Haladó adatbiztonság

Oláh Norbert

2022

Összefoglaló

1. OWASP Top Tíz 2021
   - Hozzáférés vezérlés megsértése
   - Kriptográfiai hibák

Hozzáférés vezérlés megsértése

1 - Hozzáférés vezérlés megsértése

Hozzáférés vezérlés tipikus gyengeségei

Az A01:2021-Broken Access Control az ötödik helyről a legsúlyosabb webalkalmazás-biztonsági kockázatot jelentő kategóriába került; a bevitt adatok szerint a tesztelt alkalmazások átlagosan 3,81%-a tartalmazott egy vagy több közös gyengeséget (Common Weakness Enumeration, CWE), és ebben a kockázati kategóriában több mint 318 ezer CWE előfordulás volt. A sérült hozzáférés-ellenőrzéshez rendelt 34 CWE-nek több előfordulása volt az alkalmazásokban, mint bármely más kategóriában.

Leirás

A hozzáférés-szabályozás úgy érvényesíti a házirendet, hogy a felhasználók ne tudjanak a nekik szánt jogosultságokon kívül cselekedni. A hibák jellemzően az összes adat jogosulatlan információközléséhez, módosításához vagy megsemmisítéséhez, illetve a felhasználó határain kívül eső üzleti funkció végrehajtásához vezetnek.

Gyakori kiváltók

A hozzáférés-szabályozás gyakori sérülékenységei a következők:

- Hozzáférés az API-hoz a POST, PUT és DELETE hozzáférési ellenőrzések hiányával.
- A privilégium emelése. Felhasználóként való fellépés bejelentkezés nélkül vagy rendszergazdaként való fellépés, amikor felhasználóként van bejelentkezve.
- Metaadat-manipuláció, például egy JSON Web Token (JWT) hozzáférés-vezérlő token újrajátszása vagy manipulálása, vagy a jogosultságok növelése érdekében manipulált cookie vagy rejtett mező, vagy a JWT érvénytelenítésével való visszaélés.
- A CORS hibás konfigurációja lehetővé teszi az API hozzáférést nem engedélyezett/nem megbízható forrásokból.

Gyakori kiváltók

- A legkisebb jogosultság elvének megsértése vagy alapértelmezett tagadás, amikor a hozzáférést csak bizonyos képességek, szerepkörök vagy felhasználók számára kellene biztosítani, de bárki számára elérhető.
- A hozzáférés-ellenőrzési ellenőrzések megkerülése az URL (paraméterek meghamisítása vagy kényszerített böngészés), az alkalmazás belső állapota vagy a HTML-oldal módosítása révén, vagy az API-kérelmeket módosító támadó eszközzel.
- Valaki más fiókjának megtekintésének vagy szerkesztésének engedélyezése annak egyedi azonosítójának megadásával (nem biztonságos közvetlen objektumhivatkozások).

Példa támadási forgatókönyvekre

1. forgatókönyv: Az alkalmazás ellenőrizetlen adatokat használ egy SQL-hívásban, amely hozzáfér a fiókadatokhoz:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

A támadó egyszerűen módosítja a böngésző ”acct” paraméterét, hogy a kívánt számlaszámot küldje el. Ha nem ellenőrzik megfelelően, a támadó bármelyik felhasználó fiókjához hozzáférhet.

```
https://example.com/app/accountInfo?acct=notmyacct
```

Példa támadási forgatókönyvekre

2. forgatókönyv: A támadó egyszerűen arra kényszeríti a böngészést, hogy megcélozza az URL-eket. Az adminisztrátori oldal eléréséhez rendszergazdai jogosultságok szükségesek.

```
https://example.com/app/getappInfo
https://example.com/app/admin_getappInfo
```

Ha egy nem hitelesített felhasználó hozzáfér bármelyik oldalhoz, az hiba. Ha egy nem adminisztrátor hozzáférhet az adminisztrációs oldalhoz szintén hiba.

