At this stage, the shellcode is placed inside the Extended Instruction Pointer (EIP), which is responsible for executing any given instructions. Buffer Overflows are especially common in more 'low-level' languages, such as C or C++ due to a lack of checks at compile-time within the language itself for overrun exceptions (Pornin 2016).

One particularly famous example of a Buffer Overflow being exploited in the wild (and, indeed, the first), was the so-called 'Morris Worm'. This worm, created by Robert Tappan Morris, was distributed to the MIT campus in November of 1988 (Federal Bureau of Investigation n.d.). This worm made use of (amongst other things) a vulnerability in `finger`, which is a UNIX utility that "matches an e-mail address with the person who owns it and provides information about that person" (Shultz 2001).

This worm was primarily designed to demonstrate the alarmingly relaxed cybersecurity approach common at MIT at the time. However, due to a coding error that Morris made, the worm grew out of hand and had a DoS like effect on the network. Morris was indicted and found guilty under the newly minted Computer Fraud and Misuse act, making him the very first person convicted under this law (Federal Bureau of Investigation n.d.).

2 Recommendation

The onus is on the developers and technical staff at ScottishGlen to properly prevent and mitigate the possible exploitation of a Buffer Overflow vulnerability in the context of the Kerberos authentication system on their network. There are many ways they could go about this task, however, the recommendation of this report is for these developers to implement *secure coding practices*, as this is possibly the simplest and most straightforward option available, whilst still providing adequate levels of security.

The term 'secure coding practices' refers to a series of techniques and guidelines, designed to be implemented into the standard development lifecycle, that improve and maintain the code, and ensure the possible risks introduced by a given vulnerability are adequately mitigated (Turpin, Gadsden, and OWASP Foundation 2022).

The Secure Coding Practices guide published by the Open Worldwide Application Security Project (OWASP) provides a checklist of good practice in a tech-agnostic (i.e. irrespective of what technology is being used) and easily digestible format, therefore ensuring that the development team can implement the required items in a practical manner. The checklist format is especially helpful in this regard as it allows for an easily-identifiable and collaborative method of development. A developer could, for example, carry out one of the tasks in this checklist and then tick it off in a publicly-accessible version of the document.

As mentioned previously, the C programming language (in which Kerberos is written) is vulnerable to buffer overflow attacks if secure coding practices are not in place. The OWASP guide provides a section specifically regarding memory management, which provides advice on the usage of known-unsafe functions, validation of buffer sizes (both source and destination), and ensuring non-executable stacks are used where available.

NIST also provides a similar guide (Information Technology Laboratory Computer Security Division 2021), which similarly provides guidance around the methods that should be put in place to ensure secure coding practices. Where it departs from the OWASP guide however is that it does not implement a checklist system as such, instead preferring a framework consisting of four steps: Prepare the Organisation (PO), Protect Software (PS), Produce Well-Secured Software (PW), and Respond to Vulnerabilities (RV).

A particular practice that this guide provides that should be implemented in the organisation is the addition of input validation. This is the process of ensuring the user input into the software or system that is being or has been developed is valid and within the bounds of what is expected. When adequate validation has not been implemented, it can cause a number of issues across a wide array of software, from SQL Injection and Cross-Site Scripting in Web Development, to buffer overflows elsewhere.

Implementing proper input validation, whatever that may be in any given context, is not, on its own, an adequate preventative method for Buffer Overflow attacks. However, it is a crucial mitigation step for the prevention of less sophisticated attacks (so-called 'script kiddies', for example), and, when paired with other coding practises such as the ones mentioned earlier and others such as correct variable declaration will ensure the state of security in ScottishGlen will be greatly improved.

Finally, the implementation of frequent, thorough code reviews is crucial to the use of all of the recommendations mentioned herein. Without in-depth knowledge of the code base and the changes made to it in any given time frame, knowledge about how it could be vulnerable would be difficult to come by before it is detected by a professional security analyst or, worse, a malicious actor.

3 Implementation

Kerberos has historically been vulnerable to (to date) 37 known overflow issues (MITRE n.d.). What follows is a description of a vulnerability taken from the MITRE CVE database that may provide some insight into overflows in the context of Kerberos. After that will be a section detailing the steps that an organisation can take to mitigate this vulnerability, making use of the practices mentioned in the previous section.

3.1 CVE-2017-15088

This vulnerability affects Kerberos 5 up to and including version `1.15.2` and has a CVSS score of 9.8, which is 'critical' (NIST 2017). This is an issue in `pkinit_crypto_openssl.c`'s `get_matching_data()` and `X509_NAME_oneline_ex()` functions that limits Distinguished Name (DN) fields to a length of 256 bytes, which is "very small and can be easily overflowed" (kraynopp@km.ru 2017).

This vulnerability is triggered when the relevant Certificate Authority (CA)'s DN subject line is too long, this can result in a malicious actor leveraging that and crafting their own certificate that overflows the buffer and leads to arbitrary code execution and DoS. This vulnerability only affects users who run systems that have Kerberos functionality outside of the original MIT distribution, such as those using Red Hat Linux, which implements their own Key Distribution Center (KDC) certauth plugin (NIST 2017, Mariš and Buissart 2017).

