Software and System Security Final Assessment

Mark Hinds 2001606

Tashema Mitchell 2000913

Courtnie-Ann Robinson 2002369

Jevaughn Salmon 2001369

School of Computing and Information Technology, University of Technology, Jamaica

CIT3030: Software and System Security

Lecturer: Miss Tanya Graham

November 30, 2022

Table of Contents

Introduction

Java was first released in 1995 by Sun Microsystems with the aim to “write once, run everywhere”. It is an object-oriented programming language and software platform that has been developed to power a significant portion of the digital world, offering a dependable platform on which several services and applications are built (Java, n.d.). An advantage of Java is its portability, which makes it easy to move Java code despite what it was written on. Though being introduced for more than two decades, it is still the most popular programming language for application software development (IBM, 2019).

What are Vulnerabilities?

A vulnerability is a flaw in a computer system that an attacker can exploit to launch a successful attack. They can happen because of flaws, features, or malfunctions, and intruders will try to take advantage of any of them, often by combining two or more.

Vulnerabilities in Java

Connection String Injection

Connection string parameter pollution (CSPP) refers to the use of a single practice to target or exploit uses multiple connection strings. If a system is infected, it may have multiple open connections. This means that an attacker could use flaws in any of the connected applications to gain access to files and systems in multiple organizations. Connection string injection is a type of attack that comes from the CSPP attack. Connection string injection attacks can happen when user input is used to build connection strings using dynamic string concatenation. Attackers use SQL commands to try to change the data that is stored on servers. If the string isn't checked and harmful content isn't "escaped," an attacker could get access to server resources like confidential data if the string isn't checked and harmful content isn't "escaped." This would lead to a loss of availability and integrity of information. With database connectivity and objectivity, connection string injection could change how a web application connects to its database. It can also help attackers avoid being found and get around firewalls.

Prevention

- Avoid adding user information such as a username and password, in connection strings.
- Use Secure String classes
- Parsing: Hosts should compare the variable string against a list of data and replace it with a known good value based on the parsed result.
- Caching: When a request is made, and a variable is found to be malicious, the data should be cached and returned for subsequent requests.
- Using encryption algorithm, such as AES or Two fish to make it difficult for attackers to interfere with responses
- Limiting Access by using IP address or requiring authentication to help reduce risks
- Auditing to record malicious traffic and attempts to gain access to systems.

Lightweight Directory Access Protocol String Injection

LDAP string injection is an attack used to exploit web applications that uses user input to build LDAP statements. When an application does not handle user input properly, it can be changed. It takes advantage of a weakness that is used to break the authentication process that websites use. Most times, these kinds of attacks happen to organizations that uses user data to build LDAP statements. LDAP makes bad queries to access a database and change the data there. By changing the contents of the query, attackers can change how it works and what it does. Using LDAP, an attacker can get into the directory and see information they are authorized to. They can also change information and resources in the LDAP statements and tree. Attackers can also use web applications that uses user input to make LDAP statements. These kinds of LDAP attacks affect how information is kept private, how well it works, and how easy it is to find. This makes shareholders less likely to trust the information.

Prevention

- Enforce input validation: the input should be validated against a prefer list of allowed strings or characters.
- Escape input with encoding: Escape user-controlled input strings in such a way that any control characters in the input don't change the intended meaning of the LDAP search filter.
- Configure LDAP to restrict unauthorized users to make any malicious changes to the system.

Reflected XXS

A reflected XSS attack, also called a non-persistent attack, happens when a malicious script is reflected from an online application and onto the victim's web. The attack happens when a request comes in that isn't cautiously managed. Usually, the request starts with a link that lets the attacker change how the website works and run malicious scripts. It is often used as anchor text in an email, which sends an XSS request to the exploited website and shows the attack to the user. For a reflected attack to work, attackers look for situations where input data is directly used to make a response. It often has things in it that are not meant to host scripts. Most of the time, these contents are not checked or filtered by authentication mechanism, output encoding, or other methods. Reflected XSS can be used to gain control of client computers on the server, overtaking user accounts, steal credentials, leak confidential information.

