# Minta Alkalmazás Penteszt Riport

**Ügyfél:**
**Minta ügyfél**
2026-07-06
1.0

**Kontakt:**
lxdsec
+36.....
**info@lxdsec.hu**

info@lxdsec.hu
+36.....

# Tartalomjegyzék

info@lxdsec.hu
+36.....

# Vezetői Összefoglaló

A vizsgálat célja a(z) **[RENDSZER NEVE / ALKALMAZÁS NEVE]** informatikai rendszer biztonsági állapotának felmérése és a potenciális sérülékenységek azonosítása volt. A teszt során **[SEBEZHETŐSÉGEK SZÁMA]** darab, eltérő súlyosságú sebezhetőség került feltárásra, amelyek közül **[MAGAS/KRITIKUS DARABSZÁM]** magas kockázatot jelent a(z) **[SZERVEZET / ÜZLETI TERÜLET]** működése és az adatok biztonsága szempontjából.

A legkritikusabb megállapítások közé tartoznak olyan sérülékenységek, amelyek lehetőséget adhatnak **[PL. JOGOSULATLAN HOZZÁFÉRÉS / ADATSZIVÁRGÁS / TÁVOLI KÓDVÉGREHAJTÁS]**. Ezek mielőbbi javítása kiemelten javasolt, különös tekintettel a(z) **[LEGKRITIKUSABB RENDSZER / KOMPONENS]** érintettségére.

Összességében a(z) **[RENDSZER NEVE]** biztonsági szintje **[PL. MEGFELELŐ / KÖZEPES / FEJLESZTENDŐ]**, azonban a feltárt hiányosságok kezelése szükséges a kockázatok csökkentése érdekében. A riport részletes javaslatokat tartalmaz a sérülékenységek megszüntetésére és a biztonsági szint további növelésére.

info@lxdsec.hu
+36.....

# Módszertan és hatókör

A webalkalmazás sérülékenység vizsgálat a kiszolgáló infrastruktúra vizsgálaton túlmutató, kifejezetten a webhelyek, webalapú szolgáltatások és webalkalmazások tesztelésére tervezett módszertanokat és eszközöket felvonultató vizsgálat (OWASP Web Security Testing Guide v4.2). Emellett a támadási vektorok felállításánál az Offensive Security metodológiái mellett az aktuális trendeket figyelembe véve végezzük. A webalkalmazások sérülékenység vizsgálata egy szinten automatizált eszközökkel történő vizsgálat, az automatizált vizsgálatokat követően pedig további manuális vizsgálatok történnek a sérülékenységek teljeskörű felderítése érdekében.

- Információgyűjtés
- A cél rendszer működésének feltérképezése
- Támadási felület meghatározása
- Sérülékenység felderítés
- HTTP Host header injection
    - Password reset poisoning
    - Web cache poisoning
    - Host header authentication bypass
    - Routing-based SSRF
- SQL injection (SQLi)
- Cross-site scripting (XSS)
    - Stored XSS
    - Reflected XSS
    - DOM based XSS
- Cross-site request forgery (CSRF)
- Clickjacking
- Cross-origin resource sharing (CORS)
- XML external entity injection (XXE)
- Server-side request forgery (SSRF)
- http Request smuggling
- OS command injection
- Server-side template injection
- Directory traversal

**Felhasznált eszközök:**

- BurpSuite Professional
- Nikto
- SQLmap
- Dirb
- GoBuster

info@lxdsec.hu
+36.....

- Nmap
- Kali linux Penetration Testing OS

| System | Description |
|---|---|
| 10.0.0.1 | System1 |
| 10.0.0.2 | System2 |
| 10.0.0.3 | System3 |
| 10.0.0.4 | System3 |

info@lxdsec.hu
+36.....

