Chapter 15

SQL Injection

Concepts

SQL Injection

- Unsanitised input vulnerability
- gain unauthorised access

Leads to:

- Authentication, Authorisation Bypass
- RCE
- Info disclosure
- Compromise Integrity, availability.

Example:

Select Count (*) from USERS where Username 'Blah or 1=1 --' and Password='Blah'.

- Update Table

  'blah'; UPDATE .... --
- Add records (; INSERT INTO ... --)
- Identify tables (AND 1=(SELECT...)
- delete tables (; DROP Table --)
- return more data (OR 1=1)

Types of SQL Injection.

- Inband Injection
  - Error based (System / Incorrect)
  - Union
  - EOL Comment
  - Inline Comment
  - Piggybacked query
  - Tautology
- Blind Injection
  - Time delay
  - Boolean Exploit
  - Heavy Query
- Out of band Injection.

Inband Injection

- Same Communication channel

1.1 Error Based

- reads database error messages
- inject queries accordingly.

Stored Procedure

- executing malicious prepared SQL code without sanitization.

any username OR 1=1' --

illegal / incorrect Query

- intentionally sends incorrect query

1.2 Union based

- Union of intended dataset

```
Select * from Users where id=1
UNION ALL
Select * from CreditCardDetails
```

1.3 EOL comment

- line comments (--)
- ignored by the query

```
Select * from members where
username = 'admin'--' and
Password = 'abc'.
```

1.4 Inline Comment

- bypass blacklisting, remove space
- obfuscate, determine version db.

Insert into Users values(
'Attacker', 1, /*', 0, '*/ 'mypwd')

1.5 Piggybacked Query

- adds additional query to execute
- ; is used to concatenate

Select * from emp where id =
'bob'; drop table dept;

1.6 Tautology

- Uses conditional OR clause
- bypass authentication

```
Select * from users where
name = '' OR '1' = '1';
```

Blind Injection

- No Error Message
  - turns off generic error message
  - return custom error message
- Waitfor Delay
  - when no error displayed, use waitfor delay command

- WAITFOR DELAY '0:0:10' --
- use if/else condition
- if true: sleep for 10s
- if false: Custom msg will be returned

- Boolean Exploitation
  - Multiple statements with T or F are supplied (HTTP)
  - Compare results of find Success or fail

- Heavy Query
  - Use heavy Query to perform time delay without using time fun.

- takes long to execute in engine
- use multiple JOINS

Out of Band

- communicate server at acquire database server features
- different communication channel
- DNS, HTTP requests

xp_dirtree → DNS request (Microsoft SQL Server)

Methodology

1. Info Gathering, SQL Injection detection
2. Launch SQL Attack
3. Advanced SQL Injection

- Info Gathering

- Check if connects to database server
- List input fields, hidden fields (burp suite, Tamper chrome)
- generate error (ODBC)
- use UNION
- check error messages.

- Vulnerability Detection

- Using Testing Strings

Testing String	Testing String	Testing String	Testing String	Testing String
||6	or 1=1--	%22+or+isnull%281%2F0%29+%2F*	' or 1 in (select @@version)--	+or+isnull%281%2F0%29+%2F*
'||'6	" or "a"="a	' group by userid having 1=1--	' union all select @@version--	%27+OR+%277659%27%3D%277659
(||6)	Admin' OR '	'; EXECUTE IMMEDIATE 'SEL' || 'ECT US' || 'ER'	' OR 'unusual' = 'unusual'	%22+or+isnull%281%2F0%29+%2F*
' OR 1=1--	' having 1=1--	CRATE USER name IDENTIFIED BY 'pass123'	' OR 'something' = 'some'+'thing'	' and 1 in (select var from temp)--
OR 1=1	' OR 'text' = N'text'	' union select 1,load_file('/etc/passwd'),	' OR 'something' like 'some%'	' ; drop table temp --

		1,1,1;		
' OR '1'='1	' OR 2 > 1	'; exec master..xp_cmdshell 'ping 10.10.1.2'--	' OR 'whatever' in ('whatever')	exec sp_addlogin 'name' , 'password'
; OR '1'='1'	' OR 'text' > 't'	exec sp_addsrvrolemember 'name' , 'sysadmin'	' OR 2 BETWEEN 1 and 3	@var select @var as var into temp end --
%27+--+	' union select	GRANT CONNECT TO name; GRANT RESOURCE TO name;	' or username like char(37);	
" or 1=1--	Password:*/=1--	' union select * from users where login = char(114,111,111,116);	UNI/**/ON SEL/**/ECT	
' or 1=1 /*	' or 1/*	'/**/OR/**/1/**/=/**/1	'; EXEC ('SEL' + 'ECT US' + 'ER')	

Table 15.2: Standard SQL Injection inputs

- Function Testing (Black Box test)
- Fuzz Testing
- Static / Analysis testing

- Function Testing

- Detect issues (single / double quote)
- Input sanitisation (] → identifier)
- Truncation issue (buffer overruns)
- SQL Modification.

