IT BIZTONSÁG

5. forduló

A kategória támogatója: EURO ONE Számítástechnikai Zrt.

RENDELKEZÉSRE ÁLLÓ IDŐ:

45:00

Ismertető a feladathoz

A támadók többek között ezt a csatolmányban található .NET alkalmazást hagyták több Windows gépünkön. Úgy tűnik, valamelyest obfuszkálták, de úgy érzed, a megfelelő eszközök segítségével felfedheted a titkait.

A fordulóhoz erősen ajánlott egy .NET decompiler/debugger, pl. dnSpyEx (https://github.com/dnSpyEx/dnSpy) használata.

A forduló indítása előtt bizonyosodj meg arról, hogy az eszközöd képes visszafordítani és sorról-sorra debuggolni 32-bites .NET Framework PE állományokat!

Figyelem: bár a feladathoz tartozó futtatható állomány nem okoz kárt a gépben és a feladatlap indulása idején nem létező webcímhez próbál csatlakozni, a munkakörnyezetedben futó antimalware megoldások kártékonynak jelölhetik. A biztonság kedvéért ajánlott izolált sandboxban vagy virtuális gépben futtatni és/vagy a tűzfalon a program minden be- és kifelé történő kapcsolódását blokkolni.

Felhasznált idő: 02:05/45:00 **Elért pontszám:** 0/75

1. feladat 0/5 pont

Mi az a plaintext jelszó, amit a program a belső titkosított bináris tartalom visszafejtéséhez használ?

Válaszok

A helyes válasz:
5uper_53cr37_p4ssw0rd!

Super_53cr37_p4ssw0rd!

Magyarázat

ILSpy vagy dnSpy(Ex), de akár még egy szimpla hex editor vagy a strings parancs segítségével is statikusan megtalálható az UTF-16LE enkódolású plainstring jelszó a Main metódus közepén:

```
byte[] array8 = hashAlgorithm.ComputeHash(Encoding.ASCII.GetBytes("5uper_53cr37_p4ssw0rd!"));
```

Amennyiben a strings toolt használjuk a jelszó megtalálására, az encodingot little-endian 16-bitesre kell állítani, ugyanis alapesetben csak ASCII-enkódolású stringeket keres:

```
$ strings -e l ITMapper.exe
```

2. feladat 0/10 pont

Milyen fajta visszatérő algoritmust használ a program a titkosított tartalom dekriptálásához?

Válasz

- ○ AES
- ◯ XOR
- ○ RSA
- ○ SHA512

Magyarázat

Szinte közvetlenül a jelszó hashelése után látunk egy for-ciklust, amely végigmegy a 4. tömb összes byte-ján, és mindegyiket az alábbi kód szerint módosítja:

```
ref byte ptr = ref array4[i];
ptr ^= array8[i % array8.Length];
```

A .NET-ben és egyéb C-alapú nyelvekben a ^= operátor az XOR-assignment műveletet jelöli, vagyis a művelet bal- és jobboldali értékekeit összeXOR-ozva elmenti az eredményt a baloldali változóba. Jelen esetben az XOR művelet a hashelt jelszó segítségével felfedi, dekódolja a 4. tömbbe beágyazott bináris exe fájlt, amit később majd futtatni is fog.

3. feladat 0/10 pont

Mely registry bejegyzés jelenlétét ellenőrzi a program, és állítja le önmagát, amennyiben megtalálja? A bejegyzés nevét (registry value name) add meg, nem kell a teljes kulcs/alkulcs-útvonalat beírni!

Válaszok

A helyes válasz:
WinServiceMonitor
winservicemonitor
WINSERVICEMONITOR

Magyarázat

Ez, és a további kérdések megválaszoláshoz egyre inkább szükség lesz a dinamikus analízisre, az állomány kézi debuggolására. Erre a dnSpyEx vagy dnSpy eszközök a legalkalmasabbak.

Az ITMapper állomány Main metódusának legvégén a program betölti az XOR-ral kicsomagolt exe fájlt .NET Assemblyként, majd az Invoke metódussal meghívja annak entrypointját, a program futását ezzel oda átirányítva. Hogy ezt a dnSpyEx debuggerével követhessük, még az Invoke parancs meghívása előtt F11 (Step Into) megnyomásával be tudunk lépni az Invoke metódus implementációjába, ahol a

```
return this.UnsafeInvokeInternal(obj, parameters, arguments)
```

sornál,

azon belül pedig a

```
object result = RuntimeMethodHandle.InvokeMethod(obj, arguments, this.Signature, false);
```

sornál F11-gyel be tudunk lépni a memóriából betöltött, beágyazott .NET exébe.

Belépéskor a dnSpyEx automatikusan betölti, analizálja és visszafejti nekünk az egész assemblyt, majd a Main metódus első utasításánál megállítja a debuggert, így kézzel könnyen végig tudunk lépkedni a beágyazott program utasításain.

