IT Biztonság (VIHIAC01)
Házi feladat

Malware

Szerzők:
Bencsáth Boldizsár, Futóné Papp Dorottya

2021. március 4.

Tartalomjegyzék

1. Háttér

1.1. Malware elemzési tudnivalók

A házi feladat megoldásával a hallgató betekintést kap a malware elemzés területébe és elvégzi egy ismeretlen, potenciálisan rosszindulatú bináris elemzésének első lépéseit. A malware elemzés folyamatának gyorsítására és segítésére több online szolgáltatás is elérhető. A házi feladat megoldásához az alábbi rendszereket ajánljuk:

- https://www.virustotal.com/gui/
- https://any.run/, ami a Cuckoo Sandboxot[1] használja

A házi feladat megoldása előtt kérjük a hallgatót a fenti szolgáltatások megismerésére! Használhatja a szolgáltatások dokumentációit, illetve az Interneten nyilvánosan elérhetőek bevezetők és ismertetők (többnyire angolul)[2].

A házi feladatot úgy dolgoztuk ki, hogy a leírást követve a hallgató számítógépére semmilyen kockázatot nem jelent a házi feladat elvégzése, hiszen nem adunk ki valódi kártékony binárist. Ennek ellenére nem vállalunk felelősséget, ha a hallgató jelentősen eltér a kért feladattól, pl. letölt malware mintákat és futtatja azokat. Amennyiben bármilyen kérdés merülne fel a házi feladat megoldása során, szívesen állunk a hallgató rendelkezésére e-mailben!

1.2. Kód polimorfizmus

A malware minták készítői számára elemi fontosságú, hogy az általuk készített minták megértése és detekciója minél nehezebb legyen. Ezt a célt sok esetben ún. kód polimorfizmussal érik el, ami azt jelenti, hogy a mintákat úgy készítik el, hogy a kód futási időben módosítsa önmagát. Ez sok szempontból nehezíti az elemzést és a detekciót. Egyrészt, a háttértáron perzisztens módon tárolt bináris kód nem egyezik meg a memóriába betöltött ténylegesen futtatott kóddal. A mintát általában úgy készítik el, hogy annak tárolt formájában ne legyen közvetlenül kártékony kód, csupán azok az utasítások, amik a memóriában visszaállítják a minta kártékony funkcionalitását implementáló utasításokat.

[1] https://cuckoosandbox.org/

[2] Pl. VirusTotal: https://www.youtube.com/watch?v=Sf2UdT53yFw&t=34s
any.run: https://www.youtube.com/watch?v=YLnBPNenH9k

Gépi kód szinten több lehetőségük is van a támadóknak megvalósítani a kód polimorfizmust. Egy egyszerű megoldás például, ha a kártékony funkcionalitásért felelős kódok valamilyen véletlen bitsorozattal XOR-olják ($a \oplus k$, ahol a jelöli az utasításokat, k pedig a véletlen bitsorozatot). Ekkor a bináris kódszegmense tartalmazza az XOR-olt utasításokat, valamint azt a kódot, ami visszaállítja az eredeti utasításokat az XOR művelet segítségével ($(a \oplus k) \oplus k = a \oplus (k \oplus k) = a$, a véletlen bitsorozatot belekódolhatják az utasításokba, vagy akár globális változóként tárolhatják). Ezt a sémát természetesen tetszőlegesen lehet komplikálni bonyolultabb átalakításokkal, pl. titkosítással. A legtrükkösebb malware minták gyakorlatilag egy miniatűr virtuális gépet implementálnak saját utasításkészlettel, a bináris kód pedig a saját utasításkészlet és a CPU között just-in-time fordítóként működik.

Természetesen nem csak gépi kód szinten lehet ilyen trükköket alkalmazni. Amennyiben magasabb szintű nyelven, pl. Java, Android, C#, készítik a malware mintát, különböző programozási technikákat is lehet használni a valós funkcionalitás elrejtésére. Erre a célra pl. reflexiót és dinamikus betöltést lehet használni. A reflexió egy olyan menedzselt nyelvekhez elérhető technika, amivel az alkalmazások futásidőben monitorozhatják és változtathatják saját viselkedésüket. Lehetőségük van lekérdezni a különböző osztályok struktúráját, ide értve a tagváltozókat és metódosokat, valamint tetszőleges metódusokat meghívhatnak, osztályokat példányosíthatnak. A reflexió sajátossága, hogy megkerüli az osztályok, metódusok, tagváltozók definiálásánál használ `public`, `private` és `protected` láthatósági jelzőket és az osztály minden részéhez hozzáférést nyújt. A dinamikus betöltés segítségével nem kell fordításidőben megadni a felhasznált osztályokat és interfészeket, elég futási időben kérni a just-in-time fordítótól új osztályok betöltését.

E két technikával az következő módon rejthetik el a támadók a valós funkcionalitást:

1. A valós funkcionalitást megvalósító osztályok köztes nyelvi reprezentációját (vagy annak valamilyen transzformációját) beleírják a forráskódba. A köztes nyelvi reprezentáció bitsorozatára ekkor sorozatként vagy tömbként tekintenek, amit vagy bájttömbként, vagy valamilyen sztringgé alakítással, pl. Base64 kódolás, lehet beletenni a forráskódba.

