HADMÉRNÖK

V. Évfolyam 3. szám - 2010. szeptember

Krasznay Csaba
csaba@krasznay.hu

E-KÖZIGAZGATÁSI RENDSZEREK ÉS ALKALMAZÁSOK SEBEZHETŐSÉGI VIZSGÁLATA

Absztrakt

A programozási hibák kiszűrésére körültekintő tesztelésre van szükség. Amíg azonban a funkcionális és terheléses teszteknek kifinomult módszertana van, a biztonsági hibák felderítésére csak az elmúlt években kezdtek eljárásokat kidolgozni. Jelen tanulmány célja, hogy bemutassa azokat a releváns szabványokat és ajánlásokat, melyek segítségével az e-közigazgatási rendszerek fejlesztői szisztematikus módon tudják a legfontosabb biztonsági hibákat webes alkalmazások esetén kiszűrni. A sebezhetőség-vizsgálati módszertanok széles skálájának ismertetése és a Common Criteria behatolás-tesztelési követelményei segítenek meghatározni az elvárt ellenőrzési mélységet. A tipikus hibák ismertetése után célunk ezekből kiindulva egy olyan tesztelési követelményrendszer összeállítása, mely tetszőleges internetes alkalmazás esetén személyre szabható, és segít a támadási felületek csökkentésében az elektronikus közigazgatási környezetben.

Careful testing is required to detect programming errors. However, while functional and stress tests have sophisticated methodology, procedures for detecting security errors were established in the last few years. The study aims to present the relevant standards and recommendations that help the developers of e-government systems to filter the most important security errors of web applications in a systematic manner. Introducing the wide range of vulnerability assessment methodologies and the Common Criteria intrusion testing requirements will help establishing the required depth of control. Our aim is to compile a security test standard based on common errors which is customizable for any internet application and helps reducing the attack surface of the e-government environment.

Kulcsszavak: *sebezhetőség-vizsgálat, Common Criteria, OWASP, elektronikus közigazgatás ~ vulnerability assessment, Common Criteria, OWASP, e-government*

BEVEZETÉS

Az elektronikus közigazgatás kialakulásával párhuzamosan fókuszba kerül az ezt megvalósító alkalmazások biztonsága is. Miután döntően olyan rendszerekről beszélünk, melyek közvetve vagy közvetlenül interneten keresztül is elérhetők, ráadásul az adatcserék miatt akár több, egymástól független szervezet megoldásai is szoros kapcsolatban állnak egymással, a külső informatikai támadások elleni védelem alapvető tervezési kérdés kell, hogy legyen.

Az e-kormányzati rendszerek stabil működésének egyik legfontosabb kihívását a köznyelvben hackerként elterjedt támadók jelentik. Módszereik jellemzően tisztán informatikaiak, melyek a hálózati réteget, az operációs rendszereket, az alapszoftvereket és magát az alkalmazást célozzák meg, melynek eredménye az adatszivárgástól kezdve a szolgáltatás ellehetetlenítéséig terjed. A magyar központi közigazgatás informatikai rendszerei kielégítőnek mondható hálózati védelemmel rendelkeznek, melyet a Nemzeti Hálózatbiztonsági Központ működése is elősegít, de az alkalmazási réteg felé haladva egyre több kockázattal szembesülhetünk. Különösen igaz ez a böngészőn keresztül elérhető portálokra, melyek sebezhetőségeit szinte észrevétlenül fel lehet deríteni, és ezeket kihasználva akár a védettnek gondolt adatbázisokig vagy belső hálózatokig is el lehet jutni. Egy nem kellő körültekintéssel megírt alkalmazás tehát akár olyan erőforrásokat is veszélybe sodorhat, melyekről az üzemeltető ezt nem is gondolná.

A tesztelési forgatókönyv végrehajtása nem garantálja minden távoli támadás sikeres kivédését, de nagymértékben csökkenti az automatizálható és kevesebb hozzáértést igénylő hibák kihasználásának kockázatát. Mivel magyar nyelven ilyen jellegű módszertan nem érhető el, jelen dokumentum célja hiánypótló módon segíteni a közigazgatási fejlesztők és megrendelők munkáját a behatolás-tesztelés területén.

1. BIZTONSÁG A FEJLESZTÉSI ÉLETCIKLUSBAN

Az információbiztonságra már az alkalmazás első tervezéséi lépéseinél gondolni kell. Ezért bármilyen fejlesztési életciklus modellt is használ a fejlesztő, a biztonsággal kapcsolatos lépéseket be kell építenie a modelljébe. Az egyes fejlesztési életciklus modellek eltérő elnevezéssel ugyan, de általában az alábbi lépéseket tartalmazzák: Projektindítás és tervezés, Funkcionális követelmények meghatározása, Rendszertervezés, Fejlesztés és dokumentálás, Elfogadás, Telepítés, Üzemeltetés és fenntartás, Áttekintés és kivonás.[1]

Projektindítás és tervezés

- A biztonsági igények felderítése:
  - Az alkalmazásban tárolt információk kritikusságának meghatározása
  - Alapvető biztonsági célok meghatározása
- Kezdeti kockázatelemzés
  - Fenyegetések/Sérülékenységek/Kockázatok
  - A védelmi intézkedések megvalósíthatóságának elemzése
  - A biztonsággal kapcsolatos költség/haszon elemzés elvégzése
- Biztonsági keretrendszer meghatározása
  - Lényeges biztonsági kérdések és kockázatok
  - A szolgáltatás szint megállapodás (SLA) meghatározása

Funkcionális követelmények meghatározása

- Biztonsági feladatok a projekttervben
  - Konfigurációkezelés és hozzáférés-védelem a projekt végrehajtása során

[1] Ehhez nagyon hasonló felépítésű a National Institute of Standards and Technology (NIST) SP 800-64 Rev. 2 alapvető szabványa is, de a felsorolás ennek egy pontosított változatát tartalmazza.