Source Code Review

Tools: Veracode, RIPS

Vulnerable code → SQL Injections

Static / Dynamic Code Analysis

- Launch SQL Injection Attacks

- Union SQL Injection.
  - extract database name, table
  - extract column names, data
- Error based Injection.
  - Stored Procedure.
- Bypass website logins.
- Perform Blind SQL Injection
  - extract first, second character
  - extract database user,
  - extract database name, length
  - extract column name

- extract data

- Perform Double blind SQL Injection.
- Perform blind SQL - Out of Band Injection.
- exploit Second Order SQL Injection
  - Saves first query
  - executes second with first Query
- Bypass Firewall using SQL Injection
  - Normalisation (change structure)
  - HPP (override delimiting characters)
  - HPF (HPP + Union)
  - Blind SQL Injection
  - Signature bypass (use firewall signature)
  - Buffer Overflow
  - CRLF (\r\n)

- Integration (group of methods)
- export value with regular expression
  - REGEXP (expression)

Advanced SQL Injection

- identify User level privilege
- discover DB structure (name, types)
- Column enumeration.
- DB administrators
  (sa, system, sys, dba, admin, root)
- Advanced Enumeration (tables, columns)
- features of different DBMS

	MySQL	MSSQL	MS Access	Oracle	DB2	PostgreSQL
String Concatenation	concat(,) concat_ws(delim,)	''+''	""&""	''||''	'' concat '' "+" '' || ''	''||''
Comments	-- and /**/ and #	-- and /*	No	-- and /*	--	-- and /*
Request Union	union	union and ;	union	union	union	union and ;
Sub-requests	v.4.1 >=	Yes	No	Yes	Yes	Yes
Stored Procedures	v.5.0 >=	Yes	No	Yes	No	Yes
Availability of information schema or its Analogs	v.5.0 >=	Yes	Yes	Yes	Yes	Yes

Table 15.4: Features of different DBMS

- Creating database accounts (SQL Server, Oracle, Access, MySQL)
- Password Grabbing
- Grabbing SQL Server hashes
  sys, Syslogins
- Transfer database to attacker machine
  'OPENROWSET'
- Interacting with OS shell

- Interacting with File System
  - load_file() - read, return contents
  - into outfile() - dump results
- Network Recon
  (assess connectivity, gather IP address)
  - xp_cmdshell
- finding a Bypassing Admin Panel
- PL/SQL exploitation.
- Creating Server Backdoors
- HTTP header based SQL Injections
  - xForwarded-For (identify IP)
  - User-Agent (info about agent)
  - Referer. (input without sanitisation)
- DNS exfiltration

Tools

- Sqlmap
- Mole
- Blisqy (time based blind)

Mobile Tools

- SQLi
- Droidbug SQLi Spyder
- Sqlmapchik

Evasion Techniques

evading IDS

- signature based detection systems
- compare attack strings with database during runtime.

Signature evasion

- inline comment
- char encoding
- string concatenation
- obfuscated code
- manipulate white space
- Hex Encoding
- URL encoding
- Null byte (%00)
- case variation
- IP fragmentation

Inline Comment

- evade signature with white spaces
- replaced by inline comments

  /* .... */

- also use within sql keywords

char Encoding

- char() function - without double quote

  char (114, 11, 11, 116)

String Concentenation

- split to avoid signature detection

  query || query2 || query3

obfuscated code

- encrypt, hash strings.

Manipulate white Space

- drops / adds white spaces between SQL keywords

Hex Encoding

- hexadecimal encoding
- Select = 0x736... 374

Sophisticated Matches

- OR 1=1' common Signature
- change with different names

URL encoding

- replaces character → ASCII code
- make use of double url encoding.

Null Byte

- Use %00 to bypass detection.

Case Variation

- Mix Uppercase, lowercase

Declare Variables

- defines variable to pass statements

IP Fragmentation

- split IP Packets to multiple fragments.
- reverse order, except (first, last) correct order.
- randomly.

Counter Measure

- disable xp_cmdshell
- run database with minimal rights
- Monitor db traffic using IDS
- use low privilege account for connection
- Supress error message - Custom Message
- Sanitize data as filter.
  (white list / Blacklist)
- test size, type of data.
- test string content
- reject binary data, escape sequence, comment characters
- Implement multiple validation.
- Avoid dynamic SQL construction

- Web Config → No sensitive info
- Use secure hash algorithm
- Apply least privilege rule
- handle exception.
- debug message should be removed.
- Use parameterised Query
- Use regular expression, stored procedure to detect code
- regular patch
- disable shell access
- don't show error info → users
- identify malicious SQL statements

Tools: OWASP ZAP, DSSS, Snort