PowerShell Phishing Response Toolkit
Josh Rickard

• BS in Computer Information Systems

• Higher Education – 5 years
  • SysAdmin
  • Incident Response
  • Vulnerability Management
  • A/V Management
  • Phishing Response

• GCWN & GCFA
Part 1

• Hit with massive phishing attack(s)
• Tired of asking end-users to “forward as attachment”
• Built my own Outlook Add-In
Part 2

• Tired of manually notifying hosting providers
• Built a tool to automate it

• Solutions Engineer at PhishME
Dear User,

Due to the congestion and quota for your account, the System Administrator is increasing all the user quota limit to 200GB. **Click here** to increase quota.

Thanks,
System Administrator.

From: [Email Address]
Sent: Wednesday, May 18, 2016 4:52 AM
To: [Email Address]
Subject: Security notice

Dear Staff(s),

New security requirements were put on our email servers, due to the rate of phishing. Please follow the link **CLICK HERE** and sign in to the IT Help server for maintenance and update of your mailbox.

If your mailbox is not updated soon, Your account will be inactive and cannot send or receive messages.

On behalf of the IT department, this IT Alert Notification was brought to you by the Help Desk Department. This is a group email account so it's been monitored 24/7, therefore, please do not ignore this notification, because its very compulsory.

Sincerely,
IT Department

©2015 Microsoft outlook. All rights reserved.
Pain Points

• Users report “potential” malicious emails
• Do NOT have time to respond
• New compromised accounts

• How many of you do Incident Response?
• How many of you have time to do abuse response/analysis?
• How many of you send takedown notices for phishing emails?
Phishing Response Process

• Abuse/spam/phishing mailbox receives “potential” malicious emails
• Check and verify that the malicious link is still alive
  • If malicious email has attachment, check against VT or other hash DBs
• Go to whois.domaintools.com (or another site) and find the abuse contact
  • Sometimes it’s not listed correctly or other issues
• Send a new message/takedown notice
• Email additional lists
  • Venders, Anti-Spam appliance(s), APWG (Anti-Phishing Working Group),
• Add URLs to block lists, A/V products, etc. etc.
Assumed Access

• Incident Response/E-mail team
• Access to abuse/spam/phishing mailbox
  • Network share of messages or integrate into local Outlook client
• Windows Server or local workstation (PowerShell)
• Microsoft Outlook
• Network connection

• *PPRT v2 – SQL Server
PPRT was developed

- Repeated phishing attacks
- Multiple attack vectors
- Same or similar actors

From: ADMINISTRATOR <admin@yourcompany.com>
Date: Thursday, September 24, 2015 at 10:27 AM
To: "pricet@yourcompany.com" <pricet@yourcompany.com>
Subject: WARNING: Closing & Deleting Your Account in Progress!

Account Team

Hi pricet@yourcompany.com,

We received your request to shutdown your account.

We will process your request within 24 hours pricet@yourcompany.com.

All features associated with your account will be lost.

To retain your account, kindly Cancel Request to upgrade your account.

CANCEL & DELETE REQUEST NOW

© 2015 Inc. All Rights Reserved.
Internal “procedures” lacking

- Political battles
- Responsibility blame game
- “Procedures” vary across divisions
What I needed

• Rapid response
• Automatically send notifications
• Possibly use data for other use cases
PowerShell Phishing Response Toolkit (PPRT)

- **WHOIS** is Dead
PPRT

• Modular design
• Import .msg files for processing
• Extract attachments and check against VT
• Gather data about the .msg
• Scrape URLs from .msg
• Find abuse contact e-mail (point of contact) address
• Send e-mail notification, including original e-mail as attachment
• Send additional emails to other “block lists”
• Create (Google) maps
• Store in SQL database*
CmdLets

- Invoke-PhishingResponse
- New-MessageObject
- Extract-MessageAttachment
- Invoke-VTAttachment
- Get-URLFromMessage
- New-PPRTAbuseContactObjet
- Get-AbsoluteUri
- Get-IPAddr
- Get-WhichWhois

- Check-ARIN
- Check-RIPE
- Check-APNIC
- Check-LACNIC
- New-FirstReceivedFromIPObject
- New-AllReceivedFromIPObject
- Create-FirstReceivedFromIPMap
- Create-AllReceivedFromIPMap
- Create-FirstReceivedFromIPHeatMap
New-MessageObject