    - Nyomon követés
- Biztonsági követelmények meghatározása
    - A kockázatelemzés alapján védelmi intézkedések meghatározása
- Előzetes biztonsági tesztelési terv
    - Tesztelési eljárások és erőforrások
    - Értékelési követelményrendszer meghatározása
- Biztonsági követelmények beépítése a pályázatokba és szerződésekbe
    - Az SLA szerződések tartalmazzák a biztonságot
    - Hardver és szoftver mentések, letétek
- A funkcionális alapkövetelmények tartalmazzák a biztonságot

Rendszertervezés

- Biztonsági specifikációk meghatározása
    - Rendszer/alrendszer/interfész
    - Alkalmazás/adatbázis/hardver és firmware/hálózat
- A biztonsági tesztelési terv frissítése
    - Biztonsági tesztelési eljárások kidolgozása
    - Biztonsági tesztelés abnormális és illegális körülmények között
- A biztonsági terület beillesztése a formális dokumentációba és a minőségbiztosításba

Fejlesztés és dokumentálás

- A biztonsággal kapcsolatos kód megírása és beillesztése
    - Hozzáférés-védelem a kódhoz
    - A kód dokumentálása
- A biztonsággal kapcsolatos kódok tesztelése és értékelése
- Annak ellenőrzése, hogy a jóváhagyott biztonsági komponensek megvalósultak-e

Elfogadás

- Biztonsági komponensek tesztelése
- Biztonsági tesztelés az integrált környezetben
    - A funkcionális működés és teljesítmény felmérése
    - A tesztelési hibák azonosítása
    - A teszteredmények összevetése a biztonsági követelményekkel
- A biztonsági kód telepítése a szükséges módosításokkal
- A biztonsági intézkedések dokumentálása
    - A felhasználói útmutatóknak tartalmaznia kell a biztonságos működés feltételeit
- Elfogadási tesztelés
    - Az utolsó lehetőség a sérülékenységek azonosítására
- A projekt biztonságosságának elfogadása/megerősítése

Telepítés

- Biztonsági minősítés megszerzése
- Felhasználók oktatása
- A rendszer élesüzemű telepítése

Üzemeltetés és fenntartás

- Mentési és visszaállítási tesztelések
- Biztonsági eljárások megfelelőségének ellenőrzése
- Periodikus kockázatelemzés
- Újratanúsítás
- A környezetet érintő változások hatásainak elemzése
- SLA megállapodások ellenőrzése

Áttekintés és kivonás

- A változások hatásait folyamatosan monitorozni kell
- Ha a változások olyan hatásokat váltanak ki a rendszerből, hogy azt már nem lehet gazdaságosan/biztonságosan üzemeltetni, akkor ki kell vonni a működésből
- A kivonásra megfelelő stratégiát kell kidolgozni.

A fentiek szerint a biztonsági tesztelési követelményeket már a funkcionális követelmények meghatározásánál figyelembe kell venni, majd háromféle részletességgel kell végrehajtani: kód, modul és integrált környezet szinten. Ez a három megközelítés egymástól lényegesen eltérő módszertan használatát kívánja meg.

