SQL Injection

Počítačová obrana a útok

L. Burda, J. Novotný

VŠB-TU Ostrava

3. května 2017

Základní informace

- Zkratka SQLi
- Zařazena v OWASP Top 10
- Vložení SQL dotazu do vstupu v klientské aplikaci

Útok

Ukázka

- Zranitelný kód

```
statement = "SELECT * FROM users
WHERE name = '" + userName + "';"
```

- Uživatelský vstup

```
' OR '1'='1' --
```

- Výsledek

```
SELECT * FROM users
WHERE name = '' OR '1'='1' -- ';
```

Rozdělení SQLi

- Klasické
- Blind nebo Inference
- Database management system-specific
- Kombinované

SQLi v kombinaci s jinými zranitelnostmi

- SQLi + DDoS
- SQLi + DNS hijacking
- SQLi + XSS

Následky

- Důvěryhodnost (Confidentiality)
- Autentikace (Authentcation)
- Autorizace (Authorization)
- Integrita (Integrity)

Nástroje

- sqlmap
- BBQSQL

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        _
 ___ ___| |_____ ___ ___  {1.0.5.63#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting at 17:43:06

[17:43:06] [INFO] testing connection to the target URL
[17:43:06] [INFO] heuristics detected web page charset 'ascii'
[17:43:06] [INFO] testing if the target URL is stable
[17:43:07] [INFO] target URL is stable
[17:43:07] [INFO] testing if GET parameter 'id' is dynamic
[17:43:07] [INFO] confirming that GET parameter 'id' is dynamic
[17:43:07] [INFO] GET parameter 'id' is dynamic
[17:43:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

Obrana

Prevence

- Používání parametrizovaných dotazů
- Používání uložených procedur
- Seznam povolených vstupů
- Validace uživatelských vstupů

Parametrizované dotazy

- Java EE - PreparedStatement()
- .NET - SqlCommand(), OleDbCommand()
- PHP - PDO
- Hibernate - createQuery()
- SQLite - sqlite3_prepare()

Unsecured code - PHP

```php
session_start();
$pdo = new PDO("přihlašovací údaje");

$errors = array();
if ($_POST) {
    if (empty($_POST["name"])) {
        $errors[] = "Nebylo vyplněno jméno.";
    }
    if (empty($_POST["password"])) {
        $errors[] = "Nebylo vyplněno heslo.";
    }

    if (empty($errors)) {
        $name = $_POST["name"];
        $password = hash("SHA512", $_POST["password"] . 'sůůůl');
        // Dotaz níže obsahuje nebezpečnou SQL injekci
        $idQuery = $pdo->query("
                SELECT `id`
                FROM `user`
                WHERE `name` = '{$name}' AND `password` = '{$password}'
                LIMIT 1
        ");
        $id = $idQuery->fetchColumn();

        if ($id !== FALSE) {
                $_SESSION["userId"] = $id;
                header("location:account.php");
                exit;
        } else {
                $errors[] = "Bylo zadáno špatné jméno nebo heslo.";
        }
    }
}
```

Secured code - PHP

```php
$name = $_POST["name"];
$password = hash("SHA512", $_POST["password"]);

$prepared = $pdo->prepare("
        SELECT `id`
        FROM `user`
        WHERE `name` = :name AND `password` = :password
        LIMIT 1
");

$prepared->execute(array(
        ":name" => $name,
        ":password" => $password
));
```

Secured code - C#

```
using System.Data;
using System.Data.SqlClient;

using (SqlConnection connection = new SqlConnection(connectionString))
{
  DataSet userDataset = new DataSet();
  SqlDataAdapter myCommand = new SqlDataAdapter(
            "LoginStoredProcedure", connection);
  myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
  myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11);
  myCommand.SelectCommand.Parameters["@au_id"].Value = SSN.Text;

  myCommand.Fill(userDataset);
}
```

Děkuji za pozornost