Haladó adatbiztonság

Norbert Oláh

2022

Table of Contents

Beszúrás

3 - Beszúrásos támadások

Beszúrás

Az injekció lecsúszik a harmadik pozícióba. Az alkalmazások 94%-át tesztelték valamilyen injekciós formára, 19%-os maximális előfordulási arány mellett, átlagosan 3%-os előfordulási gyakorisággal és 274 ezer előfordulással. A figyelemre méltó gyakori gyengeségek felsorolásai (CWE-k) a következők: CWE-79: Cross-site Scripting, CWE-89: SQL Injection és CWE-73: A fájlnév vagy elérési út külső vezérlése.

Beszúrás alapelvek

- Az egyik legjobban kihasznált bemeneti validációs hiba
  - A rosszul implementált bemenetet (a támadási felületen keresztül) úgy használják, hogy a támadó megváltoztassa az bemenet kontextusának szerkezetét.
    - Például. egy bementet használ, hogy létrehozzon egy SQL lekérdezést, XML-t vagy JSON-t, LDAP-t, command line-t, stb.
  - Különösen veszélyes, ha ez a bemenet megváltoztathatja a program viselkedését.
    - Áttérés az adatokról kód-kontextusra: valami, amelyet tisztán adatnak feltételeznek, valójában ”egy kód részévé” válik.
- Ahhoz, hogy a támadás működjön, a környezetnek szintaktikailag helyesnek kell maradnia a beszúrásos támadás lefutása után.
  - Muszáj érvényes XML, JSON, SQL, ... eredményeznie

Gondolhat egy olyan beszúrásos sebezhetőségre, amelyik megfelel mindkét követelménynek:

- A program valamilyen adatot állít össze a felhasználói bemenetből.
- Ezeket az adatokat közvetlenül hajtják végre (kódszerűen) (parancs beszúrás, kód beszúrás, SQL beszúrás, HTML beszúrás), vagy azok szerkezete befolyásolja a program végrehajtásának módját (XML beszúrás, JSON beszúrás).

Formális definíció:
https://cwe.mitre.org/data/definitions/94.html
OWASP oldal:
https://www.owasp.org/index.php/Injection_Flaws

SQL beszúrás

- Nagyon gyakori probléma: SQL parancs összeállítása sztringeken keresztül külső (felhasználói) bemenet használatával.

```
String userName = ctx.getAuthenticatedUserName();
String itemName = request.getParameter("itemName");
String query = "SELECT * FROM items WHERE owner = '"
            + userName + "' AND itemname = '"
            + itemName + "'";
ResultSet rs = stmt.execute(query);
```

SQL beszúrás

- A fejlesztő feltételezése alapján - a lekérdezés így néz ki: **SELECT * FROM items WHERE owner = 'admin' AND itemname = 'pen'**
- Ha azonban az itemName értéke **name' OR 'a' = 'a,**, akkor az SQL parancs lekérdezi az egyes elemeket a táblázatból: **SELECT * FROM items WHERE owner ='whoever' and itemname = 'name' OR 'a'='a'**

Az előző egyszerű példában hozzáadunk egy ”mindig igaz” feltételt, amely megváltoztatja a lekérdezés logikáját, mely az összes elem megjelenítéséhez vezet.

Az SQLi-nek számos változata van (ideértve a noSQL és az ORM változatokat is), de a fő probléma mindig ugyanaz: a felhasználói bemenet összefűzése egy sztringbe, amelyet később SQL utasításként hajtanak végre. Annak ellenére, hogy ez a valószínűleg legismertebb beszúrásos altípus - az 1990-es évek vége óta létezik -, a programozók mégis folyamatosan áldozatává válnak. A következő dián a link megmutatja, hogy a PHP + MySQL-lel kapcsolatos StackOverflow kérdésekben hány százalék tartalmazza az SQL-injection kérdéseket.

Formális definíció:
https://cwe.mitre.org/data/definitions/89.html
OWASP oldal:
https://www.owasp.org/index.php/SQL_Injection
Wikipédia:
https://en.wikipedia.org/wiki/SQL_injection
Előfordulás a StackOverflow-on:
https://laurent22.github.io/so-injections/

Feladat - SQL beszúrás

SQL beszúrás

- Nyisd meg a www.insecar.com weboldalt a böngészőben
  - http://www.insecar.com
    (vagy használd az Insecar könyvjelzőt)
- Válaszd a Search-t
  - Kísérletezz szöveggel a mezőben
  - Ellenőrizd a lekérdezéseket az alábbi helyről
    http://attacker.com/queries.txt
    (C:/Ex/WebExample_jsp/Attacker_com)
- Lépj be az alábbival a username-nél admin' #

SQL beszúrás feladat

- Oszlopok száma:
- Adatbázisok neve:
- Táblák neve:
- Admin felhasználó / jelszó:

