Coding and IT Security (VIHIBB01)
Laboratory Exercise

Penetration testing - naplófájlok elemzése

Author:
Dr. Bencsáth Boldizsár, OSCP

2021. november 19.

Tartalomjegyzék

1. Célok

A laborfoglalkozás a célja, hogy az alapjait megismerjék a hallgató a sérülékenységkezelés - ethical hacking tevékenységnek, amelynek az alapja valós támadók működésének modellezése. Azt szeretnénk megmutatni, mi a munkamódszere a tesztelőknek, vagy a valódi támadóknak és hogy az milyen nyomokat tud hagyni a naplófájlokban. Be szeretnénk mutatni, hogy a valódi és a felbérelt támadók is gyakran sok nyomot hagynak a rendszerben, és azok igenis, a legtöbb esetben felderíthetőek többek között a naplófájlok elemzésével. Az eredményekből pedig következetéseket lehet levonni, például olyan eldönthető kérdésekre is lehet felelni, hogy ellopták-e a legfontosabb adatainkat vagy sem.

2. Háttér

A sérülékenység-vizsgálók, etikus-hackerek, vagy további nevükön behatolástesztelők nagyon hasonlóan végzik dolgukat, mint a valódi feltörők, hackerek. Egyfajta munkafolyamatot követnek, ami kombinálja az információgyűjtést és a sérülékenységkihasználi kisérleteket, legyenek azok sikeresek vagy sem. Ezeket a lépéseket ismétlik mindaddig amíg elérik céljaikat, vagy megakadnak egy bizonyos ponton. Sosem egyetlen lépésben törik fel a rendszert, hanem lépések szoksor napokig, hetekig, de akár évekig történő egymásutániságával. Ez pedig azt jelenti, hogy a naplófájlok elemzésével nagyon jó esélyeink vannak, hogy megtaláljuk támadások nyomait, jó esetben még azelőtt, hogy sikerrel járnak egy komolyan támadási céllal. Ez önmagában nem elég- túl sok a naplófájl, túl sok a támadást, ha mindent megnézünk, akkor örök időkig elemezgethetünk. Ha viszont van egy gyanús esemény, annak kivizsgálása során a kapcsolódó naplóbejegyzések mennyisége már korlátos, és többnyire lehet találni nyomokat, amelyek hasznosak lehetnek az esetleges sérülékenységek kiszűrésére, a támadók kizárására, a rendszer biztonságának erősítésére.

A támadás során alkalmazott két fő módszer a következő:

- Automatikus sérülékenységellenőrzők és információgyűjtő programok alkalmazása
- Manuális kihasználási próbálkozások

Az automata sérülékenységellenőrző programok általános ellenőrzésre valók, nem egy-egy programra. Ilyen ellenőrzők pl. a Nessus, Nexpose, vagy az ingyenes, de kevéssé használható OpenVAS. Vannak kicsit speciálisabb automata ellenőrzők, mint a web támadásokra szakosodott Acunetix Web Vulnerability Scanner, vagy a DefenseCode WebScanner. Még specifikusabb WPScan ami a weben belül is csak Wordpress hibákat keres.

Az automata sérülékenységellenőrzők abban jók, hogy nagyon sok előre ismert tesztet el tudnak végezni sok célpont felé, akár ezernyi célpont felé, több tízezernyi ismert teszttel. Hátrányuk, hogy sok naplóbejegyzést eredményeznek, ha tehát valódi hackerek ezeket használják, könnyen le is bukhatnak, mert detektálni lehet a vizsgálatok végzését. Ugyanakkor ezek az eszközök csak ismert, dokumentált sérülékenységeket találnak meg többségében, és a friss, nem ismert, vagy egy saját célra fejlesztett rendszer sérülékenyésgeit még akkor sem találják meg, ha azok megtalálása triviális lenne.