2. Ha használ a készítő valamilyen transzformációt (pl. Base64 kódolás), akkor annak az inverzét implementálja a forráskód (pl. Base64 dekódolás), hogy a memóriában visszaálljanak a valós funkcionalitást megvalósító osztályok.

3. Dinamikus betöltés segítségével futásidőben betöltik ezeket az osztályokat.

4. Reflexió segítségével példányosítják a valós funkcionalitást megvalósító osztályokat és meghívják a kártékony metódus(oka)t.

2. Feladatok

2.1. Ransomware minta elemzése

Munkatársunk egy ransowmare tevékenyégét fedezte fel. Megadta a minta metaadatát: a program MD5 lenyomtatát, ami `f83fb9ce6a83da58b20685c1d7e1e546`. Megjegyezzük, hogy az MD5 lenyomatnál léteznek más, pontosabb azonosítást lehetővé tevő lenyomatok, pl. SHA256, de az MD5 lenyomatok alapvetõen ma is használhatóak malware minták azonosításra. A lenyomat alapján szerezzen információt a mintáról!

2.1.1. A feladat leírása

Első lépésként látogassa meg a VirusTotal elemző oldalt és keresse ki a megadott lenyomathoz tartozó adatlapot. Az adatlapon sok érdekes információ megtalálható a mintáról (pl. hogy a vírusirtók detektálják-e a mintát és ha igen, hogyan; milyen metaadatok találhatóak a fájl fejlécében; stb). Tanulmányozza a kikeresett adatlapot!

Második lépésként nézze meg a minta futásidejű elemzésének eredményeit az any.run szolgálgatás felhasználásával! A rendszer már elemezte ezt a mintát korábban, ezért elég a nyilvános feladatok között keresnie. Keresse ki a 2020.01.14-i elemzési eredményeket és tanulmányozza azokat!

Tanácsok a megoldáshoz:

- A VirusTotal weboldalán létezik keresési szolgáltatás, ami többféle lenyomatot elfogad, köztük az MD5-öt is.
- Az any.run szolgáltatás weboldalán a `Service` linken keresztül lehet új elemzési feladatokat feltölteni és régebbi nyilvános elemzési eredményeket kikeresni. A keresés itt is történhet lenyomatok alapján.

2.1.2. A megoldás dokumentációjával szembeni elvárások

A feladat megoldását képernyőképekkel kérjük dokumentálni. A képernyőképeknek az alábbiakat kell rögzíteniük:

- A VirusTotal adatlap `Detection` fülén található információk

- A VirusTotal adatlap `Details` fülén található `Basic properties`, `History` és `Names` bekezdések tartalma
- A VirusTotal adatlap `Behavior` fülén található `HTTP requests` bekezdés tartalma
- Az any.run szolgáltatás elemzési eredményeit mutató adatlap

A képernyőképeket egyetlen ZIP archívumba csomagolva kérjük beadni.

2.2. Matrijava

Munkatársunk egy furcsa malware mintát talált a mintaadatbázisunkban. Futtatás során a minta jelszót kér a felhasználótól, ami arra enged következtetni, hogy a támadó próbálja nehezíteni az elemzést: csak a jelszó ismeretében figyelhető meg a kártékony viselkedés. Sajnos munkatársunk nem tudja a jelszót, ezért segítséget kért.

2.2.1. A feladat leírása

A hallgató feladata visszafejteni a megadott minta működését, kitalálni a jelszót és értelmezni a kártékony viselkedést. A visszafejtéshez segítségképpen az egyetmi felhőben található virtuális gépen elérhető a JD-CLI szoftver[3], ezzel lehet Java class fájlokat decompile-olni, vagyis visszaállítani a forráskódjukat. A program a `Downloads` mappában található egy JAR fájlként (`jd-cli.jar`) és a parancssoron a `java -jar jd-cli.jar` paranccsal futtatható.

Tanácsok a megoldáshoz:

- A visszafejtendő minta természetesen nem tartalmaz semmilyen kártékony funkcionalitást, csupán kiír a parancssorra egy titkot.
- A minta egy régebbi Java Runtime Environment verzióra készült, ezért nem érdemes megpróbálni futtatni a virtuális gépben.

[3] https://github.com/kwart/jd-cli

- A minta készítői nem csak egy rétegnyi polimorfizmust kódoltak az alkalmazásba, ezért érdemes a visszafejtés folyamatát automatizálni. Az automatizáláshoz bármilyen nyelvet használhat, de meg kell tudnia hívni parancssoron a JD-CLI szoftvert.
- A visszafejtés során szüksége lehet a Java dokumentációra bizonyos osztályok működésének megértéséhez.

2.2.2. A megoldás dokumentációjával szembeni elvárások

A feladat megoldását a visszafejtés folyamatának dokumentálásával kérjük bizonyítani. Ehhez kérjük leadni:

- A visszafejtést automatizáló kommentezett forráskódot egyetlen `txt` fájlként, valamint
- A „kártékony" viselkedést implementáló legbelső osztály visszafejtett forráskódját szintén egyetlen `txt` fájlként.

A fájlokat egyetlen ZIP archívumba csomagolva kérjük beadni.