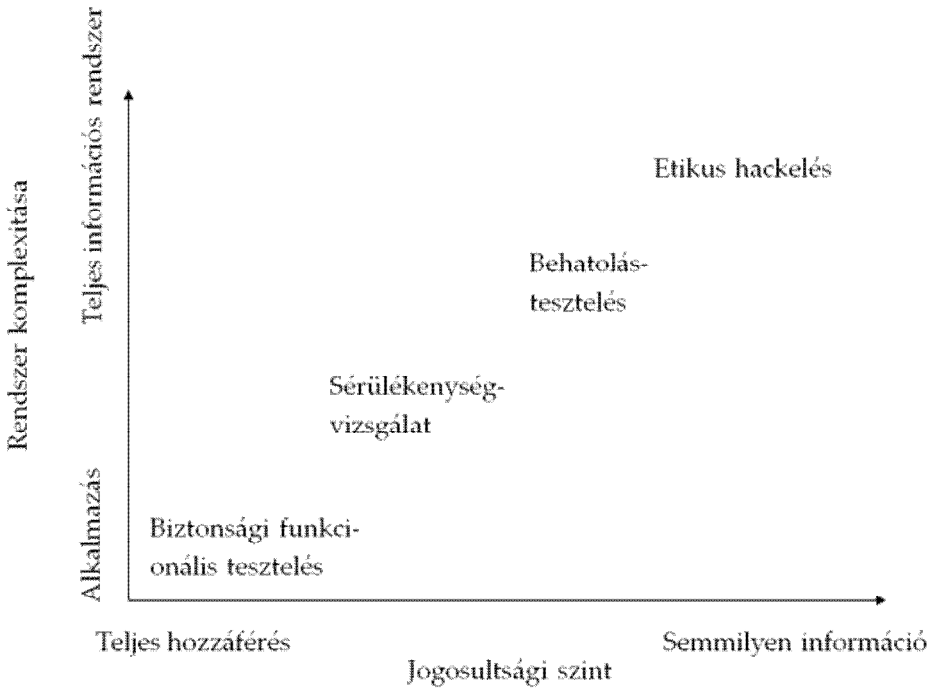
2. A BIZTONSÁGI TESZTELÉSI MÓDSZERTANOK BEMUTATÁSA

A biztonsági tesztelések módszertanával több, mérvadó ajánlás is foglalkozik. Ezek közül a legjelentősebbek az Open Source Security Testing Methodology Manual [1], mely az Institute for Security and Open Methodologies kiadványa, a NIST SP 800-115 Technical Guide to Information Security Testing and Assessment ajánlás [2], mely az amerikai kormányzat szabványa és az OWASP Testing Guide [3], amit az Open Web Application Security Project keretében fejlesztenek. A webes alkalmazások tesztelésének módszertanát ezek együttes használatával érdemes összeállítani.

A biztonsági tesztelési módszertanok csoportosítására jelenleg nincs egyezményes megállapodás, a tesztelt rendszerek ismeretétől kezdve, a hozzáférés mértékén át, a bevetett tesztelési eszközökig számos taxonómia létezik. Néhány fogalom azonban gyakran előfordul a szakirodalomban, melyek jól bemutatják a lehetséges módszereket [4].

- *Biztonsági funkcionális tesztelés*: az újonnan fejlesztett rendszer (alkalmazás) beépített védelmi kontrolljainak megfelelőségi ellenőrzése. Más néven white-box tesztelésnek is hívják, kód közeli megközelítésnek minősül. Pl. nézzük meg, hogy tényleg legalább 8 karaktert kell-e megadni jelszónak.
- *Sérülékenység-vizsgálat*: a rendszer (alkalmazás) védelmi kontrolljainál előforduló sebezhetőségek felderítése, tipikusan automatikus eszközökkel. Más néven black-box tesztelésként ismert, a modulszintű vizsgálatnak tekinthető. Pl. egyes bemeneteken olyan adatok megadása, melyekre a rendszer egyébként védett információkat ad át.
- *Behatolás-tesztelés:* az újonnan fejlesztett vagy már működő rendszer védelmi kontrolljainak kijátszása műszaki megoldásokkal, bármilyen kapcsolódó környezeti infrastruktúra felhasználásával. Két alfaja létezik, a Blue teaming, melynek során a tesztelő pontosan ismeri a tesztelt infrastruktúrát, és a Red teaming, melynél semmilyen ismerettel nem rendelkezik.
- *Etikus hackelés*: a működő rendszer védelmi kontrolljainak kijátszása bármilyen technikával, a rendszer előzetes ismerete nélkül. Ebbe beletartozik az emberi ráhatással (social engineering) történő támadás is.

A fenti technikákat a jogosultsági szint és a rendszer komplexitása szerint az 1. ábra mutatja be.



1. ábra: Biztonsági tesztelési módszertanok

A fenti módszerek közül a sérülékenység-vizsgálat és a behatolás-tesztelés a legelterjedtebb, használatát több forrásban is kötelező előírásként találhatjuk meg. A tipikusan amerikai szabványok, jogszabályok az informatikai biztonság egyik alapkövének tekintik az ilyen típusú teszteléseket. Az alábbi felsorolás összefoglalja a legelterjedtebb követelményeket.

- Payment Card Industry Data Security Standard (PCI DSS): a nagy bankkártya cégek kártyaadatok biztonságos elektronikus kezelésére vonatkozó előírása az ellenőrzés egyik legfontosabb alapkövének tekinti a sérülékenység-vizsgálatokat. A tanúsításhoz szükséges kétszintű vizsgálat mellett már az érintett rendszerek fejlesztését követően, a megrendelő számára is előírt a biztonsági tesztelés elvégzése. Az előírás 6.3.7 követelménye szerint a webes alkalmazásokat az OWASP útmutatói alapján kell fejleszteni és ezeket mintavételes módszerrel kell ellenőrizni (biztonsági funkcionális tesztelés), majd a 6.6 követelmény szerint manuális vagy automatikus módszerrel (sérülékenység-vizsgálat) legalább évente vagy minden változás után, egy erre specializálódott szervezettel (belső vagy külső) teszteltetni kell, és az esetleges hibák kijavítása után ezt a tesztet meg kell ismételni. [5]
- Federal Information Security Management Act (FISMA): Az USA 2002-ben elfogadott e-kormányzati törvénye alapján a központi ügynökségek valamennyi informatikai rendszerét biztonságos körülmények között kell üzemeltetni. [6] Az alapvető követelményeket a NIST SP 800-53 szabvány [7] tartalmazza, mely a magyar Közigazgatás Informatikai Bizottság 28. [8] ajánláshoz hasonlóan három besorolási szintet határoz meg a rendszerekre. Az előírások között RA-5 jelzéssel szerepel a sérülékenység-vizsgálat. Ez az automatikus eszközökkel végrehajtott tesztelést preferálja, melynek eredményeit egységes formájú jelentésben kell bemutatni. A vizsgálatot a szervezet által meghatározott időközönként vagy a rendszert érintő új sebezhetőség megjelenésekor kell végrehajtani. Közepes biztonsági szinten az automatikus eszköz használatát integrálni kell a változáskezelési eljárások közé, kiemelt szinten pedig a sebezhetőség-vizsgálati folyamatot kell továbbfejleszteni. A

külső értékelő által végzett, akár komplex vizsgálatok (behatolás-tesztelés) nem kötelező elem, de ajánlott teszteljárásként fel van tűntetve.