# Feltárt sebezhetőségek

A sérülékenységvizsgálat során **1 Kritikus**, **1 Magas**, **1 Közepes** and **1 Alacsony** sebezhetőséget tártunk fel:

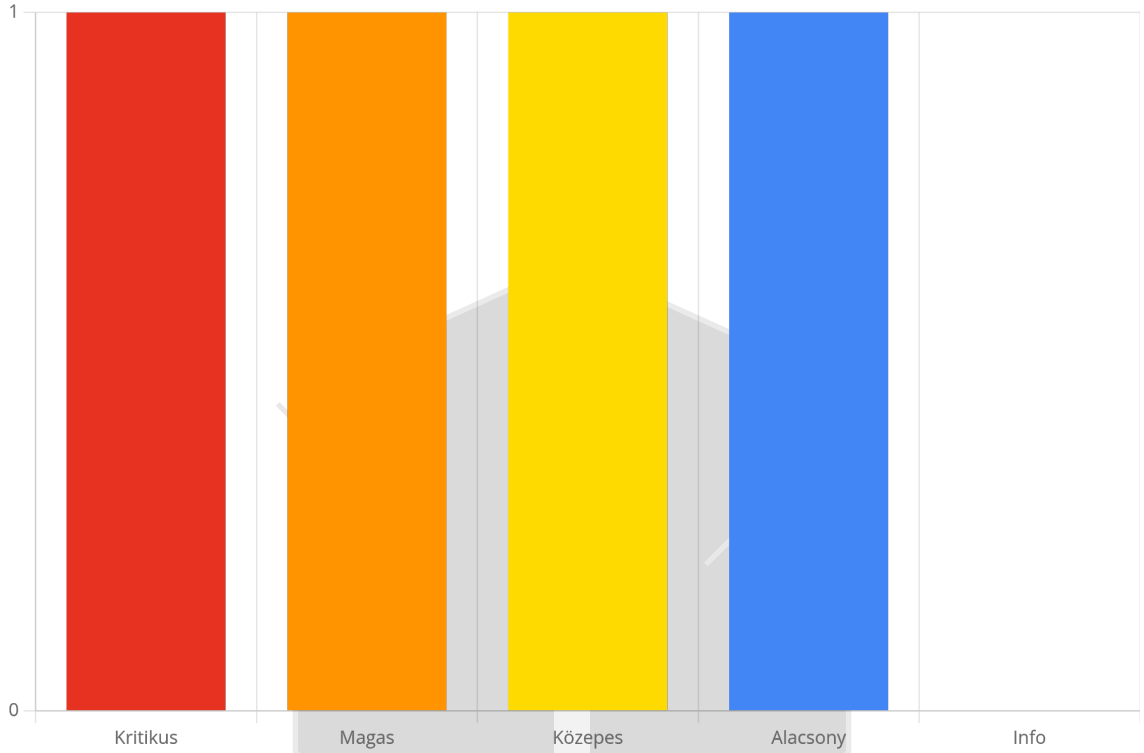


**Figure 1 - A feltárt sebezhetőségek eloszlása**

| Sebezhetőség | Besorolás |
|---|---|
| SQL injektálás (SQLi) | Critical |
| Tárolt XSS (Stored Cross-Site Scripting) | High |
| Érzékeny adatok URL paraméterekben történő közzététele | Medium |
| Session-kezelési hiányosságok | Low |

info@lxdsec.hu
+36.....

# 1. SQL injektálás (SQLi)

**Javítási állapot:**
**Besorolás:** Critical
**CVSS-Pontszám:** 9.8
**Érintett rendszer:** example.com
**Javaslat:** Az SQL injection sérülékenység megszüntetése érdekében az alkalmazás valamennyi adatbázis-művelete során előkészített SQL-utasítások (Prepared Statements), illetve ahol az alkalmazás architektúrája lehetővé teszi, tárolt eljárások (Stored Procedures) használata javasolt. Az ilyen megoldások biztosítják, hogy a felhasználói bemenet elkülönítve kerüljön kezelésre az SQL-parancsoktól, ezáltal megakadályozva azok jogosulatlan módosítását és az SQL injection támadások végrehajtását.

## Áttekintés

> **Az alkalmazás sebezhető volt SQL injection (SQL-befecskendezés) támadással szemben.** SQL injection támadás során a webalkalmazásnak megadott speciális bemeneti értékekkel a támadó befolyásolhatja az alkalmazás által az adatbázis felé küldött SQL-utasításokat. Az alkalmazott adatbázis-kezelő rendszertől (DBMS) és az alkalmazás felépítésétől függően ez lehetővé teheti az adatbázisban tárolt adatok kiolvasását és módosítását, adminisztratív műveletek végrehajtását (például az adatbázis-kezelő leállítását), illetve bizonyos esetekben akár tetszőleges kód végrehajtását is, amellyel a támadó teljes irányítást szerezhet a sérülékeny kiszolgáló felett.