Hozzáférés vezérlés tipikus gyengeségei

- Nem biztonságos közvetlen hivatkozás (Insecure direct object reference)
- Beégetett felhasználó nevek, jelszavak
- Az URL-hozzáférés korlátozásának nem megfelelő implementálása
- Sűrűn előforduló hiba: ügyféloldali biztonsági funkciók
  - Például. csak egy gomb elrejtése a felhasználói felületen
- A fájlok feltöltése szintén rendkívül kritikus funkció
  - Ha pl. egy veszélyes futtatható fájl feltölthető és hozzáférhető a Weben keresztül (ami szintén a téves konfiguráció/helytelen beállítás eredménye)

Ez kissé kapcsolódik a Secure Misconfiguration (helytelen konfiguráció) elemhez, mivel sok probléma származik a hibásan konfigurált webszerverekből vagy az alkalmazás szerverekből.

Insecure direct object reference (IDOR)

- Gyakori, hogy a fájlokat és az objektumokat közvetlenül egy paraméter értéken keresztül hivatkoznak meg
  - Például. a következő oldalon feltételezhetően a GYIK tartalma jelenik meg (faq.txt)
    http://www.a.com/document?page=faq.txt
- Ez kényelmes és karbantartható azonban van némi hátránya:
  - Fontos kérdés, hogy az oldal paramétere megfelelően van-e validálva.
  - Be lehet-e állítani egy értéket, amely azután más fájlra vagy adatforrásra mutat és amelyeknek egyébként nem szabadna elérhetőnek lennie?
- A támadó visszaélhet ezzel:
  - Hozzáférhet fájlokhoz, amelyekhez egy hétköznapi felhasználónak nem lenne szabad hozzáférnie
  - Vagy hozzáférhet akár rendszer fájlokhoz is

IDOR-nak tekintünk minden olyan esetet amikor a **rendszer nem ellenőrzi megfelelően**, hogy a felhasználó rendelkezik-e **megfelelő engedéllyel vagy jogosultsággal** az objektumon végrehajtandó művelet végrehajtásához (amelyet a felhasználó / támadó közvetlenül meghatározhat).

Formális definíció:
Link

Feladat -Insecure direct object reference

- Open the help page:
  http://www.insecar.com/help?file=help.txt
  - Now try
    http://www.insecar.com/help?file=whatever
  - Note the error message
- So why not refer to any file?
  http://www.insecar.com/help?file=../../../../../secret.txt
- Contains some secret information

Some (2-3) usernames from the server:

Védelem az IDOR ellen

- Ahelyett, hogy közvetlenül az objektumokra hivatkoznánk, vezessünk be egy absztrakciós réteget, amely ID-kat használ
  - Definiáljunk egy ”kódot” a kód által elvégzendő lehetséges műveletekhez
  - Majd képezzük le (map) ezeket az ID-kat a tényleges objektumokhoz, pl. fájlnevek
    1 → faq.txt
    2 → help.txt
    3 → about.txt
    ...
  - Ezzel egy időben, kikényszerítünk és végrehajtunk egy fehérlistás validálást

Védelem az IDOR ellen

- **Ökölszabály: mindig minimalizálni a felhasználói bemenet hatását a kódra!**
  - Csökkenteni a támadási felületeket

Ezt úgy tekinthetjük, hogy az indirection-t vagy az encapsulation-t felhasználjuk a rendszer erőforrásainak (ebben az esetben a fájlrendszer objektumai) belső állapotának védelmére.