Az újonnan betöltött, "Embedded" nevű assemblyben az első lépés, amit a program tesz, hogy az alábbi módon ellenőrzi egy registry bejegyzés jelenlétét, és amennyiben az jelen van, idő előtt kilép:

```
if (Registry.LocalMachine.OpenSubKey("SOFTWARE").OpenSubKey("Microsoft").OpenSubKey("Windows").OpenSub
```

Itt a registry bejegyzés konkrét neve base64-gyel lett enkódolva, de azt tetszőleges online toollal dekódolva, vagy akár csak az utasítást átlépve és a dnSpyEx "Locals" ablakára tekintve megszerezhetjük a bejegyzés nevét:

```
System.Text.Encoding.GetString returned "WinServiceMonitor"    string
```

4. feladat 0/10 pont

Kizárólag milyen nyelvű Windows rendszereken aktív a kártevő?

Válasz

- amerikai angol
- német
- magyar
- orosz

Magyarázat

Továbbhaladva az Embedded assemblyben, a registry ellenőrzés után a program lekéri a telepített rendszernyelvet, és összehasonlítja egy MD5 hash-sel:

```
if (!MD5.Create().ComputeHash(Encoding.ASCII.GetBytes(CultureInfo.InstalledUICulture.ThreeLetterISOLan
```

Az MD5 hash a dnSpyEx-ből többféle módon kinyerhető úgy, hogy hexdigest formában kereshető legyen. Ennek több módja is van, tetszőleges online decimal-to-hex convertereket vagy akár a CyberChefet is felhasználhatjuk erre a célra.

A dnSpyEx elhagyása nélkül is megtehetjük ezt úgy, hogy a View -> Options -> Decompiler (C# / Visual Basic) beállításaiban alul a "Hexadecimal numbers" checkboxot bepipáljuk, és így már csak a vesszőket és a szóközöket kell eltávolítani:

FE1B3B54FDE5B24BB40F22CDD621F5D0

Az így reprezentált MD5 hash-re rákeresve pedig látjuk, hogy a "hun", vagyis a magyar rendszernyelvvel hasonlította össze a program a telepített nyelvet, és amennyiben megegyezik vele, folytatja a futást, egyéb esetben kilép.

5. feladat 0/15 pont

Melyik beépített Windows rendszereszközt futtatja a program a parancssoron (cmd) keresztül?

Az eszköz nevét az útvonala, kiterjesztése és paraméterei nélkül add meg, pl.: netsh!

Válaszok

A helyes válasz:

systeminfo
systeminfo /FO CSV
systeminfo.exe
systeminfo.exe /FO CSV
C:\Windows\System32\systeminfo.exe
C:/Windows/System32/systeminfo.exe

Magyarázat

Az elérhető fizikális memória ellenőrzése után egy különös stringet dekódol a program egy foreach loop segítségével, majd azt HTML-dekódolva egy process-argument stringgé alakítja és átadja a cmd.exének:

```
ProcessStartInfo processStartInfo = new ProcessStartInfo("cmd.exe", HttpUtility.HtmlDecode(Encoding.AS
```

Itt meg lehetne állni analizálni a logikát és rájönni mit csinál pontosan, de egyszerűbb inkább az egész foreach loop után csak egy breakpointot helyezni, és a Locals ablakban megnézni, mire dekódolta a stringet a HtmlDecode hívás:

```
System.Web.HttpUtility.HtmlDecode returned    "/c systeminfo /FO CSV" string
```

A /c a cmd.exe egyik switch-e, így a ténylegesen futtatott rendszereszköz a systeminfo.

6. feladat 0/5 pont

Milyen típusú kártevő ez a szoftver?

Válasz

- ( ) worm
- (•) spyware
- ( ) ransomware
- ( ) RAT/trojan

Magyarázat

Mivel a kártevő egyetlen aktív feladata az adatgyűjtés és annak elküldése, és se nem terjeszkedik, se nem titkosít fájlokat a fájlrendszeren, illetve távoli szerverről jövő utasításokra sem vár, csak a spyware lehet a helyes megoldás.

7. feladat 0/10 pont

A program látszólag két különböző végponttal kommunikál, de az egyik egy csali.

Milyen módon titkosított a valódi távoli végponttal történő kommunikáció?

Válasz

- ○ AES
- ○ XOR
- ○ RSA
- ◯ semmilyen

Magyarázat

A valódi végpont ebben az esetben az a cím, ahová a systeminfo outputja kerül kiküldésre. Bár a kódban jelen van az RSA és az XOR titkosítás használata is ezen a ponton, azok nem a systeminfo által összegyűjtött adatot titkosítják, hanem csak a metódus nevét, ami aztán később nem is kerül elküldésre. A systeminfo output bár egy URL-encodingon átesik, de az közel sem nevezhető titkosításnak.

8. feladat 0/10 pont

Mi a valódi távoli végpont webcíme? Add meg *protokoll://domain-nev.tld* formátumban, pl.: ftp://sparring-program.io

Válaszok

A helyes válasz:
http://shaken-not-strlen.pl
http://shaken-not-strlen.pl/
http://shaken-not-strlen.pl/ep.php?agentid=21002438

Magyarázat

Egy megfordított plain string jelöli a http://shaken-not-strlen.pl/ep.php?agentid=21002438 címet ahová a systeminfo outputja kerül küldésre.

Becsapós lehet a lentebbi try ágban található 52.33.171.6 IP cím is, ahová az RSA-titkosított adat kerül küldésre, de arra a címre nem kerül semmi gépről összegyűjtött adat, illetve az a kódrészlet le sem fut a nem teljesülő if statement miatt, ezért nem azt tekintjük a "valódi" végpontnak.

KÉSZÍTETTE

Megjelenés

Világos