Mindezekért sosem végzünk automata sérülékenység-ellenőrzést manuális ellenőrzés nélkül (vagy csak elvétve). A manuális ellenőrzés épülhet az automata ellenőrzés eredményeire, azalapján az elemző új eszközöket, saját fejlesztéseket, új ötleteket használ fel, hogy hibákat tárjon fel. A manuális ellenőrzés akár sosem ismert hibákat, 0-day sérülékenységeket is feltárhat, sőt, sok ilyen felismert hiba dokumentálása és kezelése is megbízható etikus hackerek munkájára vezethető vissza, és a legtöbb esetben ezen a szakértők pénzt sem keresnek azon túl, hogy a munkabérüket a sérülékenység-ellenőrzésre kifizeti a megbízó.

A manuális ellenőrzés időrabló és nagyon felkészült személyzetet igényel, ezért minimalizálni szokták, noha ez lehet egy kulcs eleme a vizsgálatoknak.

2.1. Storyline

A mai feladat története a következő: Valaki telepített egy linux-alapú web szervert és feltelepített rá egy vadonatúj Wordpress alapú weblapot. Berakott pár kezdeti postot, de sok más nincs az oldalon. Ezután felbérelt valakit hogy próbálja meg ellenőrizni a rendszert biztonsági szempontból. Ennek sok esetben nincs értelme: Ha valóban minden új és jól beállított, akkor egy tesztelő nem fog semmit találni. De nem lehet tudni, hogy egy fejlesztő milyen banális hibát követett el, ezért az ellőrzésnek akkor is van helye, ha látszólag minden rendben.

A megbízott etikus hacker többféle módszert alkalmazott . Automata

sérülékenység-ellenőrzőket, sql-injection kihasználó eszközöket, hogy megszerezze a rendszergazda jelszavát, backdoor wordpress plugin telepítést, amin keresztül shell szintű hozzáférést szerzett a géphez a támadássorozat segítségével. Nem tudjuk, hogy elért-e root szintű jogokat a támadó. A támadó sok nyomot hagyott a naplófájlokban támadási kísérletei során, ezeket lehet elemezni, kiszűrni. Akár eszközeit is fel lehet mérni, és megtudni, mikor és milyen intenzitással támadott.

Ezalatt a gyakorlat alatt azokat az alapvető műveleteket póbáljuk megjeleníteni, amit a támadók elkövettek magas szinten, nem tudunk minden egyes apró műveletbe belemenni, mert arra nincs idő. Azt szeretnénk elérni, hogy a hackerek egyes műveleteit közelről érezzétek a naplófájlokból, hogy sugározzon belőle, hogy értitek, hogy egy támadóval van gond és értsétek a naplófájlban levő kapcsolatok, és azok szerepét amiért ezt a megértést meg lehet határozni. Ezen túl arra célzunk, hogy a nyomokat fel tudjátok dolgozni. A naplófájlokban levő információt meg lehessen érteni és a napi gyakrolatba állítani. Tudjuk pontosan, a naplófájl elemzés egy szakma, a bűnügyi feltárás egy részterülete, aminek specialisái vannak, mi azt szeretnénk, hogy példákon keresztül az alapokat megértéstek, és pl. elkerüljétek azt, hogy egy incenskezelés során egy potenciális bizonyítékot hanyag munkával károsítsatok.

3. Unix tools

A munka során alapvető UNIX/Linux eszközök használatára is szüksgé lesz. Ha ezeket az eszközöket hallomásból sem ismered, kérjük olvass utána. Nem lesz bonyolult, de aki nem hallott róla, annak kell még egy fél óra tanulás.

Minimum unix/linux ismereti lista:

- `cat` - fájl tartalom listázása
- `ls` - directory / könyvtár listázása . Többnyire `ls -la` or `ll` használatos.
- `less` - fájl szöveg tartalmának listázása és könnyű kezelése, pl. le- és felgörgetés.
- `cp` - copy file, fájl másolása. Pl: `cp a.tgz /tmp/b/z.tgz` - a.tgz átmásolva azonos jogokkal a célfájlba.

- `tar` - compress or decompress files, azaz fájlok kitömörítése és becsogolása a gzip algoritmus szerint . Minta: `tar xfvz <file.tgz>` - tömöríts ki egy helyi fájlt egy helyi alkönyvtárba egy tar tároló fájlformátm és gzip tömörített formátmumból egy megfelelő helyre.