Prevention

- Sanitize inputs and check variables- variables are the building blocks of LDAP filters, hackers use special characters in parameters to create malicious injections. All values which make the LDAP filter should be checked against a list of valid values in the Application Layer before the LDAP receives the query.
- Don’t Construct Filters by Concatenating Strings- Avoid creating LDAP search filters by concatenating strings, if the string contains a user input.
- Use Access Control on the LDAP Server- follow the principle of least privilege to add another layer of protection, ensuring that each account only has permission to perform operations needed for the user’s role.
- Set the base DN and scope of the LDAP server to match as closely as possible to the type of search being performed.
- Use a size limit to prevent the server from returning more items than expected. For example, when searching for individual user entries, if the search returns more than one entry, this will result in an error.
- Use timeouts to make sure your server doesn’t spend too much time processing searches. For most searches, a timeout of 1-2 seconds is sufficient. Set a timeout that is appropriate given your current search behavior and server performance, and you can avoid malicious searches querying large amounts of data, which may take more time.

XPATH injection

XPATH Injections creates a malicious XPath expression that allows them to read data they should not be able to read. An attacker can find out how the XML data is organized or obtain data that they may not be able to elevate their privileges on the website if the XML data is being used for authentication if the XML data is being used for authentication. XPATH attacks compromise information confidentiality, integrity, and availability, reducing the dependability of information from shareholders.

Prevention

- Use a parameterized XPath interface if applicable
- Escape the user input to make it safe to include in a dynamically constructed query
- Use a precompiled XPath query.

Second order SQL injection

A second-order SQL injection occurs when an attacker injected input is saved in the code and is subsequently used without sufficient management in a new SQL query. This type of SQL injection is known as "store and forward." When the data that was provided by the user is used by the application or any other resource and the injected code that was provided by the attacker is triggered because of successful exploitation, the data that was provided by the user becomes a threat. Integrity and confidentiality of information will both be compromised if an organization is vulnerable to second-order SQL injection. These are the impacts that it will have on the organization.

Prevention

- Input validation- this involves validating all user input to ensure that it does not contain malicious code.
- Parameterized queries to protect your database from SQL Injection attacks. Prepared statements use placeholders for user input, which prevents the input from being executed as SQL code.
- Web application firewall to filter out SQL Injection attacks.

Tools needed to initiate attacks

Dark Net

The dark net, also known as the dark web, is an internet network that can only be accessed through specific software or protocols. Tor Browser, an app from the Tor Project, can be used to access the dark web. Even though Tor Browser can be used to create an attack like XPath injection into codes, it has built-in protection for many types of tracking and deanonymization features that make it favorable for an attacker when carrying out an attack. An example of Xpath code in the dark web is `SELECT * FROM users WHERE user='<INJECT>' and pass='<MD5 OF PASS>'`

Hypertext Transfer protocol

HTTP is a set of rules for transferring files such as text, and other multimedia files over the internet. When a user opens a web browser, they are using HTTP indirectly. HTTP is an application protocol that runs on top of the TCP/IP protocol suite, which is the foundation of the internet. An example of reflected attack using http: https://insecure-website.com/search?term=gift

SQL Map

SQLmap is an open-source tool for detecting and exploiting SQL injection flaws in penetration testing. SQLmap automates the detection and exploitation of SQL injection. SQL can be used to both launch and detect malicious attacks such as SQL second order injections. An example of second order injection in SQL map:

```
#Get the SQL payload execution with a GET to a url
sqlmap -r login.txt -p username --second-url "http://10.10.10.10/details.php"
#Get the SQL payload execution sending a custom request from a file
```

sqlmap -r login.txt -p username --second-req details.txt

Blind SQL Hacker

BSQL Hacker is a tool for SQL injections that is made to take advantage of SQL injections. This tool takes all a database's schema and data mode and pulls them out automatically. This tool can be used to do different kinds of SQL attacks, such as connection string injection. An example:

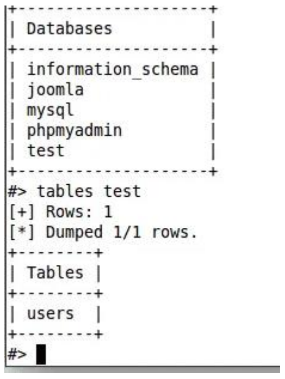
string userName = ctx.getAuthenticatedUserName();

string query = "SELECT * FROM items WHERE owner = "'" + userName + "' AND itemname = '" + ItemName.Text + "'";

The Mole

The Mole is an automatic SQL injection exploitation tool. If you give it a vulnerable URL and a valid string on the site, it can find the injection and fully utilize it, either through use of the union technique or a Boolean query-based technique. Through the get and post methods, it uses SQL injections perform malicious requests.