- Control Objectives for Information and related Technology (CObIT): Az elsősorban pénzintézeti szektorban használt IT irányítási ajánlás a DS5.5 Security Testing, Surveillance and Monitoring részben javasolja az informatikai rendszerek rendszeres biztonsági tesztelését, ám ennek módját nem jelzi. Az erre vonatkozó RACI táblázat viszont már nevén nevezi a sérülékenység-vizsgálatot, melynek végrehajtása a Megfelelőség, Audit, Kockázat és Biztonság szerepkör feladata. A biztonsággal kapcsolatos folyamat érettsége akkor tekinthető meghatározottnak, 3-as szintűnek, ha legalább ad hoc módon történnek ilyen vizsgálatok. [9]
- ISO/IEC 27002: A legismertebb információbiztonsági szabvány a 12.6. fejezetben foglalkozik a sérülékenység-vizsgálattal. Eszerint a szervezetnek bizonyos időközönként fel kell mérnie rendszerének technikai sebezhetőségeit, és megfelelő védelmi intézkedésekkel csökkenteni kell az ezekből eredő kockázatokat. A szabvány nem ad útmutatást a vizsgálat pontos végrehajtására, akár a gyártóktól érkező vagy hiteles forrásokban megjelent információk rendszeres olvasása is kielégíti a követelményeket. A lényeg, hogy a szervezet rendelkezzen ilyen típusú folyamattal. [10] A KIB 28. ajánlás ezt a szellemiséget követi az RS-2 Hibajavítás követelményben, de a folyamat szerepét meglehetősen eljelentéktelenítve.

A biztonsággal foglalkozó szabványok és ajánlások túlnyomó többsége tehát foglalkozik a biztonsági tesztelésekkel, de ezt eltérő szigorúsággal teszi. Jelen tanulmány célja, hogy ezek alapján egységes közigazgatási módszertant dolgozzon ki.

3. COMMON CRITERIA KÖVETELMÉNYEK

Egy webes alkalmazás biztonsági teszteléséhez a legjobb kiindulópontot a Common Criteria (CC) szabvány adja, melynek Common Evaluation Methodology (CEM) című kiadványának B melléklete részletesen leírja egy értékelő feladatait. [11] A CC terminológiájában a sebezhetőségi felmérés az a folyamat, melynek során a termék hibáinak vagy gyengeségeinek a létét és kihasználhatóságát elemzik. A tesztelést az értékelő végzi értékelői tesztelés módszerével. Alapszinten csak a nyilvános forrásból hozzáférhető sebezhetőségeket kell felderíteni (az ISO 27002 és a KIB 28. ajánlásoknak megfelelően), ám a rendszer biztonsági besorolása szerint ennél sokkal részletesebb vizsgálatokra is sor kerülhet. A sérülékenység-vizsgálat három lépésből áll: a lehetséges sebezhetőségek felderítése, a sebezhetőség kockázati besorolása és a sebezhetőség kihasználása annak megerősítésére, hogy az az adott környezetben valóban kihasználható.

A CC öt típusba sorolja a sebezhetőségeket:

- Kikerülés (bypassing): Ebbe a kategóriába tartozik minden olyan hiba, melynek során a rendszer beépített biztonsági eljárásait megkerülik, pl. jogosulatlan hozzáférés, kriptográfiai védelem feltörése.
- Meghamisítás (tampering): Olyan sebezhetőségek, melynek kihasználásával a termék működése módosítható, pl. a biztonsági funkciók leállítása vagy a fizikai módosítás.
- Direkt támadások (direct attacks): A permutációt vagy véletlenszerűséget felhasználó védelmi eljárások tesztelése tartozik ebbe a körbe, pl. jelszóhosszúságokon alapuló támadások.
- Megfigyelés (monitoring): Az információk átvitelének vagy kiszivárgásának figyelése és felhasználása, pl. információs csatornák lehallgatása, rejtett csatornák alkalmazása, elektromágneses sugárzás figyelése.

- Nem megfelelő használat (Misuse): Ez alatt a termék nem megfelelő dokumentációját, helytelen konfigurációját és szokásostól eltérő használatát értjük.

A fenti hibák felderítésére két útja lehet az értékelőnek. Egyrészt a vizsgálat során szembesülhet olyan sebezhetőségekkel, melyek kihasználása nem triviális, de minden valószínűség szerint lehetséges. Ezek többnyire olyan véletlen felfedezések, melyek a tesztelés „melléktermékei", nem direkt ezekre volt kíváncsi a tesztelő. Példa lehet erre egy olyan teszteredmény, mely a termékben puffer túlcsordulást okoz, ami akár a védelmi funkciók megkerülését is lehetővé teheti alaposabb elemzés után. A másik lehetséges elemzési mód a valamilyen elvek alapján felépített analízis. Ez lehet nem strukturált, amelynek során az értékelő saját tapasztalata alapján általános sebezhetőségeket keres, fókuszált, ami egyes sebezhetőnek tűnő területek alapos elemzését jelenti, vagy módszertan alapján történő, a korábban felsorolt ajánlásokra épülő.