- `grep "regexp" file` - csak azonak a soroknak a kilistázása egy fájlból, amelyek egyeznek egy reguláris kifejezésre amit a parancsorban adunk meg. A reguláris kifejezésekről sokat kell tanulnod, ha nem értesz hozzá! Nézd meg a következő linket https://www.cyberciti.biz/faq/grep-regular-expressions , vagy bármilyen más információforrsát a grepről vagy a reguláris kifejezésekről. Fontos, hogy a reguláris kifejezések az IT technológiában mindenhol jelen vannak, tehát mindenképpened kell éretened hozzá, ne akard megúszni, tanuld meg!

- `last` - kilistázza az utolsó belépéseket a rendszerbe, de csakúgy megadja az utolsó rendszerújraindítások (reboot) időpontjait is.) tárolási helye a *wtmp*[1] file. Paraméterekkel korábbi naplófájlban levő események is kikereshetőek, pl. előző hónap, de a napló-rotáció beállításai szerint sokan nemcsak 1-2 , hanem akár 12 hónapnyi naplót is eltárolnak, vagy még többet. Példa: `last -f wtmp.1` a belépéseket fogja kilistázni az előzző naplófájlból.

- `tail` - egy fájl utolsó sorainak kilistázása. Minta: `tail -n 5 file` - Ez a fájl utolsó öt sorát fogja kilistázni. Minta #2: `cat file | grep wget|tail -f 50` - ki fogja adni azokat a sorokat a fájlból, amelyek tartalmazzák a wget szót, de csak az 50 legutolsó ilyen sort.

- `wc -l` - megadja a fájl sorainak hosszát a kimeneten. Minta: `cat /etc/passwd | wc -l` - ki fog adni egy számot ami kb 50, attól függően hány sor van a passwd fájban.

- Több parancs egymásra építése: `cat file | grep "rabbit" | grep -v "hole" | tail -n 5` - veszi a file nevű fájl tartalmát → leszűri belőle a sorokat amiben a rabbit szó szerepel → de mindazokat eldobja ahol a hole szó is szerepel → és csak a legutolsó öt ilyen sort mutatja meg.

[1] https://linux.die.net/man/5/wtmp

Log rotáció (felcserélés, avultatás): A UNIX/Linux rendszereken a log rotáció, a naplófájlok folyamatos cseréje, avultatása egy tipikus tevékenység. Minden nap, vagy a hét egyik napján, vagy havonta egyszer néhány naplófájlt lezárunk, elrakjuk az archívumba és egy új naplófájlt kezdünk. A régi naplófájl tipikusan új nevet kap, pl. naplófájl.1 vagy akár tömörítjük is, naplófájl.2.gz. Egy idő után a nagyon régi naplófájlok beállítás szerint törlődnek. Nem kell elfogadnunk az alapbeállításokat, egy tipikus operációs rendszerhez képest egy konzervatív rendszergazda jóval megnövelheti a mentések számát, hogy pl. korábbi biztonsági incidensekre is legyenek rendelkezésre álló naplófájlok. Nem baj, ha egy évre is vissza tudunk nézni, ha kell, vagy még többre. A naplófájlok cseréjét tipikusan *crontab* scriptek végzik[2]. Példaképp, a `wtmp` naplóállomány tipikusan hetente egy konkrét időpontban cserélődik. A rig fájl rotálásra kerül és törlődik, a mostani elmentésre kerül a régi helyére és egy új üres fájl lesz a friss. A regi wtmp új neve `wtmp.1` és egy új üres `wtmp` fájl keletkezik. Egy jó rendszergazda nem elégszik meg ennyivel és módosítva a beállításokat esetleg jóval több hétre, vagy hónapra menti a belépések naplóját.

Esetenként a naplófájlok tömörítve kerülnek tárolásra, például a `access.log.2` tömörítésre kerül és `access.log.2.gz` vagy hasonló fájlnévvel, tömörítve kerül tárolásra.. A fájlok méretétől is függhet, hogy mennyi ideig tároljuk azokat. A `wtmp` nagyon kicsi fájl tipikusan, ezért dönthet a rendszergazda, hogy ezt a többihez képest sokkal tovább tárolja, pl. akár havonta rotálva 24 hónapig is. Ilyenkor akár egy `wtmp.23.gz` vagy hasonló fájl is megtalálható lehet a naplófájlok között.