Tipikus SQL beszúrásos támadások módszerei

SELECT * FROM item WHERE item = ’<item>’ AND owner=’<userid>’

- SQL comment string, like #,–, or /*
  <item>=’ #
  - Kikommentálja a WHERE clause többi részét - a DBMS-től függ
- Kitalálni, hogy van-e sebezhetőség - pl. próbálkozni az alábbiakkal:
  <item>=’ AND 1=1 # and <item>=’ AND 1=2 #
  - Sebezhető, ha az első lekérdezés visszatér valamivel, a második pedig nem

Tipikus SQL beszúrásos támadások módszerei

SELECT * FROM item WHERE item = ’<item>’ AND owner=’<userid>’

- Használd az ”UNION”-t
  <item>=’ union select 1,2,... from users #
  - Vegye figyelembe, hogy az UNION csak akkor működik, ha a két lekérdezés ugyanannyi oszlopot ad vissza (valamint a típusok azonosak egyes DBMS esetén)
- A query-halmozás használatával több lekérdezés is támogatható.
  <item>=’; DELETE * from users #
  - ; parancs elválasztó - ennek a rendelkezésre állása a platformon és a DBMS-en múlik

A komment sztring felhasználható egy lekérdezés egy részének (általában valamilyen WHERE clause) eltávolítására, ami hasznos lehet a hitelesítő adatok ellenőrzésének vagy a hozzáférés-ellenőrzés kikerüléséhez.

Használhatsz ”AND 1=1 #” és ”AND 1=2 #” sztringeket egymás melletti elrendezésben, amely egy módszer az SQL-beszúrás biztonsági sebezhetőségének meglétére. Ha a biztonsági rés létezik, az első lekérdezés az eredeti eredményt adja vissza, míg a második lekérdezés nulla sort ad vissza.

Vak és idő-alapú SQL beszúrás

- Nincs információ a lekérdezésnél, nem léteznek hibaüzenetek; Használhatjuk még az SQL-beszúrást?
  - Továbbra is nyerhetünk információkat ”bit-by-bit”, ha valami megjelenik az oldalon a lekérdezéstől függően.
- Igaz/Hamis SQL lekérdezések
  - A lekérdezés arra kényszeríti, hogy eredményt adjon, vagy arra is akár, hogy egyáltalán ne adjon (vagyis a webhely viselkedéséből észlelhetjük az eredményeket), az adott körülményektől függően.
    ’ AND <something_returning_true_or_false> #
  - Ha a lekérdezés eredményénél valóban SEMMI nem jelenik meg az oldalon, akkor a lekérdezés a körülményektől függően csak várni kell egy kicsit, és a késések (delay) felfedhetik a sérülékenységet. (idő alapú SQL beszúrás)
    <condition_true_or_false> AND delay(3000) #
- Ezt a támadást nem kézzel végzik el, hanem automatizált eszközökkel. (SQLMap, Albatar,...)

A támadó azt használja ki, hogy a webhely eltérő eredményeket mutat, annak függvényében, hogy volt-e sor visszaadva. Egy egyszerű példa erre a normál oldal ¿0 sorral és a hibaoldal 0 sorral. Ez a támadónak ad információt (”a logikai utasítás igaz vagy hamis”?) a kérésekhez. Egy tipikus logikai kijelentés lehet: ”Az ”A” táblázatok első sorának első karaktere metatable”? Ha igen → továbblép a második karakterre. Ha nem → megpróbálja B-t helyette.

Az idő-alapú SQLi hasonló, de egy felhasználható késleltetést hozzáadunk a kéréshez és mérjük a válaszidőt. A logikai lekérdezések optimalizálása miatt ha a feltétel hamis, a késleltetés nem kerül végrehajtásra.

SQL beszúrás elleni védekezési módszerek

- Bizonyos bemenetek szűrése (black-list)
  - Probléma: DROP → DRO/**/P
  - Probléma: karakter kódolás(például Unicode)
  - Probléma: kiszűrheti az érvényes bemenetet is
- Tárolt eljárások
  - Óvatosan használni, ha dinamikusan készülnek
- Előkészített utasítások (Prepared statements)
  - A futtatandó lekérdezést már ismeri az adatbáziskezelő, annak csak a paraméterei változhatnak
  - Használj állandó sztringet a sztring összefűzése helyett, és töltsd ki a paramétereket egy API-n keresztül.
  - Néhány DBMS még szerver-oldali (filled) prepared statement-eket is támogat

SQL beszúrás elleni védekezési módszerek

```
String query="SELECT * from table WHERE id=" + var;
```

```
String query="SELECT * from table WHERE id=?";
PreparedStatement preparedStatement=conn.prepareStatement(query);
preparedStatement.setInt(1, Integer.parseInt(var));
```