> **A webalkalmazás nem biztonságos módon dolgozta fel a felhasználói bemenetet, ezért SQL injection sérülékenységet tartalmazott.** SQL injection támadás során a támadó speciálisan kialakított bemeneti értékek segítségével képes befolyásolni az alkalmazás által az adatbázisnak küldött SQL-lekérdezéseket. Az adatbázis-kezelő rendszer típusától és az alkalmazás kialakításától függően ez lehetőséget adhat az adatbázisban tárolt adatok megtekintésére és módosítására, adminisztratív műveletek végrehajtására (például az adatbázis-kezelő leállítására), valamint egyes esetekben akár távoli kódfuttatásra is, amely a sérülékeny szerver teljes kompromittálásához vezethet.

## Leírás

A vizsgálat során **SQL injection (SQL-befecskendezés)** sérülékenységet azonosítottunk a webalkalmazásban, amelynek kihasználásával sikerült hozzáférni az adatbázisban tárolt adatokhoz.

Az SQL injection a webalkalmazások egyik leggyakoribb szerveroldali sérülékenysége. Akkor fordul elő, amikor az alkalmazás dinamikusan épít fel SQL-lekérdezéseket,

info@lxdsec.hu
+36.....

amelyekbe a felhasználói bemenet megfelelő ellenőrzés vagy biztonságos kezelési mechanizmus nélkül kerül beillesztésre. A támadó speciálisan kialakított bemeneti adatok segítségével képes módosítani a lekérdezés eredeti működését, így a végrehajtott SQL-parancs eltérhet a fejlesztők által eredetileg tervezettől.

A sérülékenység oka, hogy az alkalmazás nem biztonságos módon állítja össze az SQL-lekérdezéseket, valamint nem biztosítja a felhasználói bemenet megfelelő validálását és elkülönítését az SQL-parancsoktól. Az SQL nyelv alapvetően nem különbözteti meg a vezérlőkaraktereket és az adatokat, ezért a bemenetben elhelyezett speciális karakterek – amennyiben nincsenek megfelelően kezelve vagy paraméterezve – módosíthatják a lekérdezés szerkezetét és működését.

Például az alábbi SQL-lekérdezés egy felhasználó adatait kérdezi le a megadott azonosító alapján:

```
SELECT * FROM Users WHERE UserId = <userId>;
```

Amennyiben a felhasználói bemenetet az alkalmazás közvetlenül fűzi hozzá a lekérdezéshez, egy támadó olyan értéket adhat meg, mint például:

```
' OR 1=1
```

Ennek eredményeként a lekérdezés logikája módosulhat:

```
SELECT * FROM Users WHERE UserId = '' OR 1=1;
```

Ebben az esetben a feltétel minden rekordra igaz lesz, így a lekérdezés nem egyetlen felhasználó adatait, hanem az adatbázisban található összes rekordot visszaadhatja. Ez lehetővé teszi a támadó számára, hogy a lekérdezés működését saját céljainak megfelelően befolyásolja.

Az SQL injection többféle formában és különböző technikákkal fordulhat elő, amelyek részben az alkalmazott adatbázis-kezelő rendszertől és az alkalmazás felépítésétől függenek. Közös jellemzőjük azonban, hogy a támadó a felhasználói bemenet felhasználásával képes módosítani a dinamikusan összeállított SQL-utasításokat.

A sikeres SQL injection támadások súlyos biztonsági következményekkel járhatnak. Lehetővé tehetik az adatbázisban tárolt érzékeny adatok jogosulatlan megtekintését, módosítását vagy törlését, veszélyeztethetik az adatok bizalmasságát és sértetlenségét, valamint lehetőséget biztosíthatnak a hitelesítési és jogosultságkezelési mechanizmusok megkerülésére. Egyes adatbázis-kezelő rendszerek és alkalmazáskörnyezetek esetén a sérülékenység akár távoli kódfuttatásra is kihasználható, amely a kiszolgáló teljes kompromittálásához vezethet.

