SYNACKTIV
DIGITAL SECURITY

PowerShell for Pentesters

Juste toi, PowerShell et la cible ?
Challenge accepted!

Présenté le 07/04/2016

Par Damien Picard

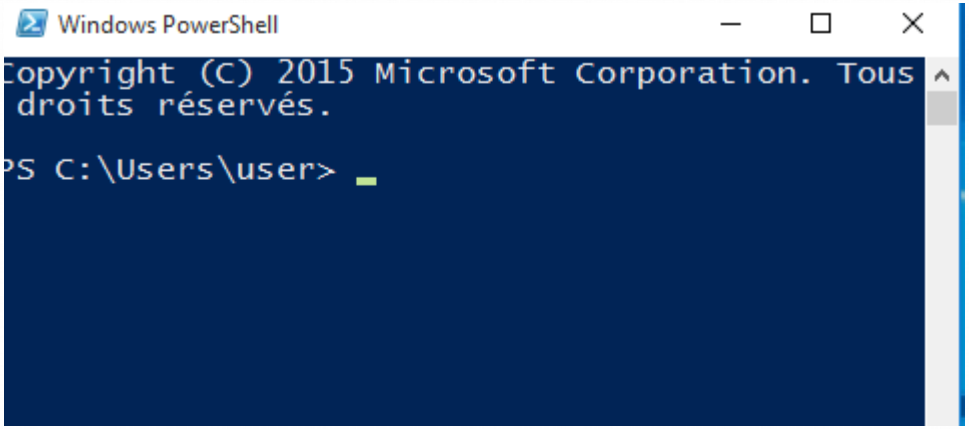
Whoami /all

- **Damien Picard**
- **Expert sécurité chez Synacktiv**
- **Passionné de sécurité, des CMS jusqu'aux OS !**

Get-Help PowerShell

- **Intégré par défaut à partir de Windows 7**
  - Remplaçant de cmd.exe
  - Basé sur la plateforme .NET
  - Fonctionne sur la base de cmdlets
  - Orienté objet
  - Enfin un shell en plein écran ...

    À partir de windows 10 seulement
  - Environnement de scripting (ISE) Inclus !



- **Utilisé pour l'administration système**
- **Et l'administration d'Office365**

Restrictions

- **Actives par défaut**
  - Les scripts doivent être signés
- **Signature de script type *Authenticode***
- **Signature vérifiée par l'interpréteur PowerShell**

```
PS C:\Users\user> .\Desktop\script.ps1
.\Desktop\script.ps1 : Impossible de charger le fichier
C:\Users\user\Desktop\script.ps1, car l'exécution de scripts
est désactivée sur ce système. Pour plus d'informations,
consultez about_Execution_Policies à l'adresse
http://go.microsoft.com/fwlink/?LinkID=135170.
Au caractère Ligne:1 : 1
+ .\Desktop\script.ps1
+ ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : Erreur de sécurité: (:) [], P
   SSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
```

Restrictions

- **Objectif : exécuter un script dans l'interpréteur**
  - Avec la configuration par défaut

- **Nombreuses manières de le contourner**
  - Les plus simples :

    Set-ExecutionPolicy Bypass -Scope CurrentUser

    Powershell.exe -ExecutionPolicy Bypass -File script.ps1

  - Les intermédiaires :

    Get-Content script.ps1 | Invoke-Expression

  - Les farfelues :

```
PS C:\temp> function Disable-ExecutionPolicy {($ctx = $executioncontext.gettype(
).getfield("_context","nonpublic,instance").getvalue( $executioncontext)).gettyp
e().getfield("_authorizationManager","nonpublic,instance").setvalue($ctx, (new-o
bject System.Management.Automation.AuthorizationManager "Microsoft.PowerShell"))
}
PS C:\temp> Disable-ExecutionPolicy
```

Restrictions : Les jails

- **Plusieurs mécanismes de « jail » spécifiques disponibles**
- **Les *VisibleCmdlets***
  - Liste blanche de cmdlets
- **Les *LanguageModes***
  - **FullLanguage** : Tout est disponible
  - **ConstrainedLanguage** : Suppression des fonctionnalités avancées .NET
  - **RestrictedLanguage** : Restriction des fonctionnalités POO
  - **NoLanguage** : Seuls les appels de cmdlets sont permis

Restrictions : VisibleCmdlets

- **Liste blanche des cmdlets disponibles**
  - Exemple : Seul *Get-Process* a été autorisé dans cette session

```
PS C:\Users\admin> Invoke-Command -Session $s2 { Get-Item C:\ }
The term 'Get-Item' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is correct and try again.
    + CategoryInfo          : ObjectNotFound: (Get-Item:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
    + PSComputerName        : 10.55.55.229

PS C:\Users\admin> Invoke-Command -Session $s2 { [powershell]::Create().AddCommand("Get-Item").AddArgument("C:\").Invoke
() }


    Répertoire :


Mode                 LastWriteTime         Length Name                                 PSComputerName
----                 -------------         ------ ----                                 --------------
d--hs         02/10/2015     21:26                C:\                                  10.55.55.229
```

