



Sign in

S3cur3Th1sSh1t / Creds Public

Notifications

Fork 239

Star 1.1k

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

Creds / PowershellScripts / jaws-enum.ps1



280 lines (277 loc) · 16.6 KB

Code Blame

Raw



```
1  <#
2  .SYNOPSIS
3  Windows enumeration script
4  .DESCRIPTION
5  This script is designed to be used in a penetration test or CTF
6  enviroment. It will enumerate useful information from the host
7  for privilege escalation.
8  .EXAMPLE
9  PS > .\jaws-enum.ps1
10 will write results out to screen.
11 .EXAMPLE
12 PS > .\jaws-enum.ps1 -OutputFileName Jaws-Enum.txt
13 Writes out results to Jaws-Enum.txt in current directory.
14 .LINK
15 https://github.com/411Hall/JAWS
16 #>
17 Param(
18     [String]$OutputFilename = ""
19 )
20
21 function JAWS-ENUM {
22     write-output "`nRunning J.A.W.S. Enumeration"
23     $output = ""
24     $output = $output + "#####`r`n"
25     $output = $output + "##      J.A.W.S. (Just Another Windows Enum Script)      ##`r`n"
26     $output = $output + "##
```

```
27 $output = $output + "##          https://github.com/411Hall/JAWS          ##`r`n"
28 $output = $output + "##                                                                 ##`r`n"
29 $output = $output + "#####`r`n"
30 $output = $output + "`r`n"
31 $win_version = (Get-WmiObject -class Win32_OperatingSystem)
32 $output = $output + "Windows Version: " + (($win_version.caption -join $win_version.version) + "`r`n")
33 $output = $output + "Architecture: " + (($env:processor_architecture) + "`r`n")
34 $output = $output + "Hostname: " + (($env:ComputerName) + "`r`n")
35 $output = $output + "Current User: " + (($env:username) + "`r`n")
36 $output = $output + "Current Time\Date: " + (get-date)
37 $output = $output + "`r`n"
38 $output = $output + "`r`n"
39 write-output "          - Gathering User Information"
40 $output = $output + "-----`r`n"
41 $output = $output + "  Users`r`n"
42 $output = $output + "-----`r`n"
43 $adsis = [ADSI]"WinNT://$env:COMPUTERNAME"
44 $adsis.Children | where {$_.SchemaClassName -eq 'user'} | Foreach-Object {
45     $groups = $_.Groups() | Foreach-Object {$_.GetType().InvokeMember("Name", 'GetProperty', $r
46     $output = $output + "-----`r`n"
47     $output = $output + "Username: " + $_.Name + "`r`n"
48     $output = $output + "Groups:   " + $groups + "`r`n"
49 }
50 $output = $output + "`r`n"
51 $output = $output + "-----`r`n"
52 $output = $output + "  Network Information`r`n"
53 $output = $output + "-----`r`n"
54 $output = $output + (ipconfig | out-string)
55 $output = $output + "`r`n"
56 $output = $output + "-----`r`n"
57 $output = $output + "  Arp`r`n"
58 $output = $output + "-----`r`n"
59 $output = $output + (arp -a | out-string)
60 $output = $output + "`r`n"
61 $output = $output + "`r`n"
62 $output = $output + "-----`r`n"
63 $output = $output + "  NetStat`r`n"
64 $output = $output + "-----`r`n"
65 $output = $output + (netstat -ano | out-string)
66 $output = $output + "`r`n"
67 $output = $output + "`r`n"
68 $output = $output + "-----`r`n"
69 $output = $output + "  Firewall Status`r`n"
70 $output = $output + "-----`r`n"
71 $output = $output + "`r`n"
72 $Firewall = New-Object -com HNetCfg.FwMgr
```

```
73     $FireProfile = $Firewall.LocalPolicy.CurrentProfile
74     if ($FireProfile.FirewallEnabled -eq $False) {
75         $output = $output + ("Firewall is Disabled" + "`r`n")
76     } else {
77         $output = $output + ("Firwall is Enabled" + "`r`n")
78     }
79     $output = $output + "`r`n"
80     $output = $output + "-----`r`n"
81     $output = $output + " FireWall Rules`r`n"
82     $output = $output + "-----`r`n"
83     Function Get-FireWallRule
84     {Param ($Name, $Direction, $Enabled, $Protocol, $profile, $action, $grouping)
85     $Rules=(New-object -comObject HNetCfg.FwPolicy2).rules
86     If ($name)      {$rules= $rules | where-object {$_.name -like $name}}
87     If ($direction) {$rules= $rules | where-object {$_.direction -eq $direction}}
88     If ($Enabled)   {$rules= $rules | where-object {$_.Enabled -eq $Enabled}}
89     If ($protocol)  {$rules= $rules | where-object {$_.protocol -eq $protocol}}
90     If ($profile)   {$rules= $rules | where-object {$_.Profiles -bAND $profile}}
91     If ($Action)    {$rules= $rules | where-object {$_.Action -eq $Action}}
92     If ($Grouping)  {$rules= $rules | where-object {$_.Grouping -like $Grouping}}