An example:



```
+--------------------+
| Databases          |
+--------------------+
| information_schema |
| joomla             |
| mysql              |
| phpmyadmin         |
| test               |
+--------------------+
#> tables test
[+] Rows: 1
[*] Dumped 1/1 rows.
+--------+
| Tables |
+--------+
| users  |
+--------+
#>
```

Secure Java Programming Practices

Being the fact that secure designs are required to produce secure programs, the best designs can however, result in poor security programs if developers are unaware of the many security downfalls ingrained in Java programming. Developing secure Java code should adhere to coding standards and best practices, leaving little to no room for exploits or malicious attacks. In recent years, even large organizations such as eBay, the CIA, and the IRS have been victimized by vulnerabilities in their applications that attackers have discovered and exploited. Below are a few safe programming practices for Java:

Using Trusted Libraries

A programming library contains prewritten code(s) that can be used by programmers to improve/build tasks such as apps and websites. This collection of reusable code is typically designed to address specific common problems. However, libraries can contain vulnerabilities which allows attackers to maliciously exploit applications. It is therefore implored that proven and up-to-date libraries are used by the developer(s) for the sake of trustworthiness, already entrusted to them by reputable companies/organizations.

Steer Clear of Serialization

The process of converting an object into a stream of bytes in order to store or transmit the object to memory, a database, or a file is known as serialization. Its primary function is to save the state of an object so that it can be recreated when needed. Any application that accepts serialized Java objects is vulnerable, even if the vulnerability is caused by a library or framework rather than your own Java code. Making an interface serializable without considering what might be exposed is one thing to avoid. Another trap to avoid is accidentally serializing a security-sensitive class by subclassing or implementing a serializable interface.

Prevent Injection Attacks

When malicious code is injected into the network, an injection attack occurs. An injection attack can include the following:

- SQL Injection- When developers write dynamic database queries that allow for user input, an attacker can include SQL commands in any screen input field's input data. The application then executes the malevolent SQL in the database due to a flaw in the code. This allows attackers to get around the application's authentication functionality and retrieve the contents of an entire database.
- XPath Injection- XPath injections, like SQL injections, can target websites that use user-supplied information to construct an XPath query for XML data. By sending malicious data to the website, an attacker can obtain detailed information on how the XML data is structured or access data that is not normally accessible.
- Cross-Site Scripting- Vulnerabilities that enable these attacks can occur anywhere a web application receives user input and generates output without validating or encoding it. To keep Java code applications secure and prevent XSS, filter your inputs with a whitelist of allowed characters and HTML encode your output for HTML contexts using a proven library.

Hash User Passwords

Passwords should never be saved in plain text. Always salt user passwords and use a recommended hashing algorithm like SHA-2. When a password is 'hashed,' it becomes a scrambled version of itself. The hash value is derived from a combination of the password and a predefined key known to the application using a hashing algorithm.

Images and steps of attacks being carry out

Conclusion

A computer system's vulnerability is a defect an attacker can exploit. Intruders will exploit faults, features, or failures. Java is the most used application-development language. LDAP produces terrible database queries to modify data. By modifying the query, attackers can alter its behavior. LDAP lets an attacker see approved directory information. SQL injection compromises data integrity and confidentiality. The goal is to get the server to run commands that give it database access. An attacker must understand software network design to attack. This article examines how an attacker could use string injection to access a web application without sufficient input sanitization or validation. Unsanitized data on a website can lead to an XSS attack. An attacker can send phishing emails with links to site users. The dark web requires special software or protocols to access. Java programming should follow coding standards and best practices to prevent exploits and attacks. In recent years, attackers have exploited application vulnerabilities at eBay, the CIA, and the IRS. Programmers can utilize a programming library to improve/build apps and websites. This reusable code addresses common issues. For trustworthiness, choose proven, up-to-date libraries. These vulnerabilities occur when a web application gets user input and creates output without verifying or encoding it. Filter Java code inputs using a whitelist of acceptable characters and encode HTML output with an established library to prevent XSS.