- $Message[]
- [switch] (Full ParameterSet)
  - FullDetails
- [switch (Partial ParameterSet)]
  - FullName
  - Subject
  - Body
  - HTMLBody
  - BCC
  - CC
  - ......
  - Headers

Returns PPRT.Message
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export-MessageAttachment</td>
<td>Export-MessageAttachment</td>
</tr>
<tr>
<td>Get-AbsoluteUri</td>
<td>Get-AbsoluteUri</td>
</tr>
<tr>
<td>Get-IPAddress</td>
<td>Get-IPAddress</td>
</tr>
<tr>
<td>Get-PhishingGeoLocation</td>
<td>Get-PhishingGeoLocation</td>
</tr>
<tr>
<td>Get-URLFromMessage</td>
<td>Get-URLFromMessage</td>
</tr>
<tr>
<td>Invoke-PhishingResponse</td>
<td>Invoke-PhishingResponse</td>
</tr>
<tr>
<td>Invoke-VIAttachment</td>
<td>Invoke-VIAttachment</td>
</tr>
<tr>
<td>New-AllReceivedFromIPObject</td>
<td>New-AllReceivedFromIPObject</td>
</tr>
<tr>
<td>New-FirstReceivedFromIPObject</td>
<td>New-FirstReceivedFromIPObject</td>
</tr>
<tr>
<td>New-MessageObject</td>
<td>New-MessageObject</td>
</tr>
<tr>
<td>New-PPRTAbuseContactObject</td>
<td>New-PPRTAbuseContactObject</td>
</tr>
<tr>
<td>New-PPRTNotificationObject</td>
<td>New-PPRTNotificationObject</td>
</tr>
<tr>
<td>Send-PhishingNotifications</td>
<td>Send-PhishingNotifications</td>
</tr>
<tr>
<td>Send-PPRTNotification</td>
<td>Send-PPRTNotification</td>
</tr>
</tbody>
</table>
[BEWARE-SUSPECTED SPAM!]INFO

Mon 6/1/2015 1:23 PM

Conferring L Jones <conferringl@verizon.net>

To  Info@verification.com

CLICK THIS LINK BELOW TO AVOID DEACTIVATION OF YOUR [REDACTED] ACCOUNT

Click Here

$MsgObj = @()  
$MsgObj = New-MessageObject -message C:\PHISHING_EMAILS\DUMMY_EMAILS\*.msg -logpath C:\PHISHING_EMAILS -FullDetails

$Response = @()  
$Response = Invoke-PhishingResponse -message $MsgObj -logpath C:\PHISHING_EMAILS -from josh.rickard@phishme.com -smtpserver smtp.office365.com
Invoke-PhishingResponse

- [New-MessageObject]
  - MessageObject
- [VT]
  - ExtractAttachment
  - SaveLocation
  - VTAPIKey
- [Email]
  - From
  - Subject
  - Body
  - BodyAsHtml
  - Credential
  - AdditionalNotification

- [Map1]
  - AllReceivedFromIPMap
- [Map2]
  - FirstReceivedFromIPMap
  - FirstReceivedFromIPHeatMap

Returns PPRT.PhishingResponse
Extract-MessageAttachment

- [New-MessageObject]
  - MessageObject
- SavePath
- [Full]
  - FullDetails
    - ->

- [Partial]
  - GetFileHash
  - DisplayName
  - FileName
  - Index
  - Position
  - Type
  - Size
  - MIMEType
  - AttachedMethod
  - AttachContentID

Returns PPRT.Attachment
Invoke-VTAttachment

- [PPRT.Attachment]
  - AttachmentHash
- [VT]
  - VTAPIKey
- Uses Posh-VirusTotal

Returns PPRT.VTResults
Begin
{
$ReturnType = @()