3.2 Prevention and Mitigation

In the case of CVE-2017-15088, the usage of secure coding practices would mitigate the risk to the user quite considerably. In particular, input validation is a recommended course of action, as this vulnerability makes use of specially crafted input into the program as a leverage point.

Due to the fact that the input in question is not intended to be entered manually (as in, say, a web form), this vulnerability does not require input validation in the traditional sense. As input can be crafted manually, however, this area must be treated as if manual input is the norm, so to speak. When the input validation check is triggered, the program would, in theory, throw an exception and either continue to function (if that is possible) or stop entirely prior to any stage where exploitation of the buffer overflow is possible.

The Software Engineering Institute (SEI) Computer Emergency Response Team (CERT) C Coding Standards (Software Engineering Institute 2016) also outlines a number of secure coding practices, of which input validation forms a significant part. Section FIO30-C provides advice on excluding user input from format strings. Whilst the CA DN in question here is not a format string, this section does provide more high-level advice regarding user input into fields not intended to be reached by a user, specifically to make use of known-safe memory copying functions in place of unsafe ones.

To conclude, if input validation on the Kerberos network authentication system is not properly implemented, it could harm the state of security in ScottishGlen. While input validation alone may not be sufficient to enhance the company's security posture, when combined with other secure coding practices, it could significantly reduce the potential impact of CVE-2017-15088 on the company. The company should also continuously implement and update its current codebase in accordance with secure coding guidance through the usage of code reviews and other such methods, particularly for the C programming language. This is in order to decrease the risk of cyber-attacks on the organization such as the one threatened by the hacktivist group.

References

Buckbee, M. (Mar. 29, 2020). *Kerberos Authentication Explained.* URL: https://www.varonis.com/blog/kerberos-authentication-explained (visited on Feb. 6, 2023).

Christensson, P. (2006). *Buffer.* In: *TechTerms.* URL: https://techterms.com/definition/buffer (visited on Mar. 11, 2023).

Federal Bureau of Investigation (n.d.). *Morris Worm.* Federal Bureau of Investigation. URL: https://www.fbi.gov/history/famous-cases/morris-worm (visited on Mar. 11, 2023).

Forum of Incident Response and Security Teams, Inc. (n.d.). *Common Vulnerability Scoring System SIG.* FIRST — Forum of Incident Response and Security Teams. URL: https://www.first.org/cvss (visited on Mar. 9, 2023).

Information Technology Laboratory Computer Security Division (Feb. 25, 2021). *Secure Software Development Framework — CSRC — CSRC.* CSRC — NIST. URL: https://csrc.nist.gov/Projects/ssdf (visited on Mar. 6, 2023).

Katz, A. (July 19, 2021). *How Does Kerberos Work? The Authentication Protocol Explained.* freeCodeCamp.org. URL: https://www.freecodecamp.org/news/how-does-kerberos-work-authentication-protocol/ (visited on Feb. 2, 2023).

kraynopp@km.ru (Aug. 10, 2017). *#871698 - Krb5: CVE-2017-15088: Buffer Overflow in Get_matching_data() - Debian Bug Report Logs.* In collab. with B. Kaduk et al. E-mail. URL: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=871698 (visited on Feb. 9, 2023).

Mariš, A. and Buissart, C. (Oct. 19, 2017). *1504045 – (CVE-2017-15088) CVE-2017-15088 Krb5: Buffer Overflow in Get_matching_data().* URL: https://bugzilla.redhat.com/show_bug.cgi?id=1504045 (visited on Feb. 9, 2023).

MITRE (n.d.). *CVE - Search Results.* URL: https://web.archive.org/web/20230309210309/https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Kerberos (visited on Feb. 7, 2023).

Nickle@NSC (Aug. 20, 2014). "Buffer Overflow Overview". URL: https://www.slideserve.com/tovi/buffer-overflow-overview (visited on Feb. 7, 2023).

NIST (Nov. 23, 2017). *NVD - CVE-2017-15088.* National Vulnerability Database. URL: https://nvd.nist.gov/vuln/detail/CVE-2017-15088 (visited on Feb. 9, 2023).

OWASP Foundation (n.d.). *Buffer Overflow — OWASP Foundation.* URL: https://owasp.org/www-community/vulnerabilities/Buffer_Overflow (visited on Feb. 7, 2023).

Pornin, T. (Feb. 23, 2016). *Answer to "Why Are Programs Written in C and C++ so Frequently Vulnerable to Overflow Attacks?"* Information Security Stack Exchange. URL: https://security.stackexchange.com/a/115508 (visited on Feb. 7, 2023).

Shultz, G. (June 20, 2001). *Everything You Need to Know about TCP/IP?S Finger Utility.* TechRepublic. URL: https://www.techrepublic.com/article/everything-you-need-to-know-about-tcp-ips-finger-utility/ (visited on Mar. 11, 2023).

Software Engineering Institute (2016). *CEI CERT C Coding Standard - Rules for Developing Safe, Reliable, and Secure Systems.* URL: https://resources.sei.cmu.edu/downloads/secure-coding/assets/sei-cert-c-coding-standard-2016-v01.pdf.

Turpin, K., Gadsden, J., and OWASP Foundation (Dec. 2022). *OWASP Secure Coding Practices-Quick Reference Guide — OWASP Foundation.* URL: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/ (visited on Feb. 23, 2023).