Esettanulmány - Facebook Notes

Facebook Notes

- A Facebook Notes célja a nagyobb hozzászólások/posztok közzététele
- A note.php oldal felel a jegyzetek (notes) kezeléséért
  - Jegyzet szerkesztése esetén a jogosultságot nem ellenőrizték helyesen - bármilyen jegyzet eltávolítható az áldozat fiókjából, ha megváltoztatja a note_id-et egy másik jegyzet szerkesztésével.
  - Példa sebezhető kérésre:

```
POST /a/note.php?note_id=[victim's note id]&publish&gfid=
[attacker's token]
Host: touch.facebook.com

fb_dtsg=[attacker's token]&charset_test=&title=&body=&privacy=
&=Publish&_dyn=&__user=[attacker's userID]
```

A REST webszolgáltatás egy része a Facebook Notesban **nem ellenőrizte, hogy a megadott note ID valóban a felhasználó birtokában volt-e**; mindaddig, amíg a támadónak érvényes fiókja volt, bárkihez tartozó jegyzetet törölhetett.

A biztonsági rés leírása:
Link

Kriptográfiai hibák

2 - Kriptográfiai hibák

Kriptográfiai hibák

Egy pozícióval feljebb lépve a 2. helyre, a korábban Sensitive Data Exposure (érzékeny adatok kitettsége) néven ismert, ami inkább egy általános tünet, mint a kiváltó ok, a hangsúly a kriptográfiával (vagy annak hiányával) kapcsolatos hibákra helyeződik. Amelyek gyakran vezetnek érzékeny adatok kiszolgáltatottságához. Az alábbi közös gyengeségek (Common Weakness Enumerations, CWE) szerepelnek: CWE-259: Hard-coded Password, CWE-327: Broken or Risky Crypto Algorithm és CWE-331 Insufficient Entropy.

Érzékeny adatok szivárgása

- Számos különféle tervezési és megvalósítási hiba és gyengeség vezethet érzékeny adatok szivárgásához
  - Nincsenek kontrollok (encryption, access control, ...), vagy a kontroll nem megfelelően van használva (beleértve a téves konfigurációt is)
  - Gyenge kriptográfia, kriptográfiai biztonsági rések
  - Jelszó tárolása és kezelése, beégetett jelszavak, session és sütik védelme
  - Információ szivárgás hibaüzeneteken keresztül
  - Automatikus kiegészítés, gyorsítótárazás, memória kiürítés, titkos csatornák
  - Sok egyéb hiba és sebezhetőség ...
- Legjobb módszer: ne kövesse el ezeket a hibákat :-)
  - Ez inkább következmény, mint sebezhetőség

Ez egyfajta ”meta-kockázat”. A 2017. évi top 10-ben a #6-ról #3-ra emelkedett majd 2021.-ben 2. lett, mivel a kérdések gyakran előfordulnak, valamint az adatok kitettségének súlyossága mind technikai, mind jogi szempontból egyre súlyosabbá vált.

EU General Data Protection Regulation (GDPR):
Link

Hogyan előzhetjük meg

- Az alkalmazás által feldolgozott, tárolt vagy továbbított adatok osztályozása. Azonosítani, hogy mely adatok érzékenyek az adatvédelmi törvények, a szabályozási követelmények vagy az üzleti igények alapján.
- Ne tároljon érzékeny adatokat szükségtelenül. Dobja el a lehető leghamarabb, vagy használjon PCI DSS-kompatibilis tokenizálást vagy akár truncat. A meg nem őrzött adatok nem lophatók el.
- Ügyeljen arra, hogy minden érzékeny adatot titkosítson.
- Biztosítsa a naprakész és a szabványnak megfelelő algoritmusok, protokollok és kulcsok meglétét; alkalmazzon megfelelő kulcskezelést.

Hogyan előzhetjük meg

- Titkosítsa az összes átvitelben lévő adatot biztonságos protokollokkal, például TLS-sel továbbított titkosítással FS (forward secrecy), a szerver általi titkosítási prioritásokkal és biztonságos paraméterekkel.
- Az érzékeny adatokat tartalmazó válaszok gyorsítótárazásának letiltása.
- Alkalmazza a szükséges biztonsági ellenőrzéseket az adatminősítésnek megfelelően.
- Ne használjon olyan hagyományos protokollokat, mint az FTP és az SMTP, érzékeny adatok továbbítására.
- Tárolja a jelszavakat erős adaptív és sózott hash algoritmusokkal, amelyeknek van egy munkatényezője (késleltetési tényező), mint például az Argon2, a scrypt, a bcrypt vagy a PBKDF2.