Az értékelési eljárás azonban elsősorban a támadási potenciál szintjétől függ. A tesztelés előtt el kell dönteni, hogy a termék milyen környezetben fog működni, és ennek alapján négy kategóriába lehet sorolni a lehetséges támadói képességeket: alap, alap-erősített, mérsékelt, magas. A Közigazgatási Informatikai Bizottság 25. számú ajánlása szerint besorolásban a magyar közigazgatási rendszereket alacsony és fokozott kihatású esetben alap-erősített, kiemelt kihatású esetben mérsékelt szintű sebezhetőség-vizsgálatnak kell alávetni. [12] Ez azt jelenti, hogy az értékelő a tesztelést olyan támadási kapacitással végzi el, ami az előírt szinten meg van adva. Természetesen az eljárás során felderíthetnek olyan sebezhetőségeket is, melyek az adott támadási szinten nem kihasználhatók, ilyenkor a fejlesztőnek/megrendelőnek kell meghoznia azt a döntést, hogy ezt a kockázatot elfogadja vagy nem.

A támadási potenciál kiszámolásához a CC kiváló segédletet nyújt. 5 paraméter segítségével meghatározható, hogy a tesztelő milyen körülmények között végezze el a vizsgálatát. Ezek a következők:

- a sebezhetőség sikeres kihasználáshoz szükséges idő (eltelt idő), lehetséges értékei: egy napnál kevesebb, egy nap és egy hét között, egy hét és két hét között, két hét és egy hónap között, egy hónap és hat hónap között, hat hónapnál több.
- a kihasználáshoz szükséges szakértelem (szakértelem szintje), lehetséges értékei: laikus (általános ismeretei vannak), profi (jól ismeri az adott terméktípust), szakértő (széleskörű ismeretei vannak a támadási technikákról), több szakértő (részterületek szakértői).
- a vizsgált termék felépítésének és működésének ismerete (termékismeret), lehetséges értékei: nyilvános információk (internetről elérhető), korlátozott információk (a fejlesztői közösségen belül ismert), érzékeny információk (csak egy speciális fejlesztői csapat által ismert), kritikus információk (csak néhány személy által ismert).
- a sikeres támadáshoz szükséges hozzáférés ideje és a próbálkozások száma (próbálkozási ablak), értékei: korlátlan (a támadás észrevétlen marad), egyszerű (egy napnál rövidebb ideig tartó hozzáférés, 10-nél kevesebb próbálkozás), mérsékelt (egy hónapnál rövidebb ideig tartó hozzáférés, 100-nál kevesebb próbálkozás), nehéz (legalább egy hónapig tartó hozzáférés vagy legalább 100 próbálkozás), nincs (a rendelkezésre álló idő vagy próbálkozásszám nem elégséges egy sikeres támadás véghezviteléhez).
- a támadás kivitelezéséhez szükséges hardverek és szoftverek (eszköztár), lehetséges értékei: szabványos eszközök (internetről szabadon letölthető), speciális eszközök (piacon beszerezhető), egyedi eszközök (adott célra fejlesztett), több egyedi eszköz (résztámadások egyedi eszközei).

A CC minden lehetséges értékhez egy számot rendel, melyek összege adja a támadási potenciált. A forgatókönyvek összeállításánál külső támadókkal számolunk, hiszen az e-

közigazgatási rendszerek üzemeltetői azzal számolnak, hogy legitim felhasználóik megbízhatóak. Természetesen a gyakorlat azt mutatja, hogy az ilyen támadásokat is szimulálni kell, ez azonban csak ajánlott, célunk az, hogy alapvető technikák felhasználásával jelentősen emeljük az ilyen rendszerek biztonsági szintjét.

Az alap-erősített támadási potenciál eléréshez a tesztelési forgatókönyv összeállításánál egy olyan támadóval kell számolni, aki bár ismeri a támadási technikákat, azokat csak nyilvánosan elérhető eszközök felhasználásával tudja végrehajtani. Célja a könnyen azonosítható hibák felderítése, melyet segítenek a rendszerről kiszivárgó információk. A szakzsargonban script-kiddie[2] névvel ellátott képességeket értjük ide. Ehhez a paraméterek: eltelt idő egy hónap, a szakértelem szintje profi, a termékismeret korlátozott információ, a próbálkozási ablak egyszerű, az eszköztár pedig szabványos.

A mérsékelt vizsgálat szimulált támadója olyan magas tudással rendelkező szakértő, aki képes hamar észrevenni az elemi hibákat, és ezekhez akár saját eszközöket is tud fejleszteni. A rendszer azonban megfelelően védett, arról semmilyen információ nem szivárgott ki. A szakszóval cracker-nek[3] vagy black hat hackernek nevezett személyek képességeit mutatja be. Ennek a támadásnak a paraméterei: eltelt idő egy hét, a szakértelem szintje szakértő, termékismeret nyilvános, próbálkozási ablak egyszerű, eszköztár egyedi eszközök.

Az OWASP Application Security Verification Standard Project (ASVS) a Common Criteriához hasonló elvek alapján különböző szintű ellenőrzésekre tesz javaslatot az elkészült alkalmazásban, így a hasznos kiegészítője lehet a szabványnak.

4. TIPIKUS TÁMADÁSI FELÜLETEK

Az Open Web Applications Security Project (OWASP) keretében 2003 óta rendszeresen közzéteszik a webes alkalmazásokra vonatkozó nagy kockázatú sebezhetőségeket. Az OWASP Top 10 2010-es kiadása [13] alapján a behatolás-tesztelés olyan módszertana állítható össze, mely segít a legtipikusabb hibák felderítésében, így jelentősen növekedhet az alkalmazás biztonsági szintje. A legnagyobb kockázatú hibák felsorolása az alábbiakban található.

