International Journal of Research in Advanced Engineering and Technology
www.newengineeringjournal.in
ISSN: 2455-0876
Received: 19-11-2022, Accepted: 05-12-2022, Published: 21-12-2022
Volume 8, Issue 2, 2022, Page No. 26-30

Security concerns of C++ & Java

Dr. Omar Amer Abouabdalla, Manar Chikh Basatna, Salma Jahan, Mahimur Rahman Khan
Faculty of Information & Communication Technology, Limkokwing University of Creative Technology, Inovasi, Jalan Teknokrat, Cyberjaya, Selangor, Malaysia

Abstract
Computer programming is the core of computer science, and the most popular and commonly used languages in Computer programming are C++ and Java. With the recent rise of the software sector, more people are becoming curious to enhance their skills by learning programming languages. However, while there are over 500 programming languages accessible today, only a few of them are extensively used. Some of the security concerns of these two programming languages were summed up and compared. According to the security concerns and the results of the experiment, the research mentioned the important security concerns in these two of the most popular programming languages. For performance-critical applications that require speed and effective memory management, C++ is a popular language. It's employed in a variety of fields such as software and game development, virtual reality, robotics, and scientific computing. Java is a programming language that may be used to create apps for a variety of platforms. Java is used on desktops, servers, mobile phones, tablets, Blu-ray players, televisions, and web browsers, and Java-based applications can be written for any of these platforms. For embedded device programming, we use C++. In other words, C++ almost always employs a one-time programming code.Java, on the other hand, is not intended for one-time programming. Java is used in applications that can be further developed or upgraded based on the needs and requirements.

Keywords: java, C++, programming language, security

Introduction
In our current generation, almost nothing is possible without the existence of programming languages. Programming languages are tremendously crucial and decisive because they define the relationship, semantics, and grammar that allows programmers to communicate effectively with devices and machines. Programming is extremely helpful as well as influential for comprehending in addition to learning to innovate, establish an eco-friendly solution to a global problem, additionally to augment and increase the intensity of computers and the internet in our daily lives. Programming is indispensable for accelerating a machine's input and output processes. It is also critical to accurately automate, gather, handle, evaluate, and analyze data and information processing. Today's industry employs dozens of programming languages.
C++ is predominantly an extension of the C language and is a general-purpose, object-oriented, middle-level programming language. C++ is an example of a hybrid language due to the fact that coding can be performed in multiple formats based on various situations. It is a programming language that is used to create software programs and packaged software such as games, desktop apps, animations, film makers, in addition to many more operating systems. C++ was basically designed with a focus on productivity, reliability, and adaptivity of use, with an emphasis on systems programming and embedded, resource-constrained software as well as large systems. C++ has also proven useful in a variety of other contexts, with key strengths in software infrastructure and resource-constrained applications.
Java is basically a high-level, object-oriented, class based programming language with a low number of implementation dependencies. Java is basically one of the most popular programming languages, owing to its compatibility. Java can be used for a variety of tasks, including software development, mobile application development and large-scale system development.Java was used to create Wiki pages, Uber, Minecraft, LinkedIn, Android OS and the Mars Rover Controller. [8] Java is fundamentally a computing platform for developing applications. As a result, Java is regarded as one of the fastest, safest, and most dependable programming languages used by most organizations to build their projects.

Background
History of C++
C++ was previously created by Bjarne Stroustrup at Bell Laboratories beginning in 1979. Since C++ is an attempt to add object-oriented features along with other improvements to C, it was previously known as "C with Objects." Stroustrup named the language C++ in 1983 as it evolved afterwards with time. C++ stands for "C incremented". In 1989, the American National Standards Institute (ANSI) established a committee for (exact

description of computer language) C++. In 1995, the first draft standards were published. (Object Oriented Programming with C++, Second Edition, 2022)
C++ evolved slowly after C++98 until the C++11 standard was released in 2011, adding numerous new features, expanding the standard library, and providing more facilities to C++ programmers. C++ was ranked fourth on the TIOBE index, a measure of programming language popularity, in 2022, after Python, C, and Java. (Stroustrup: FAQ, 2022)

Table 1

Version	Date of Release	Updates
C++ 98 (ISO/IEC 14882:1998)	October, 1998	The ever first version published
C++ 03 (ISO/IEC 14882:2003)	February, 2003	Value initialization is introduced
C++ 11	August, 2011	Nullptr, Delegating Constructors, Uniform Initialization Syntax, Lambda expressions, Automatic Type Deduction and Decltype, and Rvalue References are all introduced
C++ 14	August, 2014	Polymorphic lambdas, digit separators, generalized lambda capture, and variable templates are all introduced
C++ 17	December, 2017	Fold expressions and hexadecimal floating point literals are introduced
C++ 20	December, 2020	Time zone and calendar library, std::string functions

The C++ programming language came in many versions. These language versions are compiler implementations based on specifications created by the ISO C++ community, which oversees the language's development.