#download and import Posh-VirusTotal
if (!((Test-Path "$Home\Documents\WindowsPowerShell\Modules\Posh-VirusTotal")))
{
    if ($result -eq 'Yes')
    {
    }
    else
    {
        exit
    }
}
Process
{
    foreach ($hash in $AttachmentHash.Hash)
    {
        $VTFileReport = @()
        $VTFileReport = Get-VTFileReport -Resource $hash -APIKey $VTAPIKey
        if ($VTFileReport.ResponseCode -eq 1)
        {
            $result = [System.Windows.Forms.MessageBox]:Show("The following SHA256 hash was already been submitted to VirusTotal.\n't $hash", 'Warn', [System.Windows.Forms.MessageBoxButtons]::None, [System.Windows.Forms.MessageBoxIcon]::None, 'VirusTotal Submission' -Folder $logpath -CustomMessage "Hash has been previously submitted to VirusTotal"
            $props = [
                AttachmentHash = $hash
                SubmissionStatus = $SubmissionStatus
                VTFileReport = $VTFileReport
                VTSubmissionResult = $null
            ]
        }
    }
}
$ReturnObject = New-Object -TypeName PSObject -Property $props

}  # End if ($Result -eq $True)

$SubmissionStatus = 'Hash not Found or Submitted'

$props = {
    AttachmentHash = $Hash
    SubmissionStatus = $SubmissionStatus
    VTFileReport = $VTFileReport
    VTSubmissionResult = $SubmitToVT
}

$ReturnObject = New-Object -TypeName PSObject -Property $props

}  # End if ($Result -eq $True)

# return $ReturnObject
Get-URLFromMessage

• [PPRT.Message]
  • MessageObject

Returns PPRT.PhishingURL
New-PPRTAbuseContactObject

- [PPRT.PhishingURL]
  - URLObject

- Checks if URL is alive
- Gets the AbsoluteUri
- Get-IPAddress
- Get-WhichWhois
- Check-
  - ARIN
  - RIPE
  - AFRNIC
  - APNIC
  - LACNIC

Returns PPRT.AbuseContact
function Check-RIPE ()
{
    [CmdletBinding()]
    param (  
        [parameter(Mandatory = $true, Position = 1, HelpMessage = 'Please provide a IP address')] $ipaddress
    )

    <#
    .SYNOPSIS
    Takes IP address as input and queries RIPE's WHOIS implementation for the IP address
    
    .DESCRIPTION
    Takes a IP Address and searches for RIPE's WHOIS abuse contact email for that IP address
    Returns this contact email address
    
    .PARAMETER ipaddress
    Specifies the specific IP address belonging to RIPE
    
    .EXAMPLE
    C:\PS> Check-RIPE -ipaddress '195.42.65.82'
    #>
    
    $abusecontact = Invoke-RestMethod -Uri "http://rest.db.ripe.net/abuse-contact/$ipaddress"
    $result = $abusecontact.'abuse-resources'.'abuse-contacts'.email
    return $result
}

$rawdata = Invoke-RestMethod -URI "http://rdap.arin.net/bootstrap/ip/$ipaddress"

HANDLE = $rawdata.entities.Handle

HANDLEData = Invoke-RestMethod -URI "https://rdap.arin.net/registry/entity/$HANDLE"

$VcardData = @{$($HANDLEData.entities.vcardArray) | select -ExpandProperty SyncRoot}

$VcardEmailAddress = $VcardData | Select-String -Pattern $regex

$VcardMatches = $VcardEmailAddress.Matches.Value

return $VcardMatches

for($i = 0; $i -lt ($rawdata.entities.vcardArray).count; $i++)
{
    foreach ($item in $rawdata.entities.vcardArray.SyncRoot[$i])
    {
        [array]$result += $item
    }
}

$parsedResult = $result | Select-String -Pattern $regex

if ($parsedResult.count -gt 0)
{
    return $parsedResult
}
return 'NO POC FOR ARIN'
Get-AbsoluteUri

• [PPRT.PhishingURL]
  • URLObject

• Note:
  • List of 342 short URLs used for filtering

• Checks if URL is alive
• Gets the ResponseUri
• Recursively expand shortened URLs/links

• Returns
  • OriginalURL
  • EncodedURL
  • AbsoluteURL
  • URLAuthority

Returns PPRT.Uri
We have received a phishing attempt (attached) that is using an IP registered to this contact. Please remove this site as soon as you can: http://emailverificationform.weebly.com/.

In addition, any logs you can provide surrounding the registration or usage of this site would help us understand who is targeting our environment.

Thank you!
We have received a phishing attempt (attached) that is using an IP registered to this contact. Please remove this site as soon as you can: http://emailverificationform.weebly.com/.

In addition, any logs you can provide surrounding the registration or usage of this site would help us understand who is targeting our environment.