Magyarázat

Az SQL beszúrás egy olyan típusú sebezhetőség, amelynek egyértelműen van ”ellenszere”. A (szerver-oldali) prepared statement-et előre összeállítja az adatbázis-szerver, és ehhez köti a paramétereket. Így még akkor is, ha a támadó SQL támadási sztringet ad, csak paraméter értékként dolgozza fel, és nincs hatással magára az SQL műveletre. Előfordulhat, hogy vannak olyan könyvtárak is, amelyek állítólag ”prepared statement” funkciót kínálnak, de a gyakorlatban az SQL lekérdezést az ügyfél oldalára helyezik (általában a paraméterek elkerülésével). Az ilyen ”ügyféloldali prepared statement”-nek sérülékenyek lehetnek az SQL-beszúrás ellen.

Prepared statement-ek a MySQL-ben: https://dev.mysql.com/doc/refman/5.7/en/sql-syntax-prepared-statements.html

Egyéb beszúrásos hibák

Parancs beszúrás

```
String btype = request.getParameter("backuptype");
String cmd = new String(
   "cmd /K C:\\utils\\backup.bat "+btype+" & C:\\utils\\cleanup.bat");
Runtime.getRuntime().exec(cmd);
```

- Egy shell parancs sztring a felhasználói bemenetből összeállítva más programokat indít el egy paraméterrel.
- Várható parancs, ha érték **backuptype** például **FULL**:
  cmd /K C: utils backup.bat FULL &C:utils cleanup.bat
- Azonban, ha **backuptype** egy ”**FULL& del * /q**”, a végrehajtott parancs a következő lesz:
  cmd /K C: utils backup.bat FULL del * /q &C:utils cleanup.bat

Parancs beszúrás

- A kérdésnek a fenti példa esetén történő javítása triviális.
  - A paraméterek validálása fehérlista alapján
  - Általában ez nem jellemző.

A parancs beszúrásnak sokféle formája lehet. Néhány tipikus példája *nix rendszerben: a program megpróbálja pingelni a felhasználó specifikus IP-ket (összefűzi a ”ping”-et a címmel), vagy megpróbál használni egy specifikus NTP szervert (összefűzni az ”ntp ”-t a szerver címmel).

Kiegészítés

A parancs beszúrás egyik módja a programozási nyelv és az OS által használt alacsony szintű funkciók áttekintése. Ez lehet egy sztring-paraméterrel rendelkező függvény (például a system()), amelyet ”létező” shell parancsként hajtanak végre, vagy egy olyan funkció, amelyben a parancs és a paraméterek el vannak különítve (például az execv()). Az utóbbi esetben, mivel a paramétereket a parancstól elkülönítve küldik el, az azokat módosító támadó nem lesz képes módosítani a parancsot. Ez hasonló azzal, hogy az előkészített utasítások megakadályozzák az SQL-beszúrást, ha jól használják.

Formális definíció:
https://cwe.mitre.org/data/definitions/78.html
Gyakorlati összefoglaló:
https://www.owasp.org/index.php/Command_Injection

Esettanulmány - ImageMagick

ImageMagick távoli kódfuttatás (RCE)

- Az ImageMagick egy szoftvercsomag bitmap képek létrehozásához, szerkesztéséhez, összeállításához vagy konvertálásához.
  - Számos képfeldolgozó plugin függ az ImageMagick könyvtártól.
    - például PHP's imagick és nodejs's imagemagick
- CVE-2016-3714 - A shell metakarakterek nem megfelelő szűrése a kód végrehajtásához vezet.
  - Delegált szolgáltatás
    - A fájlok külső könyvtárakkal is feldolgozhatók.
    - A külső könyvtárakat **system ()** hívással hívjuk meg.
  - Alapértelmezett delegált parancs a https kérésekhez.
    - "wget" -q -0 "%o" "https:%M"
    - %M az aktuális link a bemenettől

ImageMagick sérülékenység kihasználása

- Az %M paramétert nem szűrték helyesen, tehát a következő linkre.
  - https://example.com”;—ls ”-la
- A következő parancs végrehajtását okozza.
  - ”wget” -q -0 ”%o” ”https://example.com”; —ls ”-la”
- A legveszélyesebb esetei az svg és mvg fájlok
  - Példa az mvg fájlra:

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg";|ls "-la)'
pop graphic-context
```

ImageMagick sérülékenység kihasználása

- Az ImageMagick megpróbálja kitalálni a fájl típusát tartalom szerint.
  - A sérülékenység kihasználása nem függött a kiterjesztéstől
  - Átnevezve exploit.mvg-t exploit.png-re megkerüli a fájltípus-ellenőrzéseket

Több információ a sérülékenységről:
https://imagetragick.com/
Videó demonstráció
Videó demonstráció

Köszönöm a figyelmet!