History of Java
Java as per said is an Object-Oriented programming language. Java was originally created in the early 1990s by the man named James Gosling. The team started this project to create a language for digital devices like set-top boxes and televisions. Java, by James Gosling and his team, held a group discussion and came up with several names such as SILK, RUBY, JAVA, DNA, and so on and so forth. Because Java's name was so unique, it was decided after much debate. Java gets its name from a type of coffee bean called Java. TIME MAGAZINE named Java one of the Ten Best Products of 1995, based on principles such as Robust, Portable, Platform Independent, High Performance, Multithread, and so on.
Java is currently used in web programming, mobile platforms, game modes, e-business solutions, and so on and so forth. Since JDK 1.0, Java has undergone a few changes. In addition to language changes, the Java Class Library has undergone significant transformations over the years, growing from a few hundred classes in JDK 1.0 to over three thousand in J2SE 5.

Table 2

Java SE Version Name	Version Number	Date of Release
JDK 1.0 “OAK”	1.0	January, 1996
JDK 1.1	1.1	February, 1997
J2SE 1.2 “Playground”	1.2	December, 1998
J2SE 1.3 “Kestrel”	1.3	May, 2000
J2SE 1.4 “Merlin”	1.4	February, 2002
J2SE 5.0 “Tiger”	1.5	September, 2004
Java SE 6 “Mustang”	1.6	December, 2006
Java SE 7 “Dolphin”	1.7	July, 2011
Java SE 8	1.8	March, 2014
Java SE 9	9	September, 2017
Java SE 10	10	March, 2018
Java SE 11	11	September, 2018
Java SE 12	12	March, 2019
Java SE 13	13	September, 2019
Java SE 14	14	March, 2020
Java SE 15	15	September, 2020
Java SE 16	16	March, 2021
Java SE 17	17	September, 2021
Java SE 18	18	March, 2022

We can see from the table above that the naming and version number have changed over time:
- Versions 1.0 and 1.1 are referred to as JDK (Java Development Kit).
- The platform is known as J2SE from version 1.2 to 1.4. (Java 2 Standard Edition).
- Sun introduces internal and external versions beginning with version 1.5. The internal version continues from the previous one (1.5 after 1.4), but the external version has a significant jump (5.0 for 1.5). This may

cause some confusion, so keep in mind that version 1.5 and version 5.0 are simply two different version names for the same thing.
- From Java 6 and onwards, the version name became Java SE.

Major versions were released every two years, but Java SE 7 took 5 years after its predecessor, Java SE 6, and 3 years after that for Java SE 8 to be available to the public.

Related Works

Many studies have been conducted to compare Java and C++.

A Memory corruption vulnerability is a special type of fault in software that could lead to unintentional modification to the content at a memory location and thus compromise the data dependency of a running program. In attempting to exploit a buffer overflow vulnerability, for example, an attacker typically overwrites adjacent memory locations [1].

Java vulnerabilities can happen in many different components of Java. For example, in a runtime environment, deserialization, scripting, and concurrency components have in the past been known to contain critical vulnerabilities. In plugins, the Java Deployment Toolkit and Java Web Start have been popular targets [2].

Memory corruption vulnerabilities in C/C++ applications enable attackers to execute code, change data, and leak information. These attacks rely on the fact that C/C++ require the programmer to manually enforce spatial safety (bounds checks) and temporal safety (lifetime checks) [3].

The platform defines APIs spanning major security areas, including cryptography, access control, and secure communication [8]. The Java Cryptography Architecture (JCA) contains APIs for cryptographic hashes, keys and certificates, digital signatures, and encryption [9]. Nine cryptographic engines are defined to provide either cryptographic operations (encryption, digital signatures, hashes), generators or converters of cryptographic material (keys and algorithm parameters), or objects (keystores or certificates) that encapsulate the cryptographic data [7].