A1 – Beszúrásos támadások (Injection): A beszúrásos hibák, melyek közé elsősorban az SQL, operációs rendszer és LDAP injection értendő, olyankor történnek, amikor a támadó nem megbízható adatokat küld a parancsfeldolgozó felé parancsként vagy lekérdezésként. A támadó kód eléri a parancsfeldolgozónál, hogy az nem kívánt parancsot hajtson végre, vagy érzékeny adatokat szivárogtasson ki.

A2 – Cross-Site Scripting (XSS): Az XSS hibák jellemzője, hogy egy alkalmazás nem megbízható adatokat vesz át és küld tovább a böngészőn keresztül megfelelő ellenőrzés és szűrés nélkül. Az XSS lehetővé teszi a támadónak, hogy szkripteket hajtson végre az áldozat böngészőjében, mellyel el tudja téríteni a felhasználó session-jét, weboldalakat tud megváltoztatni vagy át tudja irányítani a felhasználót egy kártékony oldalra.

A3 – Hibás hitelesítés és sessionkezelés (Broken Authentication and Session Management): Az alkalmazások hitelesítéshez és sessionkezeléshez kapcsolódó funkciói sok esetben nem megfelelően lettek implementálva, ezért lehetővé válik a jelszavak, kulcsok, session tokenek megszerzése, vagy más hibák előidézése, melynek segítségével a támadó más, jogosult felhasználó nevében tud eljárni.

A4 – Nem biztonságos direkt objektumhivatkozás (Insecure Direct Object References): Direkt objektumhivatkozásról akkor beszélünk, amikor a fejlesztő felfed egy hivatkozást valamilyen belső, implementációhoz szükséges objektum felé, mint pl. egy fájl, könyvtár

[2] http://en.wikipedia.org/wiki/Script_kiddie
[3] http://en.wikipedia.org/wiki/Hacker_%28computer_security%29#Black_hat

vagy adatbázis tábla. Megfelelő hozzáférés-védelem vagy más biztonsági megoldás nélkül a támadó vissza tud élni ezekkel a hivatkozásokkal, és nem jogosult hozzáférést szerezhet az ezekben tárolt adatokhoz.

A5 –Cross-Site Request Forgery (CSRF): A CSRF támadás során a támadó kényszeríti az authentikált felhasználó böngészőjét arra, hogy egy hamisított HTTP kérést küldjön egy sebezhető webalkalmazás felé, mely tartalmazza az áldozat session cookie-ját és más hitelesítési információkat. Ez lehetővé teszi, hogy az áldozat böngészője olyan kéréseket küldjön a sebezhető alkalmazás felé a támadó nevében, melyről azt hiszi, hogy az legitim forrásból érkezik.

A6 – Helytelen biztonsági beállítások (Security Misconfiguration): A megfelelő biztonság eléréséhez meg kell határozni az alapvető beállításokat, és ezeket meg is kell valósítani az alkalmazásokban, keretrendszerekben, alkalmazásszervereken, webszervereken, adatbázis-szervereken és minden más érintett platformon. Mivel a legtöbb rendszer nem olyan alapbeállítással kerül telepítésre, mely az elvárható biztonsági szintet valósítja meg, ezeket a konfigurációkat meg kell határozni, implementálni kell, és folyamatosan fenn kell tartani. A folyamat során a szoftverek frissítéseire is figyelemmel kell lenni.

A7 – Nem megfelelő kriptográfiai tárolás (Insecure Cryptographic Storage): Számos webalkalmazás nem megfelelően kezeli az érzékeny adatokat, mint pl. a hitelesítési adatok, mert ezeket nem titkosított vagy lenyomatolt formában őrzi. A támadók ezért megszerezhetik vagy módosíthatják a gyengén őrzött adatokat, ami számos visszaéléshez vezethet.

A8 – URL hozzáférés korlátozásának hibája (Failure to Restrict URL Access): A webes alkalmazásokban fontos az URL-ek hozzáférési jogainak ellenőrzése, mielőtt a felhasználó elérne egy védett hivatkozást vagy akciógombot. Ezt a hozzáférési jog ellenőrzést viszont minden esetben meg kell tenni, amikor ezeket a védett oldalakat elérik, mert különben a támadó elérheti a rejtett oldalakat.

A9 – Nem megfelelő szállítási réteg védelem (Insufficient Transport Layer Protection): Az alkalmazások gyakran nem megfelelő hitelesítést, titkosítást és bizalmasság-sértetlenség védelmet használnak az érzékeny hálózati forgalomban. Amikor viszont használnak, akkor is gyenge algoritmusokkal, lejárt vagy érvénytelen tanúsítványokkal teszik ezt, vagy egyszerűen nem megfelelően használják a kriptográfia adta lehetőségeket.

A10 – Nem ellenőrzött átirányítások és továbbítások (Unvalidated Redirects and Forwards): A webes alkalmazások gyakran irányítják át vagy továbbítják a felhasználókat más oldalakra, és használnak nem hiteles adatokat a forrásoldal megállapítására. Megfelelő ellenőrzés nélkül a támadók átirányíthatják a felhasználókat adathalász vagy kártékony oldalakra, vagy a továbbításokkal nem jogosult hozzáférést szerezhetnek.