Hogyan előzhetjük meg

- Az inicializálási vektorokat a működési módnak megfelelően kell kiválasztani. Sok üzemmód esetében ez egy CSPRNG (kriptográfiailag biztonságos pszeudo - véletlenszám - generátor) használatát jelenti. A nonce-t igénylő üzemmódok esetében az inicializálási vektorhoz (IV) nem szükséges CSPRNG. Minden esetben az IV-et soha nem szabad kétszer használni egy rögzített kulcshoz.
- Mindig hitelesített titkosítást használjon egyszerű titkosítás helyett.

Hogyan előzhetjük meg

- A kulcsokat kriptográfiai véletlenszerűen kell generálni, és bájtmező formájában kell tárolni a memóriában.
- Győződjön meg arról, hogy adott esetben kriptográfiai véletlenszerűséget használnak, és azt nem tervezték előre kiszámítható módon vagy alacsony entrópiával.
- Kerülje az elavult kriptográfiai funkciókat és kitöltési sémákat, például MD5, SHA1, PKCS number 1 v1.5 .

Transport layer security

- Feladat: a szerver és a böngésző között a továbbított adatok titkosságának biztosítása
  - De facto szabvány: transport-layer protection (SSL/**TLS**)
  - Számos egyéb lehetőség létezik más rétegekre is (IPSec, VPN, XML encryption, ...)
- Tipikus SSL/TLS problémák
  - **Az SSL már gyenge, helyette TLS**
  - Kerüld a gyenge kriptográfiát (pl. DES/3DES vagy MD5)
  - Kulcs mérete kisebb mint 2048 bit akkor nem számít biztonságosnak
  - Az önaláírt (vagy lejárt) tanúsítványok megkönnyítik az adathalász támadásokat
  - A biztonságos (HTTPS) és a nem biztonságos (HTTP) tartalom keverése ugyanazon a weboldalon szintén érzékeny adat szivárgásához vezethet

Emlékeztető,: ha nem állítjük be a ”Secure” flag-et a sütiken, akkor a támadó potenciálisan becsaphatja a felhasználót abban, hogy egy nem biztonságos csatornán küldi el a sütiket!
NIST ajánlás a kulcskezeléshez (NIST külön kiadvány 800-57),
1 rész: Általános
Link
Teszteld le az oldalad TLS beállításait:
Link

HTTPS érvényesítése

- Hogyan biztosíthatjuk, hogy az oldalak csak a HTTPS-en keresztül érhetők el?

https://example.org/a ✓ http://example.org/a ✘

- HTTP Strict Transport Security (HSTS)
  - Egy HTTP válasz header, amely megmondja a böngészőnek, hogy minden további kommunikációt a webhellyel csak a HTTPS-en keresztül végezzen el
  - Példa HTTP-válasz headerre (minden nagyobb böngészőben támogatott - néhány mobil böngésző, például az UC nem támogatja):

```
Strict-Transport-Security: max-age=60000; includeSubDomains
```

HTTPS érvényesítése

- Az redirect rule hozzáadása a szerveroldalra szintén lehetőség:
  - Példa Apache HTTPd config prescribing such redirecting:

```
<VirtualHost *:80>
        ServerAlias *
        RewriteEngine On
        RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
</VirtualHost>
```

A védelemhez a böngészőre kell támaszkodnunk, de a védelem további rétegekkel való ellátása mindig jó dolog.
HTTP Strict Transport Security:
Link
Dokumentáció a mod_rewrite-hoz:
Link

Köszönöm a figyelmet!