3.1. Ellenőrző kérdéssek

Nem készültél fel a laborra ha hasonló kérdésekre nem tudsz válaszolni.

- Mi az eredménye a következő parancs végrehajtásának? `ls -la /etc/hosts`?
- Mi az eredménye a következő parancs végrehajtásának? `cat /etc/hosts`?
- Mi az eredménye a következő parancs végrehajtásának?`cat /var/log/syslog`?

[2] https://linux.die.net/man/5/crontab

- Hogy lépsz ki a következő parancsból `less /var/log/syslog`?
- Hány sorra számítasz eredménynek egy ilyen parancsra `grep dsfsr4krikszkraksz /var/log/syslog`?
- Hány sorra számítasz eredménynek egy ilyen parancsra `cat /var/log/apache2/access.log | wc -l`?
- Megérted miért van olyan eredménye a köv. kérdésre `"abrakadabra"` ennek `".*ka..br.*"`?
- Negérted miért `"[0-9][0-9]"` matchel ezzel? `00 01 02.. 99`?
- Ha a fájl mérete a /var/log/syslog fájlnak 100 kb akkor a `cat /var/log/syslog | wc -l` lehet kb 11000?
- Miért ad egy rendszeren a `last -f /var/log/wtmp.5` belépési információkat úgy 5-6 hónappal azelőtt?

4. Feladatok

A laborgyakorlat bemutató videóból és önálló feladatokból áll.

4.1. Bemutató video

A labor vezetett részében bemutatjuk pár képernyőmentésen keresztül, hogyan dolgozott a támadó, az etikus hacker, milyen képernyőket látott munkája eredményeként. Nem megyünk bele a részletekbe, mert az túl sok időt vesz igénybe, de azt szeretnénk érthetővé tenni, hogy a folyamatban több eszközt használt, nem mindig járt sikerrel, és egyre fókuszáltabban koncentrálva elérte célját, vagy legalábbis úgy tűnik,... Ezután jönnek a hallgatók önálló feladatai.

4.1.1. Letöltés, felkészülés

- Töltsük le a megtámadott rendszer logjait a virtuális gépre, mert linux eszközöket fogunk használni
- Egy alkönyvtárban tömörítsük ki az etc.tgz fájl tartalmát a tar xfvz etc.tgz parancs segítségével
- Fontos, hogy a root felhasználóként való kitömörítés akár biztonsági veszélyeket is rejthet magában (a kitömörített fájlok esetében a root root marad, de a többi felhasználó kitömörített fájljai azokkal az uid-kkel és gid-ekkel jönnek létre, amit az eredeti rendszer használt. Előfordulhat, hogy egy betömörített állományban bob 1002-es user volt, de ahol kitömörítjük, ott mallory az 1002-es felhasználó, és így az ő birtokában lesz pl. néhány forráskód fájl és módosíthatja is akár ezeket, mi meg később a módosítottat fordítjuk újra. Setuid gond is lehet, bár erre lehet, hogy részben felkészültünk. Mindenesetre ez gondot tud okozna. Ha nem root-ként tömörítünk ki, akkor nincs probléma, hisz a user más nevében nem tud kitömöríteni, így minden egy felhasználó kezén lesz. A gond az, hogy így meg backupot nem lehet készíteni, hiszen a kitömörítés veszteséges: elveszítjük a felhasználói azonosítókat. Nehéz dolog a rendszergazda élete.
- a kitömörített fájlok az etc alkönyvtárban az elemzett gép etc alkönyvtárjának tartalma, teljesen (beleértve a kódolt jelszavakat is, amelyeket akár fel is lehet törni)

- A tar xfvz log.tgz parancs segítségével tömörítsük ki a /var/log alkönyvtár teljes tartalmát. A kitömörítés egy log nevű alkönyvtárba fogja rakni az eredményeket

