Coursework 1

Nicholas Cumplido 826488
CSCM28: Security Vulnerabilities and Penetration Testing

March 3rd 2020

Contents

1 Introduction to the vulnerability

Almost any digital entity that deals with data can be an injection vector, particularly in web applications. An attacker can exploit flaws in applications by entering malicious data to the application, mainly through parts of an application where a user can input data.
Although the vulnerabilities are easy to see in the source code, flaws in the code can go undetected due to the size of the application. Flaws in applications are mainly present in Legacy code.
Injection vulnerabilities are frequently found in SQL, NoSQL, OS command, object relational mapping, LDAP (Lightweight Directory Access Protocol), expression languages, XML parsers and SMTP headers.
Attackers use tools such as fuzzers and scanners to detect injection flaws.
Injection attacks can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access and even complete host takeover. The impacts that injection attacks have on businesses depends on the needs of the application and data.
An application is vulnerable to attack when: appropriate data validation, filtering or sanitation isn't taken on user entered data. When malicious data is used with object-relational mapping.
Source code review is the best method of detecting if applications are vulnerable to injections, closely followed by thorough automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs.
Basic counters to injection attacks involve, frequently monitoring the application and the data it stores, vulnerability scanning and good architecture of the application.

2 Technical overview of the vulnerability

The types of SQLI are:

- In-band SQLi (Classic SQLi) This is the most common and easiest type, the channel used to attack the application is the same channel that is used to receive the results.
- Error-based SQLi This method is a version of In-Band SQLi that uses error messages to determine the structure of a database. While error messages are useful for the development of an application, they should be secure enough to not allow attackers to understand the structure of a database.
- Union-based SQLi This is another version of In-Band SQLi. It makes use of the SQL operator UNION to combine two or more SELECT statements into a single result. This result is then as part of the HTTP response.
- Inferential SQLi (Blind SQLi) Unlike the other methods, this method takes time to execute but is just as dangerous as the attacks. The attacker is not able to see the result of the attack as no transferred from the application. Instead, the database structure can be recreated by the attacker by sending payloads. Then, from observing the application's response and behaviour, the attacker can reconstruct the database.
- Boolean-based (content-based) Blind SQLi This attack is a version of Inferential injection that involves sending an SQL query to the application's database, the response is forced to respond with a true or false result. Even though no data is returned from the database, the attacker will be able to infer the response. The HTTP response will indicate whether the result is TRUE or FALSE, if the HTTP response remained the same the result is TRUE, if it remained the same the result is FALSE.
- Time-based Blind SQLi As another version of inferential-based SQLi, the time taken for the database to respond is measured to indicate whether the response is TRUE or FALSE. The time taken for the HTTP data to return is used to determine whether the result is TRUE or FALSE. Although no data is returned from the database, this attack is slow.

3 Example code of how the vulnerability is exploited

Below is a block of simple SQL statements that can be used for a user to login:
uName = getRequestString("username");
uPass = getRequestString("userpassword");
The two lines above make the SQL statement below, this is the final query sent to the database. sql = 'SELECT * FROM Users WHERE Name ="' + uName + '" AND Pass ="' + uPass + '"'
If an attacker were to input the query: " or ""=" into the username and password field, a valid statement would be created:

SELECT * FROM Users WHERE Name =”” or ””=”” AND Pass =”” or ””=””

This query will return all rows from the ”Users” table, since OR ””=”” is always TRUE.

4 Defence against injection

Preventing injection requires keeping data separate from commands and queries, an example of this would be to insert character as separators for the statements. Safe APIs are a preferred way of defending against injection attacks. These APIs have been proven to be effective in defending against injection attacks. White-list server side input validation is another method of defending against attacks.
A significant problem in defending against attacks is the use of special character, one way to solve this is ”parameterising” queries. This depends on the language however.
In this block of code, SQL statements are stored in variable and then passed through functions that provide type checking and validation.
string sql = ”SELECT * FROM Customers WHERE CustomerId = @CustomerId”;
SqlCommand command = new SqlCommand(sql);
command.Parameters.Add(new SqlParameter(”@CustomerId”, System.Data.SqlDbType.Int));
command.Parameters[”@CustomerId”].Value = 1;

5 Exploration of the number of effected systems, or number of occurrences in the wild

TalkTalk gets record £400,000 fine for failing to prevent October 2015 attack

An SQL injection attack on TalkTalk in October 2015 led to a fine from ICO of £400,00. The attack could have been prevented with ”basic steps” according to ICO. The personal information of 156,959 customers were accessed by the attacker and 15,656 customers’ bank details were accessed also. Not only was TalkTalk unaware that the database installed was outdated, but version was also no longer supported by the provider.

Russian Hackers Amass Over a Billion Internet Passwords

In 2015 a Russian crime ring stole over 1.2 billion username and password combinations as well as emails, in the then-known largest collection of stolen credentials. The credentials and information stolen are used for over 420,00 websites. As the sites that were attacked were still vulnerable, they could not be named at the time. Victims included household names and small internet sites.

Hackers Breach 53 Universities and Dump Thousands of Personal Records Online

Harvard, Stanford, Cornell and Princeton, are among 53 universities that were attacked in 2012. The group of hackers known as Team Ghostshell took ownership of the attack on Twitter and posted information of around 36,000 people.

LulzSec hacks Sony Pictures, reveals 1m passwords unguarded

Sony Picture's website was hacked the group LulzSec in June 2011. It used a basic SQL attack to gain one million users' personal information, 3.5 million coupons and 75,00 digital music codes. The main factor that allowed Sony Pictures to be hacked was that there were very few defenses in place. Resources and time limited the group to how much data they could retrieve. LulzSec claim to not hack for personal gain, they claim that they did the public and Sony a favour.

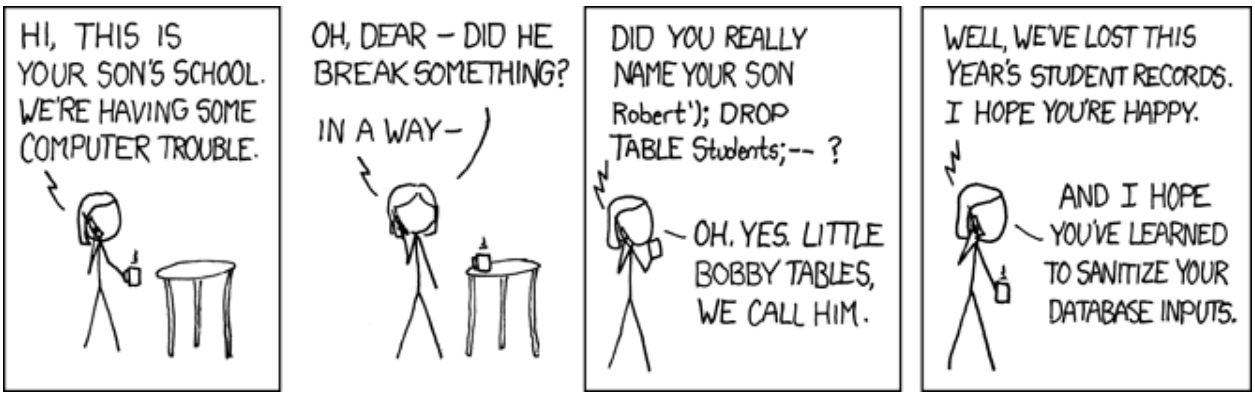

Figure 1: Exploits of a Mom" from xkcd.com