## Javaslat

- Az alkalmazás teljes területén következetesen **előkészített SQL-utasítások (Prepared Statements)** használata javasolt. A paraméterezett lekérdezések

info@lxdsec.hu
+36.....

biztosítják, hogy a felhasználói bemenet ne módosíthassa az SQL-utasítás eredeti működését, ezáltal hatékony védelmet nyújtanak az SQL injection támadásokkal szemben.
- Ahol az alkalmazás architektúrája lehetővé teszi, **tárolt eljárások (Stored Procedures)** alkalmazása javasolt. A megfelelően implementált tárolt eljárások paraméterezett lekérdezéseket használnak, így szintén hatékonyan csökkentik az SQL injection sérülékenységek kockázatát.
- Minden felhasználói bemenetet **szigorúan validálni** kell. Az alkalmazás kizárólag az elvárt formátumú, típusú és tartalmú adatokat fogadja el. A védekezés alapját nem a potenciálisan rosszindulatú bemenetek szűrése vagy módosítása, hanem a megfelelő bemeneti validáció és a paraméterezett lekérdezések alkalmazása kell, hogy képezze.
- Az esetleges sikeres SQL injection támadások hatásának mérséklése érdekében az adatbázist elérő alkalmazásfelhasználó jogosultságait a **legkisebb jogosultság elvének (Principle of Least Privilege)** megfelelően kell kialakítani. Az adatbázis-felhasználó kizárólag a működéshez feltétlenül szükséges jogosultságokkal rendelkezzen.
- Az SQL injection sérülékenységek megelőzésével kapcsolatos részletes fejlesztői ajánlásokért javasolt az **OWASP SQL Injection Prevention Cheat Sheet** útmutató alkalmazása, amely a biztonságos adatbázis-kezelési gyakorlatokat és a védekezési módszereket részletesen ismerteti.

## További információ

- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

info@lxdsec.hu
+36.....

# 2. Tárolt XSS (Stored Cross-Site Scripting)

**Javítási állapot:**
**Besorolás:** High
**CVSS-Pontszám:** 7.2
**Érintett rendszer:** example.com
**Javaslat:** A felhasználói bemenetet minden esetben az elvárt és megengedett értékek alapján szükséges validálni és szűrni. Emellett biztosítani kell, hogy az adatok a HTTP-válaszokba történő beillesztés előtt a megfelelő kontextusnak megfelelően kódolásra (contextual encoding) kerüljenek. Ezáltal megelőzhető a kliensoldali kódfuttatást lehetővé tevő sérülékenységek, például a Cross-Site Scripting (XSS) kialakulása.

## Áttekintés

**A vizsgálat időpontjában a webalkalmazás a felhasználói bemenetet ellenőrzés nélkül tárolta, majd azt nem biztonságos módon illesztette be az HTTP-válaszokba. Ennek következtében az alkalmazás tárolt (Stored) Cross-Site Scripting (XSS) sérülékenységre volt fogékony.**

A Stored XSS sérülékenységek kihasználása nem igényel felhasználói interakciót, mivel a rosszindulatú kód a szerveren tárolásra kerül, és minden olyan felhasználó számára végrehajtódhat, aki az érintett tartalmat megtekinti. Ezáltal ezek a sérülékenységek általában súlyosabb kockázatot jelentenek, mint a Reflected XSS típusú sebezhetőségek.

## Leírás

A vizsgálat során a webalkalmazásban **tárolt (Stored) Cross-Site Scripting (XSS)** sérülékenységet azonosítottunk. A nem megfelelő bemeneti validáció és adatkódolás következtében sikerült rosszindulatú szkripteket a webalkalmazásba injektálni és azokat perzisztensen eltárolni.

A Cross-Site Scripting (XSS) egy gyakori webes sérülékenység, amely akkor fordul elő, amikor az alkalmazás nem megfelelően kezeli a felhasználói bemenetet, és így lehetővé teszi rosszindulatú szkriptek bejuttatását az alkalmazás által kiszolgált tartalomba. XSS támadások esetén a támadó JavaScript kódot ágyaz be a webalkalmazás által megjelenített tartalomba.