- A gzip -d lslar.gz parancs segítségével tömörítsük ki az lslar fájlt. Ez a fájl a gépen lefuttatott ls -laR parancs eredménye és tartalmazza a teljes directory listát. Ez azért fontos, mert tartalmazhat információt mindenféle létrehozott suid-os programról, vagy bármilyen gyanús fájllal kapcsolatban. Nagyon hasznos tud lenni utólag, ha van egy pillanatképünk amiben ellenőrizhetünk kérdéseket (pl. volt-e .ssh directoryja egy adott felhasználónak és ott volt-e .authorized-keys fájl, ami lehetővé teszi ssh belépést jelszó nélkül)

Az apache access.log (combined verzió) formátuma: minta:

10.44.1.84 - - [01/Sep/2019:15:05:00 -0500] "GET / HTTP/1.1" 200 20480 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36"

Csak a fontosabb elemeket kiemelve: A log első mezőjében a lekérdező IP címe található, az első kötjelt követő kötőjel azt jelzi, hogy nem volt ún HTTP BASIC hitelesített az ügyfél. Ezután szögletes zárojelben a leérdezés ideje látható az időzóna megjelölésével (-0500 fura, mert a UTC a +0000), utána macskakörmökben látszik a HTTP kérés típusa célja és verziója, tehát a GET a típus, a / a letöltött oldal címe és az 1.1 a HTTP verziója. A státusz kód következik: 200 - rendes letöltés. (Ugye emlékszel még, a HTTP letöltési kódokkal foglalkoztunk az első mérésen amikor az nginx-et állíottuk be redirect-re?!) A letöltött adat hossza 20480 byte. Végül lényeges az agent-string naplóba mentése ami a kliens által kiadott, akár hamis azonosító, hogy milyen böngésző végezte a letöltsét, egész részletesen. Sok sérülékenységkereső eszköz "elárulja magát" és az agent stringbe a valódi támadó szoftver nevét és verziószámát írja. Többségükben megkérhetőek az eszközök, hogy hamisítsanak oda valami mást, de pl. egy etikus hacker általában nem próbálja álcázni magát, hiszen nem cél az elrejtőzés, sőt, vizsgálhatják azt is, hogy a hálózati operátorok képesek-e észervenni, hogy ők milyen eszközökkel, módszerekkel, célpontokkal, milyen támadásokat végeztek. Noha a mérés során korlátozott számú feladatot adtunk ki, a érdemes otthon később átnézegetni a naplókat, hogy még milyen érdekes felfedezéseket lehetett volna találni a logokban, mert van még jópár, amire nem csináltunk feladatokat!

4.1.2. Fájlok

Alapvetően három fájlcsoporttal lehet dolgozni: az etc könyvtár tartalma a konfigurációs állományok tartalmát hordozza, és főként csak a beállítások vizsgálatára használható, de ott nagyon fontos tud lenni egy nyomozásban. Például: ha a támadó kiskaput, backdoort telepített, annak nyoma lehet az etc struktúrában. Fontos fájl az lslalr.txt, ami a gép teljes alkönyvtárstruktúráját listázza. Ehhez sem szoktunk csak úgy nyúlni. Akkor van rá szükség, amikor már van valami nyomunk, és szeretnénk tudni, hogy a máshonnan jelölt fájl tényleg létezik-e, mikori a dátuma, mik a jogai stb. Az lslar.txt fájl jelentősége, hogy a nyomozás elején könnyű és gyors létrehozni és utólag egy hiteles képet mutathat a fájlokról még akkor is, ha nem bizonyító erejű, nem tartalmaz hash-eket. Ez egy történelmi bizonyíték, érdemes megcsinálni, "mert még jó lehet valamire." A mérés során felhasznált legfontosabb fájlok a log alkönyvtárban vannak, ami a /var/log fájljait tartalmazza. Mivel alapvetően webes támadások történtek, az apache2/access.log és ugyanitt az error.log analízise fontos. Érdemes még megemlíteni a /var/log/messages és /var/log/syslog fájlokat, amik a legtöbb oprendszer szintű logokat tartalmazzák, és persze a /var/log/wtmp ami a belépési adatokat és a /var/log/wtmp.1 ami az egy hónappal vagy héttel korábbi rotált log adatokat tartalmaza.