A Common Criteria szerint besorolás a következő:

- Kikerülés: A2, A4, A5, A8
- Meghamisítás: A1, A10
- Direkt támadások: A3, A7
- Megfigyelés: A9
- Nem megfelelő használat: A6

5. VIZSGÁLATI MÓDSZERTAN

A webes alkalmazások teszteléséhez az Open Web Application Security Project keretében megalkotott módszertanok tekinthetők a leghasznosabbaknak, legalábbis ezekre hivatkozik a legtöbbet a szakirodalom napjainkban. Az OWASP kétfelé bontotta az ellenőrzési folyamatokat. Az OWASP Code Review Guide (CRG) [14] célja a létrehozott kódok manuális ellenőrzése, white-box módon. Gyakorlatilag a biztonsági funkcionális tesztelésnek

felel meg. Az OWASP Testing Guide (TG) a már elkészült alkalmazás black-box teszteléséhez, azaz sérülékenység-vizsgálatához ad segítséget.

A CRG tehát a kódok áttekintésére készült, mely folyamat célja meggyőződni arról, hogy a forráskódban a megfelelő biztonsági kontrollok implementálásra kerültek és úgy működnek, ahogy a tervezés szerint működniük kell. Ellenőrzi továbbá azt is, hogy a biztonsági kódokért felelős fejlesztők követték a számukra megállapított eljárásrendeket. Mivel a behatolás-tesztelés, de még a sérülékenység-vizsgálat sem képes teljes körűen felderíteni az esetleges hibákat, egyedül a white-box tesztelésnél van lehetőség a rosszul megírt kódrészleteket átfogóan megtalálni. A biztonsági funkcionális tesztelés a humán erőforrás és a célszoftverek alkalmazását is szükségessé teszi. Az automatizált eszközök itt a leghatékonyabbak, de nem képesek teljes mértékben pótolni a tesztelési szakembert, hiszen az egyes kontextusok döntően befolyásolhatják a kódolás megfelelőségét vagy nem megfelelőségét.

A CRG a legfontosabb technikai kontrolloknak az authentikációt, az authorizációt. a sessionkezelést, az input validációt, a hibakezelést, az alkalmazás telepítését és a kriptográfiát tartja, többé-kevésbé összhangban a Common Criteria 2. kötetében leírt biztonsági funkcionális követelményekkel. A legkomolyabb programozáskor elkövetett hibaforrásoknak pedig a puffer túlcsordulást, az operációs rendszer és SQL injektálást, az adatvalidációt, a cross-site scriptinget, a cross-site request forgery-t, a naplózási hiányosságokat, a session sértetlenségét és a versenyhelyzeteket tartja, szinkronban az OWASP Top 10 megállapításaival. A CRG részletesen bemutatja azokat a tipikus hibákat, amiket a biztonsági funkciók implementálásánál el szoktak követni, illetve azt is, hogyan lehet a programozási hibákat felderíteni.

A biztonsági funkcionális tesztelés azonban idő- és erőforrás-igényes, tehát drága. Emellett a szükséges szakértői háttér is nehezen elérhető Magyarországon. Célravezetőbb a TG-ben leírt sérülékenység-vizsgálatokat végrehajtani, ami nem jelent ugyan akkora bizonyságot, mint a CRG folyamatai, de így is jelentősen tudja csökkenteni a támadási felületeket. A TG az OWASP Top 10 fenyegetéseit veszi sorra, és írja le, hogy milyen tesztelési eljárásokkal lehet meggyőződni arról, hogy a kész alkalmazás ellenáll ezeknek. Black-box tesztelés esetén az automatikus teszteszközök már körülményesebben használhatók, de még bevethetők. Ezért az egyes tesztesetekhez konfigurálható céleszközöket szoktak használni, mely feltételezi, hogy a tesztelő jól ismeri ezt a területet.

A TG 9 kategóriában 66 tesztesetet határoz meg. Az alkalmazás típusától és megvalósításától függően kell ezeket végrehajtani.

Információszerzés:

- Webes keresők robotjai
- Felderítés webes keresők adatbázisában
- Alkalmazás belépési pontjainak azonosítása
- A webes alkalmazás ujjlenyomatának tesztelése
- Alkalmazás felderítés
- Hibaüzenetek elemzése

Konfigurációmenedzsment tesztelés

- SSL/TLS tesztelés
- DB Listener tesztelés
- Infrastruktúra konfigurációmenedzsmentjének tesztelése
- Alkalmazás konfigurációmenedzsmentjének tesztelése
- Fájlkiterjesztés kezelésének tesztelése
- Régi, mentett és nem hivatkozott fájlok
- Infrastruktúra és alkalmazás adminisztrátori interfészek
- HTTP metódusok és XST tesztelés

Authentikációs eljárások tesztelése

- Authentikációs adatok átvitele biztonságos csatornán keresztül
- Felhasználói adatbázis tesztelése
- Kitalálható (szótár alapú) felhasználói fiókok felderítése
- Nyers erejű tesztelés
- Az authentikációs séma kikerülésének tesztelése
- Sebezhető emléketető jelszó és jelszó visszaállítási lehetőségek
- Kilépés és böngésző cache menedzsment
- CAPTCHA kódok
- Többfaktorú authentikáció tesztelése
- Versenyhelyzetek kezelése

Sessionkezelés

- A sessionkezelési séma tesztelése
- Cookie attribútumok
- Session fixálás
- Kiterjesztett sessionváltozók tesztelése
- CSRF

Authorizáció tesztelése

- Elérési útvonalak tesztelése
- Az authorizációs séma kikerülése
- Jogosultság kiterjesztése

Üzleti logika

- Az üzleti logika tesztelése

Adatvalidáció:

