MySQL and SQL Column Truncation Vulnerabilities

Míg a legjobban megvitatott biztonsági probléma az SQL-injection a web alkalmazások körében, addig léteznek más SQL-specifikus problémák is. Pl az SQL kérések túlméretezésével nem igazán foglalkoztak, de ettől függetlenül ez is vezethet biztonsági problémákhoz.

Ez sokszor buffer-overflow-hoz vezet, amiről a legtöbb web-fejlesztő nem is tud, vagy éppen nem vesz figyelembe.

Ezért elég sok olyan biztonsági probléma léphet fel az SQL lekérdezések túlméretezésének eredményeképpen, amikkel senki nem foglalkozik.

max_packet_size

MYSQL-ben létezik egy max_packet_size opció, aminek az értéke alap beállításoknál csupán csak 1Mbyte, ez szabályozza a kliens-szerver közötti adat folyam maximális méretét, vagyis ebből következik, hogy egy túlnyújtott SQL lekérdezés sosem éri el a szervert, vagyis sosem futhat le.

Biztonsági hibához vezethet, ha a támadó túlnyújtja a megadott értéknél a lekérdezés hosszát. Egy jó példa erre, egy olyan script ami a látogató adatait logolja. Kellően megnyújtott user-agent, sessionid stb-vel túlméretezhetjük a beirandó SQL parancsot, és így már nem fog a lekérdezés egy packetbe beleférni.

Egy másik valós példa, amikor egy web alkalmazás törölni kívánja a session id-ket, php-ben feldolgozza melyikeket akarja törölni és egyesével hivatkozik rájuk egy lekérdezésen belül. Ha túl sok a session id akkor, túlméretezett lesz a lekérdezés és egy idő után betelik az id-ket tartalmazó tábla, és nem lesz használható session.

Ezek után minden web fejlesztőnek figyelnie kellene arra, hogy sose legyen túlméretezett a lekérdezésük.

SQL Column Truncation Vulnerabilities

Ha a bemenetet, legalábbis a hosszát nem ellenőrizzük akkor SQL Column Truncation sebezhetőség következhet be. "SQL Column Truncation Vulnerability" -nek nevezem azt a biztonsági problémát, ami akkor következik, ha túlméretezett bemenetet adunk az SQL lekérdezésnek (INSERT), és beillesztéskor az adatbázis megcsonkítja azt. Alapból a MySQL a kapott inputot a mező hosszára vágja és csak egy figyelmeztetést (warning) dob. Legjobb ebben, hogy ezt a figyelmeztetést általában a web alkalmazások nem észlelik, vagy legalábbis nem kezelik le. MySQL-ben a STRICT_ALL_TABLES sql_mode aktiválásával elérhetjük, hogy ezek a figyelmeztetések hibákká (error) váljanak. A legtöbb web alkalmazás default módú szervereken fut,

de ha az alkalmazás még használná is a szigorúbb sql módokat akkor sem kellene hibát okoznia. Ettől függetlenül egy hosszúság ellenőrzés sosem árt.

Ahhoz hogy megértsük miért vezet ez biztonsági hibához, próbáljuk elképzelni a következőket:
- az alkalmazás egy fórum ahova bárki regisztrálhat
- az adminisztrátor felhasználóneve ismert: 'admin'
- MySQL fut, default módban
- nincs korlátozás a felhasználónév hosszában
- az adatbázisban a felhasználónév mező hossza 16 karakter.

Egy támadó beregisztrálhatná az 'admin ' nevet, viszont ez hibával térne vissza, mivel az a név már regisztrálva van.

SELECT * FROM user WHERE username='admin '

Ez azért történhet meg, mert a MySQL a stringeket nem binárisan hasonlítja össze default módban, hanem sokkal lazábban. Egyik lazasága a string végi szóközök elhagyása, így az 'admin ' -ból 'admin' is lehet, vagyis megegyeznek, és pont ezért okoz hibát a regisztrációnál.

Ha a támadó megpróbálja a 'admin x' felhasználónevet beregisztrálni, akkor az már sikerülni fog neki, mert 17 karakteres. A mező mérete csak 16, így 17 karakteres felhasználónevet lehetetlen megtalálni benne. Beillesztéskor 16 karakterre vágja a 17-et és beírja, így 'admin ' felhasználónevet kapunk eredményül.

Az eredmény: a táblánk két admin-t tartalmaz, különböző hosszal, viszont szóközökkel a végén, amik a lekérdezésekben nem számítanak. Innentől kezdve már csak az alkalmazáson van a hangsúly, hogy az hogy kezeli a különbségeket. Egy kis sebezhető pszeudó kód:

```
$userdata = null;
if (isPasswordCorrect($username, $password)) {
  $userdata = getUserDataByLogin($username);
  ...
}
```

A kód a következő SQL lekérdezést használja:

SELECT username FROM users WHERE username = ? AND passhash = ?

megállapítani a felhasználó jelszavát ezen módon egy biztonsági hibát okoz.

SELECT * FROM users WHERE username = ?

Mivel a támadó egy admin usert csinált, ezért tudja a jelszavát is, és mivel az igazi admin előrébb van a táblában, ezért a lekérdezésnél is az fog hamarabb megjelenni.

Konklúzió

Mind a két probléma amit leírtam új dolog, mind a kettő vezethet valós biztonsági hibához. Mivel eddig senki nem kereste efajta sebezhetőségeket és most publikáltam, valószínűleg a következő hetekben új biztonsági jelentések bukkannak fel a nyílt-forrású web alkalmazások terén.

Plusz megjegyzés (Bucsay Balázs):

Kipróbáltam mind a két módszert, tökéletesen működnek, viszont ezek is elég szélsőséges sebezhetőség fajták.

MySQL a max_packet_size értékét a max_allowed_packet-ből kapja, amit mind a my.cnf konfig fájlba, mind működés közben lehet állítani.
Ez alap beállításon 16Mb-ra van állítva.

Php-n belül és command line-ban is error-t dob a MySQL ha nem fér bele a packetbe a küldendő adat, viszont php alatt ezt a hibát nem kötelező lekezelni. Lekezelés nélkül, tökéletesen folytatódik a process, így alkalom nyílik a hibásan megírt alkalmazások kihasználása.

Eredeti szerző: Stefan Esser
http://www.suspekt.org/2008/08/18/mysql-and-sql-column-truncation-vulnerabilities/

Fordította: Bucsay Balázs - earthquake[at]rycon[dot]hu
http://www.rycon.hu/