**Module 15: SQL Injection**

**Concept**

* It is a flaw in web apps and not a database or web server issue
* Technique to take advantage of un-sanitized input vulnerabilitie to pass SQL cmds through a web app for execution by a backend database
* Gain unauthorized access, retrieve info

**Technologies**

* Server-side technology
* Exploit
* Susceptible databases
* Attack

**Types of SQL injection**

* **In-band sql injection:** The same communication channel to perform the attack and retrieve the results.
* **Blind/inferential sql injection:** Have no error messages from the system to work on.
* **Wait for delay, BENCHMARK()**
  + **Boolean Exploitation:** compare the response page to infer whether the injection is successful
  + **Hevay Query:** Use multiple joins on system table, retrieve a significant amount of data and taks a long time to execute. **Example:** SELECT \* FROM products WHERE id=1 AND 1 < SELECT count(\*) FROM all\_users A, all\_users B, all\_users C
* **Out-of-band sql injection:** Different communication channels to perform the attack and obtain the results
  + For example, in a Microsoft SQL Server, an attacker exploits the x**p\_dirtree command t**o send DNS requests to a server controlled by the attackerCopyright

**SQL injection methodology**

* Information gathering and vulnerability detection
  + identify data entry paths: analyze web GET and POST requests
  + extract info through error messages
* Launch attack
  + perform union sql injection, exreact database name, tables, column names, 1st field data
  + perform error based sql injection
  + bypass website logins using sql injection
  + perform double blind sql injection, based on time delays.
  + perform blind sql injection using out-of-band exploitation technique
  + exploit second-order sql injection
  + bypass firewall:
    - normalization method
    - HPP (HTTP parameter pollution) technique
    - HPF(HTTP parameter fragmentation) technique
    - blind sql injection
    - signature bypass
    - buffer overflow method
    - crlf technique
    - integration method
* advanced sql injection
  + database, table, column enumeration
  + create datavase accounts
  + password grabbing
  + grabbing sql server hashes
  + transfer database to attacker’s machine: An sql server can be linked back to an attacker’s DB via **OPENROWSET**. This can be accomplished by connecting to a remote machine on port **80**.
  + interact with os
  + interact with the file system, **LOAD\_FILE(), INFTO OUTFILE()**
  + network reconnaissance
  + PL/SQL exploitation
  + Create server backdoors
  + http header-based sql injection: X-Forwarded-For, User-Agent, Referer
  + DNS exfiltration

**Tools**

* sqlmap
* Mole
* blisqy

**Evasion Techniques**

* **In-line Comment:** Obscures input strings by inserting in-line comments between SQL keywords.
* **Char Encoding:** Uses a built-in CHAR function to represent a character.
* **String Concatenation:** Concatenates text to create an SQL keyword using DB-specific instructions.
* **Obfuscated Code:** Obfuscated code is an SQL statement that has been made difficult to understand.
* **Manipulating White Spaces:** Obscures input strings by inserting a white space between SQL keywords.
* **Hex Encoding:** Uses hexadecimal encoding to represent an SQL query string.
* **Sophisticated Matches:** Uses alternative expression of ”OR 1=1”.
* **URL Encoding:** Obscures an input string by adding the percent sign (%) before each code point.
* **Null Byte:** Uses the null byte (%00) character prior to a string to bypass the detection mechanism.
* **Case Variation:** Obfuscates SQL statement by mixing it with upper and lower case letters.
* **Declare Variables:** Uses variables to pass a series of specially crafted SQL statements and bypass the detection mechanism.
* **IP Fragmentation:** Uses packet fragments to obscure the attack payload, which goes undetected by the signature mechanism.
* **Variations:** Uses a WHERE statement that is always evaluated as “true”, so that any mathematical or string comparison can be used.

**Countermeasure**

* disabled shell access to the database
* IDS, IPS
* reject entries contain binary data, escape sequences, and common char
* Use type-safe sql parameters
* defenses in the application: input validation
* detect sql injection attacks, detect regular expressions used in sql injection
* Tools: OWASP ZAP, DSSS, Snort