- Nem perzisztens XSS
- Perzisztens XSS
- DOM alapú XSS
- Cross Site Flashing
- SQL Injection
- LDAP Injection
- ORM Injection
- XML Injection
- SSI Injection
- XPath Injection
- IMAP/SMTP Injection
- Kódbeszúrás
- Operációs rendszer parancsok kiadása
- Puffer túlcsordulás
- Összetett adatvalidációs tesztelés
- HTTP splitting/smuggling tesztelés

Túlterheléses támadások (DoS) tesztelése

- SQL wildcard támadás
- Ügyfélfiókok zárolása
- DoS puffer túlcsordulások tesztelése
- Felhasználó által meghatározott objektumallokáció
- Felhasználói adatbevitel, mint körbeforgó számláló
- Felhasználó által bevitt adatok lemezre írása
- Erőforrás sikertelen elengedése

- Túl sok adat tárolása egy session-ben

Web service tesztelés

- Web service információszerzés
- WSDL tesztelés
- XML struktúra tesztelés
- XML tartalom szintű tesztelése
- HTTP GET paraméterek/REST tesztelés
- SOAP csatolások
- Visszajátszásos támadás

AJAX tesztelés

- AJAX sebezhetőségek
- AJAX tesztelés

ÖSSZEFOGLALÁS

Az 1. fejezetben ismertetésre került a biztonságos alkalmazásfejlesztés folyamata, melynek alapvető része a biztonsági tesztelés. A 2. fejezetben bemutattuk a szóba jöhető módszertanok és azok a megfelelőségi követelmények, melyek kötelező módon előírják a sebezhetőség-vizsgálatok lefolytatását. Ezekből egyértelműen következik, hogy a modern e-közigazgatási rendszerek fejlesztésénél elkerülhetetlen az ilyen vizsgálatok lefolytatása. A 3. fejezetben meghatározásra került, hogy a magyar elektronikus közigazgatási rendszerekben milyen támadói profilt kell szimulálni a követelmények alapján. A 4. fejezetben a legfontosabb webes biztonsági fenyegetések leírásával egyértelművé tettük, milyen hibák kiküszöbölése elvárható minimálisan a fejlesztőktől, és mely területekre kell a tesztelőknek fókuszálnia. Végül az 5. fejezetben megneveztük azt a két módszertant, melynek segítségével a biztonsági funkcionális tesztelés és a sérülékenység-vizsgálat a leghatékonyabban megtervezhető.

Az érintett rendszerek megrendelőinek ezek alapján egyértelmű segédlet áll rendelkezésre a biztonsági vizsgálatok követelményeinek meghatározásához. A pontos keretek természetesen csak a rendszer ismeretében határozhatók meg, de ha a biztonság fontos kritérium, – márpedig a tapasztalatok szerint az – már a tender kiírása során figyelmet kell fordítani az ilyen részletekre is.

IRODALOMJEGYZÉK

1. Herzog, Pete. *Open-Source Security Testing Methodology Manual 2.2.* s.l. : Institute for Security and Open Methodologies, 2006.
2. Scarfone, Karen, et al. *Technical Guide to Information Security Testing and Assessment.* Gaithersburg : National Institute of Standards and Technology, 2008. NIST Special Publication 800-115.
3. Meucci, Matteo (szerk.). *OWASP Testing Guide V3.0.* s.l. : Open Web Application Security Project, 2008.
4. Krasznay, Csaba. A rendszer próbája: az etikus hackelés és penetrációs tesztelés. *krasznay.hu.* [Online] június 18, 2008. [Cited: április 22, 2010.] http://www.krasznay.hu/presentation/ethical_hacking_krasznay.ppt.
5. PCI Security Standards Council LLC. *Payment Card Industry (PCI) Data Security Standard Requirements and Security Assessment Procedures Version 1.2.1.* s.l. : PCI Security Standards Council LLC, 2009.

6. National Institute of Standards and Technology. Federal Information Security Management Act (FISMA) Implementation Project. *NIST.gov - Computer Security Division - Computer Security Resource Center.* [Online] National Institute of Standards and Technology, április 13, 2010. [Cited: április 22, 2010.] http://csrc.nist.gov/groups/SMA/fisma/index.html.

7. —. *Recommended Security Controls for Federal Information Systems and Organizations.* Gaithersburg : National Institute of Standards and Technology, 2009. NIST Special Publication 800-53.

8. e-Közigazgatási Keretrendszer Kialakítása projekt. *IT biztonsági műszaki követelmények.* Budapest : Miniszterelnöki Hivatal Elektronikuskormányzat-Központ, 2008. Közigazgatási Informatikai Bizottság 28. számú ajánlása.

9. IT Governance Institute. *Control Objectives for Information and related Technology (COBIT®) v4.1.* Rolling Meadows : IT Governance Institute, 2007. ISBN 1-933284-72-2.

10. International Organization for Standardization. *Code of practice for information security management.* Geneva : International Organization for Standardization, 2005. ISO/IEC 27002:2005.

11. Common Criteria Recognition Arrangement. *Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 3.* s.l. : Common Criteria Recognition Arrangement, 2009. ISO/IEC 18045.

12. Krasznay, Csaba. Szoftverfejlesztői követelmények minősített környezetben. *Hadmérnök.* 2009, Vol. IV, 4.

13. Williams, Jeff and Wichers, Dave. *OWASP Top 10 - The Ten Most Critical Web Application Security Risks.* s.l. : Open Web Application Security Project, 2010.

14. Keary, Eoin (szerk.). *OWASP Code Review Guide.* s.l. : Open Web Applications Security Project, 2008.