Restrictions : LanguageModes

- **Liste noire de fonctionnalités du langage**
  - Exemple : Session en mode *ConstrainedLanguage*

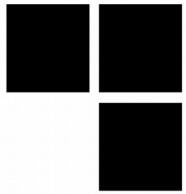
```
[10.55.55.229]: PS C:\Users\Administrator\Documents> [powershell]::Create().AddCommand("whoami").Invoke()
Cannot invoke method. Method invocation is supported only on core types in this language mode.
At line:1 char:1
+ [powershell]::Create().AddCommand("whoami").Invoke()
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : MethodInvocationNotSupportedInConstrainedLanguage

[10.55.55.229]: PS C:\Users\Administrator\Documents> powershell.exe  -Command {[powershell]::create().addcommand("whoami
").invoke()}
dom12\administrator
```

- **Solution : Contraintes AppLocker pour empêcher de relancer powershell.exe**

Restrictions : Matrice d’efficacité

	FullLanguage VisibleCmdlet	Constrained Language	Restricted Language	NoLanguage
Sans AppLocker				
Avec Applocker				Cloud Azure



Intrusion !

Intrusion : Maintenant qu'on a un shell

- **Enfin un shell qui permet de surfer facilement sur le web !**
- **Récupération de charges utiles**
  - Scripts, exécutables, documents malveillants ...
- **Exfiltrations de données**
  - Envoi de données sur un serveur web ...

Intrusion : Maintenant qu'on a un shell

- **Multiples classes / cmdlets permettant de communiquer avec les services réseau**
  - **Classe TcpClient :** Établissement de connexions TCP, scans réseau, etc

```
PS C:\Users\admin> {param($h, $p, $t) $p | % {@{socket=(New-Object System.Net.Sockets.TcpClient);port=$_}} | % {$_.Socke
t.BeginConnect($h, $_.Port, $null, $_.Socket)} | % {@{socket=$_.AsyncState;state=$_.AsyncWaitHandle.WaitOne($t);task=$_}
} | ? {$_.Socket.Connected} | % {@{ep=$_.Socket.Client.RemoteEndPoint;x=$_.Socket.Close()}}| % {$_.ep} | Select-Object A
ddress, Port}.Invoke('www.google.com', (1..1024), 1)

Address         Port
-------         ----
216.58.208.196    80
216.58.208.196   443
```

  - **Classe WebClient :** Permet d'effectuer des requêtes HTTP
  - **Cmdlet New-WebServiceProxy :** Interaction simplifiée avec les web services SOAP

- **Intégrable à des scripts d'exploitations**

- **Authentification AD transparente**

Intrusion : Contournement d'antivirus

- **Objectif : Exécuter un malware connu sans déclencher d'antivirus**

- **Proposition : L'injection réflexive en mémoire à l'aide de Powershell**
  - Non trivial mais existant sur le Web : *Invoke-Mimikatz.ps1*

    https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1

- **Problème : Ce script déclenche plusieurs antivirus**

Intrusion : Contournement d'antivirus

- **Les antivirus se déclenchent à l'écriture sur le disque ?**
  - Exécution depuis le réseau avec le cmdlet *Invoke-Expression*
  - Utilisation du cmdlet *Invoke-Command* depuis une autre machine

