PROBLÉMA-ANALÍZIS ENTERPRISE RENDSZEREKBEN

5. forduló

adnovum

A kategória támogatója: Adnovum Hungary Kft.

RENDELKEZÉSRE ÁLLÓ IDŐ:

12:00

Ismertető a feladathoz

Felhasznált idő: 01:48/12:00 **Elért pontszám:** 0/34

1. feladat 0/10 pont

Milyen támadások kivédése ellen célszerű certificate pinninget alkalmaznunk?

Válaszok

- [ ] a felhasználó hibás adatokat tölt fel
- [x] egy rosszhiszemű internetszolgáltató aki a DNS-t kontrollálja
- [x] támadó aki egy root CA-t kontrollál
- [ ] támadó aki képes a felhasználó eszközére malware-t telepíteni
- [ ] terheléses támadás

Magyarázat

1. válasz: Nem. A certificate pinning a felhasználói input validálására nem nyújt megoldást.

2. válasz: Nem. Ehhez nem szükséges certificate pinning megteszi helyette a TLS, mert a támadó nem tud hiteles CA által aláírt tanúsítványt szerezni a kérdéses domainre.
3. válasz: Igen. Pont ez a lényege a certificate pinning-nek, hogy a kliens csak egy bizonyos tausítványt (vagy CA-t) tart megbízhatónak, tehát egy random root CA tanusítványát nem fogja elfogadni.
4. válasz: Nem. Az ezköz feletti teljes irányítás és root jogosultságok megszerzése a kliens oldali alkalmazás szintű védőmechanizmusok kiiktatására ad lehetőséget.

Frissítés (2021.11.24): versenyzői visszajelzés alapján a 2.válaszlehetőség is elfogadható: nem feltétlenül elég, hogy a támadó egy root CA-t kontrollál, még el kell érnie, hogy az ő szerverével kommunikáljon az alkalmazás. Ezek hiányában a valós/jó szerverrel fogunkk kommunikálni, így mindegy, hogy root CA-t kontrollál a támadó. A 2-es válasz pedig pont ezt a "problémát" oldja meg, hiszen így a támadó elérheti, hogy az ő szerverével kommunikáljunk.

2. feladat 0/12 pont

Alkalmazásunk felhasználók regisztrációját teszi lehetővé email segitségével. Amikor megadják emailüket az alkalmazás automatikusan legenerál nekik egy 32 véletlenszerű karakterből álló ideiglenes jelszót és közvetlen eltárolja az adatbázisban egy szöveges oszlopban majd elküldi nekik emailben, hogy beléphessenek vele első alkalommal és megváltoztathassák az általuk választott jelszóra. Milyen problémákat vet fel ez a viselkedés?

Válaszok

- [x] Jelszavak emailben való küldése nem biztonságos
- [x] Az ideiglenes jelszót hashelés nélkül tároljuk az adatbázisban tehát könnyen ellopható ha hozzáférnek az adatbázishoz
- [ ] Ezt az ideiglenes jelszót nem csak hashelni de saltolni is kell, mivel különben próbálgatással feltörhető ha ellopják az adatbázis tartalmát
- [x] Ideiglenes jelszó generálása fölösleges regisztrációkor

Magyarázat

1. válasz: az emailek plaintextben tárolódnak és általában örökké akár több köztes rendszeren is beleérve a felhasználó gépét
2. válasz: a jelszót elhashelve a támadók nem tudják majd azt fölhasználni azonnal amint hozzáférnek hogy belépjenek a rendszerbe mivel vissza kell fejteniük a hash milyen értékből származik
3. válasz: a saltolás ebben az esetben nem teszi biztonságosabbá a jelszót a feltételezéssel ellentétben mivel a jelszavunk már eleve 32 véletlenszerű karakterből áll tehát nem kell növelni az entrópiáját. Ezt a módszert akkor kell használni amikor a felhasználó saját jelszót ad meg ami túl kiszámitható lehet.
4. válasz: fölösleges komplikáció amely biztonsági kockázatot hordoz magában. Ezt a módszert jelszó visszaállitásnál szokás használni elsősorban.

3. feladat 0/12 pont

Az alábbi osztályok egy egyszerű számológépet valósítanak meg.

```
public class SimpleCalculator {
    private Map<String, Operation> operations;

    public SimpleCalculator() {
        this.operations = new HashMap<>();
        this.operations.put("+", new Addition());
        this.operations.put("-", new Subtraction());
    }

    public int perform(int a, int b, String operation) {
        return operations.get(operation).perform(a, b);
    }
}

public interface Operation {
    int perform(int a, int b);
    String render(int a, int b);
}

public class Addition implements Operation {
    public int perform(int a, int b) { return a + b; }
    public String render(int a, int b) { return a + "+" + b; }
}

public class Subtraction implements Operation {
    private Addition addition = new Addition();
    public int perform(int a, int b) { return addition.perform(a, (-1) * b); }
    public String render(int a, int b) { return a + "-" + b; }
}
```

Szeretnénk továbbfejleszteni, azonban előtte szeretnénk megérteni, milyen lehetséges tervezési problémák vannak a jelenlegi megvalósítással.

A SOLID objektum-orientált tervezési alapelvek közül melyeket sérti a fenti kód?

Válaszok

- [x] Single responsibility principle
- [x] Open/closed principle
- [ ] Liskov substitution principle
- [x] Interface segregation principle
- [x] Dependency inversion principle

Magyarázat

1. válasz: egy művelet több mindent is csinál: számol, és a megjelenítéséért is felelős

2. válasz: újabb művelet hozzáadásánál módosítanunk kell a SimpleCalculator osztályt.
3. válasz: a Liskov substitution principle-t nem sérti a fenti kód, az Addition és a Substraction is használható mint Operation.
4. válasz: az Operation interface több, nem szorosan összetartozó metódust is tartalmaz (a render nincs használva a fent vázolt használati esetben)
5. válasz: a függőségek jelenleg nem befecskendezettek, hanem az osztályokba vannak hard-codeolva.