Thank you!
The attached email is a phishing email: http://emailverificationform.weebly.com/
Josh Rickard

Phishing Links 20160522

To: 'anti-phishing-email-reply-discuss@googlegroups.com'

emailverificationform.weebly.com,20160522
PPRT – Geo-Location Mapping

• Import .MSG (or *.MSG) to Invoke-PhishingResponse
• Switches = different maps
  • MapAllIPs
  • HeatMap
  • MapStartingIP
• Exports PPRT Map to .html file
Gathering Email Data from 35 email messages
Processing RE-VALIDATE YOUR MAILBOX

X-Exchange-Antispam-Report-Test: UriScan:
X-Exchange-Antispam-Report-CFA-Test: BCL:0;PCL:0;RULEID:(8121501046)(3002001);SRVR:BLUPRO1MB469;BCL:0;PCL:0;RULEID:;SRVR:BLUPRO1MB469;
X-MS-Exchange-Organization-SCL: 1
X-Microsoft-Exchange-Diagnostics: 1;BLUPRO1MB469;23:uBN\kWmnQ5mBw3ooJPRgXFADDec4JErX2A/oePvHjMQGJGkkvBzl4Tw39Ldkty1/IwIOOnwkpP25w83k2Zw2ErFqCbhzzLAPrInPxbCq50c49V7yZEPVr8yXDcdHvbflecY1C3dxFNSLm1WCCLwV5sUnTPxmo6Y2UvCepINoaqgL6C3D6wcJd2mmYAnQIQY8TrqNLliJUGYdx19dgFpMPgyoJNBxndeEXxKEbKot1IoV7Q3emm1s0WmQ9Gg+bxqW360W4iVtdhj2N7Buw==;5:0dNk40qzbbnLsLBYD9Q4IX09od74/P08Fkj92icgjvnWdeXjKRO53C87YezwK/_AOz0yru.Lstezk14oohChSt7ErE3LlUqY8jcuK9Wn17zy5boAL+R0yFqtw5xrU+vyy0kdVCsDKHO/RycUuw==;24:fgNbwKcrqLgy/xsCDqWnN2Myvq1BuCv+aLgPuaPNKc1BTv5r5+LKeo2571ckpm7VHtal750AryguudvXFr/Rtm9y7AD6DKXX+C2xSpAltJs==;
20:2bx1LGReXfadiwrAI7xzkjWHkg9k1GDMA7zJnatDjdf7PgpG7azv9hK187QOA1080r0w1H2Bnoq9gt60FvFqWw==
SpamDiagnosticOutput: 1:23
SpamDiagnosticMetaData: NSPM
X-MS-Exchange-CrossTenant-OriginalArrivalTime: 01 Sep 2015 18:24:59.4755
(UTC)
X-MS-Exchange-CrossTenant-Id: e3fe5fde-e-f7e9-401b-a51a-355e01b05a89
X-MS-Exchange-CrossTenant-FromEntityHeader: Internet
X-MS-Exchange-Transport-CrossTenantHeadersStamped: BLUPRO1MB469
X-MS-Exchange-Organization-AuthSource: BN1ASS011FD053.protection.gbl
X-MS-Exchange-Organization-AuthAs: Anonymous
X-MS-Exchange-Transport-EndToEndLatency: 00:00:01.0242515
MIME-Version: 1.0
MapAllIPs
foreach ($item in $MessageObject)
{
    $ReceivedFromIP = (Parse-EmailHeader -InputFileName $item.Headers).From | Select-String -Pattern $regex -AllMatches | ForEach-Object -Process { $_.Matches } | ForEach-Object -Process { $_.Value }

    foreach ($ip in $ReceivedFromIP)
    {
        $ipLocation = ''
        $ipLocation = Invoke-RestMethod -Uri "http://freegeoip.net/xml/$($ip)"

        if (((($ipLocation.Response.Latitude -ne 0) -or ($ipLocation.Response.Longitude -ne 0)))
        {
            if (![[string]]::IsNullOrWhiteSpace($ipLocation.Response.Latitude))
            {
                if (![[string]]::IsNullOrWhiteSpace($ipLocation.Response.Longitude))
                {
                    $originalPolyline = "lat: $($ipLocation.Response.Latitude), lng: $($ipLocation.Response.Longitude)"
                    $polyline += $originalPolyline
                }
            }
        }
    }