When an array is declared in C, space is reserved for it, and the array is manipulated by means of a pointer to the first byte. At runtime, no information about the array size is available, and most C compilers will generate code that will allow a program to copy data beyond the end of an array, thereby overwriting adjacent memory space. If interesting information is stored somewhere in such adjacent memory space, it could be possible for an attacker to overwrite it. On the stack, this is usually the case: it stores the addresses to resume execution after a function call has completed its execution [4].

Java provides security and robustness by building a high level security model atop the foundation of memory protection. Unfortunately, any native code linked into a Java program– including the million lines used to implement the standard library– is able to bypass both the memory protection and the higher-level policies. We present a hardware assisted implementation of the Java native code interface, which extends the guarantees required for Java's security model to native code [10].

It describes subtler buffer overflow problems 2 caused by common C functions and even by their safer alternatives [11]. Buffer overflows can be prevented by taking measures such as writing secure code, performing bound checking, static and dynamic code analysis and runtime code instrumentation [12].

Buffer overflows are commonly associated with C/ C++ programs since these languages do not provide any built-in protection against accessing and overwriting of data in memory. Many C/C++ functions such as gets, printf, strcpy, strcat, etc. are susceptible to buffer overflows and thus compromise the security of the application. [13].

Heap memory is dynamically allocated at run-time by the application. The buffer overflow, which is usually exploited on the stack, is also possible in this kind of memory. However exploitation of such heap-based buffer overflows usually relies on finding either function pointers or by performing an indirect pointer attack on data pointers in this memory area, but these pointers are not always present [14].

Comparison

Memory Corruption

Memory Corruption can be identified as a type of vulnerability that might transpire in an application due to memory alteration without any explicit assignment. Memory corruption is the most prevalent type of high-severity rule in C++. Because Java's type system prevents memory corruption, which includes vulnerabilities such as buffer overflows, format-string vulnerabilities, and use-after-free errors, there are no analogous rules. However, exploiting memory corruption in C programs such as C++ has become more difficult due to the introduction of memory-protection technologies such as address space layout randomization (ASLR) and data execution prevention (DEP).

Injection

Injection flaw is a sort of drawback which permits a hacker or attacker to transfer malicious code with the usage of an application within a system. It refers to an attacker's ability to execute malicious code in a language other than C++ or Java. SQL is the language used for SQL injection, and HTML, which can include Flash or JavaScript, is used for XSS. ENV33-C and ENV34-C are the two C rules that cover injection. Because Java includes more subsystems than C++, it has more injection rules. SQL injection, for example, is possible in both

C++ and Java, but only Java provides a standard library for connecting to SQL databases (the JDBC); thus, only Java has a SQL injection rule.

Privilege Escalation

Privilege escalation falls in the category of network attack which is used to acquire unauthorized access to any system within the perimeter of security. Both C++ and Java have privilege escalation rules, but the privileges differ significantly between the two languages. Java has an internal privilege system that allows some code in the same program to run without restrictions, while other code in the same program can be denied certain privileges, such as the ability to write a file to disc or send data over the network. Applets typically have limited privilege, and the most widely publicized Java exploits exploited Java's internal privilege model, granting themselves the same privilege as desktop Java applications. As a result, the Java rules address the issue of internal privilege escalation (IPE), which undermines Java's internal security model. C++, on the other hand, lacks an internal privilege model; there is no feature in C++ that a program can use to limit what another portion of the program can do.

Internal data management

Pointer values are used in programming languages like C and C++ to manage application memory and protect data. Hackers can utilize pointers to gain access to confidential information, which is unfortunate. Users who request data cannot be verified by pointers. As a result, pointers will allow a hacker to access memory without first verifying their authorization. Java, on the other hand, blocks any illegal data access using its own internal memory and data management methods. Infiltrating these systems is substantially more challenging.

Pointers are used in C++, which can lead to illegal access to memory blocks if they are obtained by other applications. The no pointer policy in Java ensures that access is only granted after authorization. It also has its own memory management system built in.

Leftovers

Only seven C++ rules and nine Java rules are found in the remaining categories. In other words, these categories only provide Java vulnerabilities that are also present in C++. JNI03-J is the single Java rule governing C++ code execution. This is the first Java Native Interface (JNI) rule, and it did not fit into any other category. Because it describes JNI code, which is typically written in C++, the rule is of high severity.

Undefined Behavior

Because C++ is intended for compiler writers and platform builders, this definition allows writers and builders to allow a programme that exhibits undefined behavior to do whatever it wants, including allowing itself to be hacked.