4.1.3. Első lépések, gyakorlás

Nézzünk körül, mivel is van dolgunk. Se mi, se a hallgató nincs teljesen képben mit is látunk, így pl. midnight commanderből, vagy bármiből, bárhol nézzünk bele a fájlokba. Mivel lehet, hogy sosem látott a hallgató naplófájlt, nézzük meg, hogy dátum szerint növekszik, vagy a directory listben, hogy kb. milyen elemeket látunk. Vegyük észre a nagy méretüket, és hogy ez elég áttekinthetetlen manuálisan, de persze minden egyes sor értelmezhető magában.

- pl. midnight commanderben a fájl listában használjuk a keresés funkciót az "rws" stringre többször. Emlékeztessük a hallgatókat az előző mérésre, hogy ez egy setuid-es program és mondjuk el, hogy ez azt jelenti, hogy ennek akár root joga is lehet, ha támadó rakta be, akkor ez lehet egy backdoor is. Kérdezzük meg miért van a talált suid-es programnak root joga, miért kellhet?

- a log alkönyvtárban nézzük meg a syslog fájl tartalmát. Ez az egyik legfontosabb operációs szintű logfile, de a legkevesebb támadás-közeli információ lesz benne, mert a támadó nem sokat dolgozott az operációs rendszeren, inkább a webet támadta
- bemutatjuk a "last" parancs használatát fájl specicikálásával. last -f wtmp. két logint látunk a 10.44.2.208 címről. A logrotate által lecserélt wtmp.1-re lesz feladat.
- nézzünk bele a log/apache2/access.conf és error.log fájlokba. . Semleges sorokat nézegessünk meg, hogy mit tartalmaznak. (ip, user auth, idő, lekérdezés jellege, URL, code, méret, agent string stb.) nem fontos a teljesség, nem arról szól a mérés
- menjünk be az etc/motd fájlba és nézzük meg, hogy ezt írja ki a fájl ha beloginolnánk a gépre (Message-of-the-day)

4.2. Feladatok

4.2.1. Feladat

Mi a debian verziója a vizsgált operációs rendszernek? Az /etc/debian.version tartalmazza ezt az információt

4.2.2. Feladat

Lépett-e be a vblog és a root felhasználón kívül bárki a wtmp.1 fájl szerint szeptember 1 -október 20 között?

4.2.3. Feladat

Mikor bootolt utoljára a rendszer a wtmp fájlok vizsgálata alapján? (ellenőrizendő: /var/log/wtmp és /var/log/wtmp.1)

4.2.4. Feladat

A nessus nevű sérülékenységfelderítő eszközt honnan futtatta a támadó? Az apache access logokban a nessus agent stringnek "Nessus" szöveget használta

4.2.5. Feladat

Az access.log.1 fájlban látható, ahogy a támadó elkezdi használni a feltöltött shellkódot (plugins/shell/shell.php). Az első parancsokat GET paraméterként írta be, csak utána nyitott távoli shellt. Mi volt az első lefuttatott parancs amit kiadott?

4.2.6. Feladat

Hozott-e létre új 0-ás uid-jű felhasználót a támadó (kiskapu) (ellenőrizendő: /etc/passwd fájl tartalma)

4.2.7. Feladat

Milyen verzióját használta a támadó az sqlmap-nek? (web napló fájl alapján access.log és/vagy access.log.1 stb.)

4.2.8. Feladat

A támadó feltöltött egy első plugint a wordpress-be próbálkozásnak az action=upload-plugin log üzenetnél található módon. De ez nem működött. A GET /wp-content/upgrade/info9/info9.php HTTP/1.1 kérésre nem 200-as error code generálódott, hanem? (hasonlóképpen a /var/log/apache/access.log.1 fájl vizsgálata lehet szükséges)

4.2.9. Feladat

Milyen plugint töltött fel a támadó utoljára, amit használni is tudott? Végrehajtott egy action=upload-plugin hívást és utána rögtön meghívta a feltöltött backdoort. Mi a neve a php fájlnak. A megoldás az access.log.1 része.

4.2.10. Feladat

Honnan futtatta a támadó a WPScan scriptet? (az apache.log.1 aktuális sorának eleje tartalmazza az IP címet)