    $props = @{
        marker = "[($polyline -join ',')]"
        subject = $item.Subject
        SentFromAddress = $item.SenderEmailAddress
        SentFromType = $item.SenderEmailAddress
        ReceivedTime = $item.ReceivedTime
        EmailBody = $item.Body
    }

    $tempAllIPObject = New-Object -TypeName PSObject -Property $props
    $AllIPObject += $tempAllIPObject
MapStartingIP
$firstReceivedFromIP = (Parse-EmailHeader -InputFileName $Item.Header).From | 
  Select-String -Pattern $regex -AllMatches | 
  ForEach-Object -Process { $_.Matches } | 
  ForEach-Object -Process { $_.Value }

# calling first received from header returned from parse-emailheader. Location is [0]
$soriginalIpAddress = Invoke-RestMethod -Uri "http://freegeoiip.net/xml/$($firstReceivedFromIP[0])"

$tempStartingIPObject = @()

# getting all first received from IP from headers and creating markers
if (((soriginalIpAddress.Response.Latitude -ne 0) -or (soriginalIpAddress.Response.Longitude -ne 0))
{
  if (![string]:.IsNullOrWhiteSpace(soriginalIpAddress.Response.Latitude))
  {
    if (![string]:.IsNullOrWhiteSpace(soriginalIpAddress.Response.Longitude))
    {
      # adding json markup data to object. This will be passed to Get-PhishingGeoLocationStartingIps cmdlet
      $props = @{
        marker
          = "{"title": "$($Item.subject -replace "", ")', 'lat': "$($originalIpAddress.Response.Latitude)'}, 'lng': "$($originalIpAddress.Response.Longitude)'}, 'description': "<div><div></div><h1>$($Item.Subject -replace "", ")</h1><div><div><b>Subject</b>: $($Item.Subject -replace "", ")</div></div><p><b>Received Time</b>: $($Item.ReceivedTime)</p><b>Sender Email Address</b>: $($Item.SenderEmailAddress)<p><b>Sender Email Type</b>: $($Item.SenderEmailType)</p><b>Phishing URL</b>: $($Item.URL.RawPhishingLink)</b><div></div>"

subject = $msg.Subject
SentFromAddress = $msg.SenderEmailAddress
SentFromType = $msg.SenderEmailType
ReceivedTime = $msg.ReceivedTime
EmailBody = $msg.Body

$tempStartingIPObject = New-Object -TypeName PSObject -Property $props
HeatMap
Resources

• Posh-VirusTotal - https://gist.githubusercontent.com/darkoperator/9138373/raw/22fb97c07a21139a398c2a3d6ca7e3e710e476bc/PoshVTInstall.ps1
• ARIN: http://rdap.arin.net/bootstrap/ip/$ipaddress
• RIPE: http://rest.db.ripe.net/abuse-contact/$ipaddress
• LACNIC: http://rdap.lacnic.net/rdap/ip/$ipaddress
• APNIC: http://rdap.apnic.net/ip/$ipaddress“
• FreeGeolp - "http://freegeoip.net/xml/$(ip)"
Why use this?

• Depends on your internal process
• Great for open networks, like Higher Education and organizations with disbursed employee’s
Why do you care?

• Some organizations can not afford the latest/greatest appliance
• The sooner you shut down phishing websites the better
• Less chance of users falling for phishing scams
What do we gain?

• DATA!
  • IP addresses
  • Time stamps
  • E-mail subjects
  • Location
  • Visual maps for management
  • VirusTotal Attachment Results
  • Hashes
What else?

- Automate phishing email response received to abuse/phishing/spam inboxes
- Forward phishing URLs to firewalls/blacklists/etc.
- Add them to individual workstations HOST files
- Sinkhole DNS
PPRT

This PowerShell Module is designed to send notifications to hosting companies that host phishing URLs by utilizing the major WHOIS/RDAP Abuse Point of Contact (POC) information.

1. This function takes in a .msg file and strips links from a phishing URL.
2. After getting the phishing email, it is then converted to its IP Address.
3. Once the IP Address of the hosting website is identified, then we check which WHOIS/RDAP to search.
4. Each major WHOIS/RDAP is represented: ARIN, APNIC, AFRNIC, LACNIC, & RIPE.
5. We call the specific WHOIS/RDAP’s API to determine the Abuse POC.
Contact

• https://github.com/MSAdministrator/PPRT---PowerShell-Phishing-Response-Toolkit

• @MS_dministrator