MET00-J are the two Java rules that are classified as unexpected behavior. Method arguments and MSC02-J must be validated. Create a strong random number generator. Java does not have a concept of undefined behavior. These rules are of high severity because an attacker can use violations of them to circumvent an authentication mechanism and gain elevated privileges. Both of these issues can occur in C++ and Java code.

Evaluation

The preceding analysis shows that all of the high-severity Java rules also apply to C code, with the exception of those in Java's most serious category, internal privilege escalation (IPE). C++ is not eligible for IPE because it lacks an internal privilege model. We should also mention that memory corruption is C++'s most serious security flaw, which has no effect on Java code. Finally, Java has a slightly lower proportion of high-severity rules than C++.

Consider the scope of these rules now. Some rules are simply not applicable to many programs. Concurrency rules in C++ and Java, for example, are inapplicable to single-threaded programs. As a result, a developer writing a single-threaded program must follow fewer secure coding rules than a developer writing a multithreaded program.

C++ code that is privileged by a platform but must interact with unprivileged code is the closest analogue provided by C++. This would apply to UNIX programs with root privileges or Windows programs with administrative privileges. As a result, if you're writing unprivileged C++ code, you can ignore C++'s four EPE rules.

Conclusion

The security comparison of Java vs C++ is now comprehensive. We have seen that C++ and Java share many similarities. Both are also well secured when in use.

However, as previously stated, there are some significant security differences between C++ and Java. pointers, memory management, injection, and other significant differences are examples. However, when it comes to real-world applications of C++ and Java, there is a significant difference between these programming languages.

For embedded device programming, we use C++. In other words, C++ almost always employs a one-time programming code. For example, the code in your dishwasher, fridges, television, and so on and so forth. Java,

on the other hand, is not intended for one-time programming. Java code is used in applications that can be further developed or upgraded based on the needs and requirements.

References

1. Xu J, Mu D, Chen P, Xing X, Wang P, Liu P. CREDAL. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016. https://doi.org/10.1145/2976749.2978340
2. J Wook. Recent Java exploitation trends and malware. *Black Hat USA 2012 Las Vegas*, 2016. https://dl.acm.org/doi/pdf/10.1145/3196494.3196540
3. Burow N, McKee D, Carr SA, Payer M. CUP, 2018. *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. https://doi.org/10.1145/3196494.3196540
4. Younan Y, Joosen W, Piessens F. Runtime countermeasures for code injection attacks against C and C++ programs. ACM Computing Surveys,2012:44(3):1–28. https://doi.org/10.1145/2187671.2187679
5. Bicknell A. *6 reasons why Java is more secure than other languages*. Educative: Interactive Courses for Software Developers, 2019. Retrieved 2022, from https://www.educative.io/blog/why-java-is-more-secure-than-other-languages
6. Svoboda D. (2015, October 5). *Is Java More Secure than C?* SEI Blog. Retrieved 2022, from https://insights.sei.cmu.edu/blog/is-java-more-secure-than-c/
7. Meng N, Nagy S, Yao D, Zhuang D, Argoty W, GA. Secure coding practices in java. Proceedings of the 40th International Conference on Software Engineering, 2018. https://doi.org/10.1145/3180155.3180201
8. Javasecurityoverview.http://docs.oracle.com/javase/8/docs/technotes/guides/ security/overview/jsoverview.html
9. Javacryptographyarchitecture.http://docs.oracle.com/javase/7/docs/technotes/ guides/security/crypto/CryptoSpec.html.
10. Chisnall D, Davis B, Gudka K, Brazdil D, Joannou A, Woodruff J, *et al*. CHERI JNI. *ACM SIGARCH* Computer Architecture News,2017:45(1):569-583.
11. Wagner D, Foster J, Brewer E, Aiken A. A first step towards automated detection of buffer overrun vulnerabilities. Proceedings of the Year Network and Distributed System Security Symposium (NDSS), San Diego, CA, 2000, 3–17.
12. Fu D, Shi F. Buffer overflow exploit and defensive techniques. IEEE International Conference on Multimedia Information Networking and Security (MINES); Nanjing, China, 2012, 87–90.
13. Ramasamy S, Singh A, Singal D. Enhancing the Security of C/C++ Programs using Static Analysis. Indian Journal of Science and Technology, 2016, 9(44).
14. Bulba and Kil3r. Bypassing Stackguard and stackshield. Phrack, 2000, 56.