Tárolt XSS esetén a támadás célja, hogy a szkriptkód az alkalmazás adatbázisában vagy más perzisztens tárolójában rögzítésre kerüljön, majd később minden olyan felhasználó számára kiszolgálásra kerüljön, aki az érintett oldalt megnyitja. A sérülékeny oldal egyszerű meglátogatása is elegendő ahhoz, hogy a kód a felhasználó böngészőjében végrehajtásra kerüljön.

info@lxdsec.hu
+36.....

A támadás során a rosszindulatú szkriptek az alkalmazásba kerülnek, eltárolódnak, majd később az HTTP-válaszok részeként visszaadásra kerülnek. A böngészőben végrehajtott kód képes lehet sütik (cookies), munkamenet-tokenek (session tokenek) vagy egyéb érzékeny adatok megszerzésére.

Sikeres kihasználás esetén a támadó a sértett felhasználó jogosultságai alatt képes műveleteket végrehajtani az alkalmazásban, illetve hozzáférhet annak adataihoz. Amennyiben az érintett felhasználó kiemelt jogosultságokkal rendelkezik, a támadó akár a teljes webalkalmazás feletti kontrollt is megszerezheti.

## Javaslat

- Biztosítani szükséges, hogy minden feldolgozott adat a lehető legszigorúbb szűrésen és validáción menjen keresztül. A validálást az elvárt és megengedett bemeneti értékek alapján kell megvalósítani.
- A felhasználói adatok HTTP-válaszokba történő beillesztése előtt minden esetben megfelelő **kimeneti kódolást (output encoding)** kell alkalmazni. A kódolásnak kontextusfüggőnek kell lennie, azaz az adott HTML dokumentumbeli elhelyezés (pl. HTML body, attribútum, JavaScript kontextus) alapján a megfelelő escape-elési szabályokat kell használni.
- Azon HTTP válaszok esetén, amelyek nem tartalmaznak HTML vagy JavaScript tartalmat, javasolt a `Content-Type` fejléc megfelelő beállítása (pl. `text/plain`), valamint az `X-Content-Type-Options: nosniff` fejléc alkalmazása a tartalom típusának helytelen értelmezésének megakadályozására.
- További védelmi rétegként javasolt **Content Security Policy (CSP)** bevezetése, amely szabályozza, hogy mely kliensoldali szkriptek futhatnak le, és melyek kerülnek blokkolásra.
- Az XSS sérülékenységek megelőzésével kapcsolatos részletes iránymutatások az **OWASP Cross-Site Scripting Prevention Cheat Sheet** dokumentumban találhatók.

## További információ

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

info@lxdsec.hu
+36.....

# 3. Érzékeny adatok URL paraméterekben történő közzététele

**Javítási állapot:**
**Besorolás:** Medium
**CVSS-Pontszám:** 5.9
**Érintett rendszer:** example.com
**Javaslat:** Az érzékeny adatok védelme érdekében kerülni kell azok URL paraméterként történő továbbítását. Ehelyett az adatokat HTTP üzenet törzsében (body) javasolt továbbítani, például POST kérés használatával. Ez csökkenti annak kockázatát, hogy az adatok böngésző gyorsítótárban, webszerver naplófájlokban, Referer fejlécekben vagy egyéb köztes rendszerekben rögzítésre kerüljenek, ezáltal mérsékelve a jogosulatlan hozzáférés lehetőségét.

## Áttekintés

A vizsgálat során megállapításra került, hogy a webalkalmazás érzékeny adatokat HTTP kérésekben URL paraméterként továbbított. Az URL-ben átadott adatok a böngésző gyorsítótárában (cache) tárolásra kerülhetnek, továbbá különböző rendszerekben – például webszerver naplófájlokban, HTTP Referer fejlécekben vagy megosztott infrastruktúrákban – is rögzülhetnek.

Ennek következtében fennáll annak a kockázata, hogy harmadik felek jogosulatlanul hozzáférhetnek az így továbbított érzékeny információkhoz.

## Leírás

A vizsgálat során megállapításra került, hogy a webalkalmazás érzékeny adatokat a „motiondata” URL paraméterben továbbított. Az URL-ben szereplő érzékeny információk több ponton is kiszivároghatnak, illetve nem kívánt módon naplózásra kerülhetnek.