References

Dizdar. (2022, April 8). *SQL Injection Attack: Real Life Attacks and Code Examples*. Bright Security. https://brightsec.com/blog/sql-injection-attack/

GeeksforGeeks. (2022, August 22). *How a connection string injection attack is performed?* https://www.geeksforgeeks.org/how-a-connection-string-injection-attack-is-performed/

GeeksforGeeks. (2022, July 21). *What is connection string parameter pollution?* https://www.geeksforgeeks.org/what-is-connection-string-parameter-pollution/

How a connection string injection attack is performed? (2022, August 22). GeeksforGeeks. https://www.geeksforgeeks.org/how-a-connection-string-injection-attack-is-performed/

IBM. (2019, May 8). *What is Java?* https://www.ibm.com/cloud/learn/java-explained

Imperva. (2022, February 15). *Reflected XSS attacks.* https://www.imperva.com/learn/application-security/reflected-xss-attacks/#:~:text=Reflected%20XSS%20attacks%2C%20also%20known,enables%20execution%20of%20malicious%20scripts

Java. (n.d.). *What is Java and why do I need it?* https://www.java.com/en/download/help/whatis_java.html

Koussa, S. (2020, February 5). *LDAP injection primer for Java developers*. Reshift Security. https://www.reshiftsecurity.com/ldap-injection-primer-for-java-developers/#:~:text=Examples%20of%20LDAP%20Injection%20In%20this%20example%20there,LDAP%20search%20filter%20to%20confirm%20the%20login%20credentials

Kumar, M. (2012, October 8). *The mole - Another automatic SQL injection exploitation tool*. The Hacker News. https://thehackernews.com/2011/12/mole-another-automatic-sql-injection.html#:~:text=The%20Mole%20is%20an%20automatic,a%20boolean%20query%20based%20technique.&text=Support%20for%20injections%20using%20Mysql,Server%2C%20Postgres%20and%20Oracle%20databases

Lutkevich, B. (n.d.). *What is an LDAP injection?* TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/LDAP-injection

Meltzer. (2022, July 29). *What is a Programming Library? A Beginner's Guide (2022)*. CareerFoundry. Retrieved November 22, 2022, from https://careerfoundry.com/en/blog/web-development/programming-library-guide/

OWASP Foundation. (n.d.). *XPATH injection*. https://owasp.org/www-community/attacks/XPATH_Injection

Research Team. (2022, January 15). *Most common vulnerabilities in Java and how to fix.* Offensive 360. https://offensive360.com/most-common-vulnerabilities-in-java-how-to-fix/

Singh. (2021, July 12). *Reflected XSS Vulnerability in Depth - GeeksforGeeks*. GeeksforGeeks. Retrieved November 22, 2022, from https://www.geeksforgeeks.org/reflected-xss-vulnerability-in-depth/

Soroker, T. (2021, February 9). *Best Practices for Writing Secure Java Code - Coralogix*. Coralogix. Retrieved November 22, 2022, from https://coralogix.com/blog/best-practices-for-writing-secure-java-code/

Synopsys. (n.d.). *What is LDAP injection and how does it work?* https://www.synopsys.com/glossary/what-is-ldap-injection.html

Singh. (2021, July 12). *Reflected XSS Vulnerability in Depth - GeeksforGeeks*. GeeksforGeeks. Retrieved November 22, 2022, from https://www.geeksforgeeks.org/reflected-xss-vulnerability-in-depth/

Team. (2021, December 21). *Second-Order SQL Injection Attack - Explained With Examples*. Offensive 360 - O360. Retrieved November 22, 2022, from https://offensive360.com/second-order-sql-injection-attack/

Waterman, A., & Smith, N. (2022, November 6). *SQL injection in Java.* RedLambda. https://www.redlambda.com/sql-injection-in-java/

What is a Second-Order SQL Injection and how can you exploit it successfully? (2019, July 24). Published InfoSec Write Up. Retrieved November 22, 2022, from https://infosecwriteups.com/the-wrath-of-second-order-sql-injection-c9338a51c6d