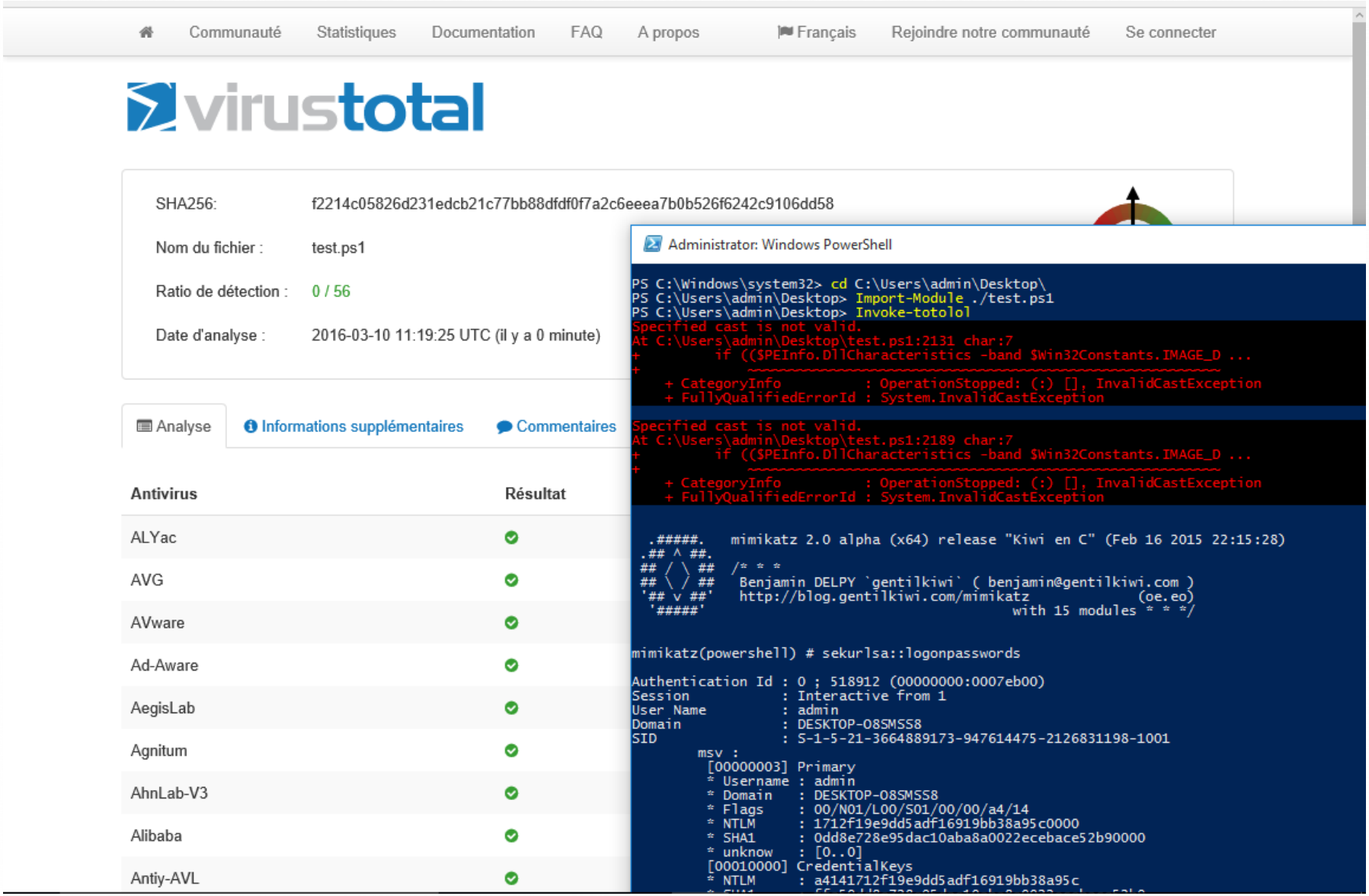
- **Les antivirus utilisent des heuristiques simples ?**
  - Chaînes statiques dans le binaire

    Il suffit de les encoder en base64
  - *ParameterSets*

    Permutation avec des entiers
  - Noms de fonction contenant Mimikatz

    Remplacement avec une chaîne pseudo-aléatoire

Intrusion : Contournement d'antivirus



Intrusion : Un client lourd .NET

- **PowerShell possède une intégration forte avec le framework .NET**
- **Possibilité d'injecter du code C# dans l'interpréteur**
  - Cmdlet *Add-Type*
  - Pas de vérification de signature de code par défaut
- **Par extension permet d'accéder aux API natives Windows avec DLLImport**
  - Déjà utilisé par certains malware

Intrusion : Un client lourd .NET

- **Possibilité de charger un assembly .NET**
  - API *[Reflection.Assembly]*
  - Cmdlet *Add-Type*
  - Pas de vérification de signature ici non plus

- **Instanciation d'objets**

  $obj = New-Object MaClasse

- **Appels de méthodes**

  $obj.Method

  [Classe]::StaticMethod

Intrusion : Client lourd .NET

- **En somme, possibilité d'instancier des objets et d'exécuter du code .NET**

- **Permet l'introspection de client lourd .NET**
  - Recherche d'URL
  - Recherche de vulnérabilité
  - Recherche de nom d'utilisateurs / mots de passe

- **Mais on peut faire mieux !**

Intrusion : Client lourd .NET

- **L’outil dnSpy**

- **Décompilateur / Debugger .NET**
  - Facilite la recherche de vulnérabilité dans un client lourd
  - Permet d’instrumenter le code
  - Analyse statique et dynamique

- **Problème : impossible de lancer un binaire dans un dossier utilisateur**
  - La politique AppLocker empêche le lancement de binaires non signés dans les répertoires sous *C:\Users*

Intrusion : Client lourd .NET

Contournement d’AppLocker

- **Mais c’est une vulnérabilité ?!!**

- **Deux points de vue :**
  - « Si tu as PowerShell, tu as PowerShell. Donc non.»
  - « Je passe outre AppLocker, je ne devrais pas. Donc oui. »

Contournement d'AppLocker

- **Mais c'est une vulnérabilité ?!!**

- **La réponse de Microsoft :**
  - « *AppLocker is not a security boundary.* »

> Hello,
>
> Thank you for contacting the Microsoft Security Response Center (MSRC). AppLocker is not a security boundary and we do not currently service issues related to it.
>
> Again, we appreciate your report.
>
> Regards,
>
> Jonathan
> MSRC

AVEZ-VOUS
DES QUESTIONS ?

MERCI DE VOTRE ATTENTION,

SYNACKTIV
DIGITAL SECURITY