93     $rules}
94     $output = $output + (Get-firewallRule -enabled $true | sort direction,applicationName,name | f
95     $output = $output + "-----`r`n"
96     $output = $output + " Hosts File Content`r`n"
97     $output = $output + "-----`r`n"
98     $output = $output + "`r`n"
99     $output = $output + ((get-content $env:windir\System32\drivers\etc\hosts | out-string) + "`r`n"
100    $output = $output + "`r`n"
101    write-output "        - Gathering Processes, Services and Scheduled Tasks"
102    $output = $output + "-----`r`n"
103    $output = $output + " Processes`r`n"
104    $output = $output + "-----`r`n"
105    $output = $output + ((Get-WmiObject win32_process | Select-Object Name,ProcessID,@{n='Owner';e
106    $output = $output + "-----`r`n"
107    $output = $output + " Scheduled Tasks`r`n"
108    $output = $output + "-----`r`n"
109    $output = $output + "Current System Time: " + (get-date)
110    $output = $output + (schtasks /query /FO CSV /v | convertfrom-csv | where { $_.TaskName -ne "Ta
111    $output = $output + "`r`n"
112    $output = $output + "-----`r`n"
113    $output = $output + " Services`r`n"
114    $output = $output + "-----`r`n"
115    $output = $output + (get-service | Select Name,DisplayName,Status | sort status | Format-Table
116    $output = $output + "`r`n"
117    write-output "        - Gathering Installed Software"
118    $output = $output + "`r`n"
```



```
207     $output = $output + " System Files with Passwords`r`n"
208     $output = $output + "-----`r`n"
209     $files = ("unattended.xml", "sysprep.xml", "autounattended.xml", "unattended.inf", "sysprep.inf"
```

```
210 $output = $output + (get-childitem C:\ -recurse -include $files -EA SilentlyContinue | Select
211 $output = $output + "`r`n"
212 $output = $output + "-----`r`n"
213 $output = $output + " AlwaysInstalledElevated Registry Key`r`n"
214 $output = $output + "-----`r`n"
215 $HKLM = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer"
216 $HKCU = "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer"
217 if (($HKLM | test-path) -eq "True")
218 {
219     if (((Get-ItemProperty -Path $HKLM -Name AlwaysInstallElevated).AlwaysInstallElevated) -eq
220     {
221         $output = $output + "AlwaysInstallElevated enabled on this host!"
222     }
223 }
224 if (($HKCU | test-path) -eq "True")
225 {
226     if (((Get-ItemProperty -Path $HKLM -Name AlwaysInstallElevated).AlwaysInstallElevated) -eq
227     {
228         $output = $output + "AlwaysInstallElevated enabled on this host!"
229     }
230 }
231 $output = $output + "`r`n"
232 $output = $output + "-----`r`n"
233 $output = $output + " Stored Credentials`r`n"
234 $output = $output + "-----`r`n"
235 $output = $output + (cmdkey /list | out-string)
236 $output = $output + "`r`n"
237 $output = $output + "-----`r`n"
238 $output = $output + " Checking for AutoAdminLogon `r`n"
239 $output = $output + "-----`r`n"
240 $Winlogon = "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
241 if (get-itemproperty -path $Winlogon -Name AutoAdminLogon -ErrorAction SilentlyContinue)
242 {
243     if ((get-itemproperty -path $Winlogon -Name AutoAdminLogon).AutoAdminLogon -eq 1)
244     {
245         $Username = (get-itemproperty -path $Winlogon -Name DefaultUserName).DefaultUsername
246         $output = $output + "The default username is $Username `r`n"
247         $Password = (get-itemproperty -path $Winlogon -Name DefaultPassword).DefaultPassword
248         $output = $output + "The default password is $Password `r`n"
249         $DefaultDomainName = (get-itemproperty -path $Winlogon -Name DefaultDomainName).Default
250         $output = $output + "The default domainname is $DefaultDomainName `r`n"
251     }
252 }
253 $output = $output + "`r`n"
254 if ($OutputFilename.length -gt 0)
255 {
```

```
256         $output | Out-File -FilePath $OutputFileName -encoding utf8
257     }
258     else
259     {
260         clear-host
261         write-output $output
262     }
263 }
264
265 if ($OutputFilename.length -gt 0)
266 {
267     Try
268     {
269         [io.file]::OpenWrite($OutputFilename).close()
270         JAWS-ENUM
271     }
272     Catch
273     {
274         Write-Warning "`nUnable to write to output file $OutputFilename, Check path and per
275     }
276 }
277 else
278 {
279     JAWS-ENUM
280 }
```