Az ilyen módon továbbított adatok potenciálisan megjelenhetnek az alábbi helyeken:

- Referer fejlécben
- webszerver naplófájlokban
- megosztott rendszerekben vagy köztes komponensekben
- böngésző előzményekben (browser history)
- böngésző gyorsítótárban (browser cache)
- vizuális megfigyelés útján (shoulder surfing)

Ezáltal fennáll az érzékeny adatok jogosulatlan hozzáférésének és kiszivárgásának kockázata.

info@lxdsec.hu
+36.....

## Javaslat

- Az érzékeny adatok védelme érdekében minden esetben a HTTP üzenet törzsében (body) szükséges azokat továbbítani, például POST kérés alkalmazásával. Ez csökkenti annak kockázatát, hogy az adatok URL paraméterekben kerüljenek továbbításra és különböző köztes rendszerekben (pl. logok, böngésző előzmények, Referer fejlécek) rögzítésre kerüljenek.
- A kommunikáció biztonsága érdekében a teljes adatátvitelt titkosított csatornán, **HTTPS protokoll használatával** szükséges biztosítani, amely megakadályozza az adatok illetéktelen lehallgatását és módosítását a hálózati forgalom során.

info@lxdsec.hu
+36.....

# 4. Session-kezelési hiányosságok

**Javítási állapot:**
**Besorolás:** Low
**CVSS-Pontszám:** 3.6
**Érintett rendszer:** example.com
**Javaslat:** A felhasználói munkamenetek esetében be kell vezetni inaktivitás alapú automatikus kijelentkeztetést (session timeout). A felhasználók meghatározott idejű inaktivitást követően automatikusan kijelentkeztetésre kerüljenek, így csökkentve a jogosulatlan hozzáférés kockázatát, különösen megosztott vagy felügyeletlen munkaállomások esetén.

## Áttekintés

A vizsgálat során hiányosságokat azonosítottunk a webalkalmazás munkamenet-kezelésében. A felhasználói sessionök időkorlátozás nélkül érvényesek maradtak, és nem igényeltek újrahitelesítést a munkamenet során.

Ennek következtében egy jogosult hozzáféréssel rendelkező személy, illetve egy olyan támadó, aki hozzáfér egy már aktív munkamenettel rendelkező eszközhöz, visszaélhet a helyzettel, amennyiben a korábbi felhasználó nem jelentkezett ki kifejezetten az alkalmazásból.

A megfelelő session lejárat (session timeout) és újrahitelesítési mechanizmus hiánya növeli a jogosulatlan hozzáférés kockázatát, különösen megosztott vagy felügyeletlen munkaállomások esetén.

## Leírás

A vizsgálat során megállapításra került, hogy a felhasználói munkamenetek időkorlátozás nélkül voltak érvényesek. Ennek következtében egy már aktív session jogosulatlan felhasználása lehetővé válhat, amennyiben az adott munkamenet nem kerül explicit módon lezárásra a felhasználó által.

A sérülékenység kihasználására például olyan esetben kerülhet sor, amikor egy harmadik fél hozzáfér egy olyan eszközhöz, amelyen a felhasználó munkamenete még aktív állapotban van. További kockázatot jelent, hogy a session tokenek újrafelhasználhatók lehetnek abban az esetben, ha azok valamilyen módon kiszivárognak (például naplófájlokon, helyi rendszereken vagy proxy szervereken keresztül).

Ez a hiányosság növeli a munkamenet-eltérítés (session hijacking) kockázatát, különösen megosztott vagy nem megfelelően felügyelt környezetekben.

info@lxdsec.hu
+36.....

## Javaslat

- A webalkalmazásban a felhasználói munkamenetek esetében be kell vezetni az **inaktivitás alapú automatikus lejáratot (session timeout)**, amely meghatározott idejű tétlenség után automatikusan megszakítja a munkamenetet.
- A session timeout időtartamát az alkalmazás biztonsági követelményei és az adott jogosultsági szint kritikalitása alapján szükséges meghatározni. Általános ajánlás szerint ez az érték körülbelül **egy óra és egy nap közötti tartományban** lehet, a rendszer kockázati besorolásától függően.

info@lxdsec.hu
+36.....

# A változások listája

| Verzió | Dátum | Leírás | Szerző |
|---|---|---|---|
| 1.0 | 2026-07-06 | Draft | lxdsec |

# Lenyomat

LXDSec
info@lxdsec.hu
+36.....