Demonstrating bulk reverse passive DNS lookup with PowerShell for IT security investigations: the case of the Phorphiex botnet

Posted on August 25, 2021

IP addresses are straightforward input data for IT security investigations: they are technically necessary for nodes of the Internet to communicate. Hence, if they are not deleted in some tricky way after cybercrime has been committed, or they are to be found in any of the logs before the commitment, they help a lot to unfold what has actually happened.

IBM Xforce exchange is a forum reporting many security incidents that are relevant for those who are in charge of maintaining IT security. In the present blog we shall pick one of their reports and check how we can extend the given information with WhoisXML APIs using PowerShell which comes installed on Windows and can be used on Linux and Mac OS X, too. We assume lower intermediate skills of PowerShell programming to follow the description below.

We will be using passive DNS data, as provided by WhoisXML API's Reverse DNS API: we can get a list of network communication events observed by passive DNS sensors in which a given IP resolves to an actual domain name. This means that the given IP at the given time was communicating on the Internet under the given domain name; an information not obtainable from the Domain Name System itself. Meanwhile it helps a lot to reveal the character of the malicious opponent, as it will be clear from the following demonstrative research.

1. Getting data from an IBM XForce collection

Our subject will be the recently observed activity of the Phorphiex botnet. Quoting IBM's description (at https://exchange.xforce.ibmcloud.com/collection/Phorpiex-Botnet-Extortion-Activity-Monitoring-76265914d081e79d158260bf5385a9da

, visited on 19 August 2021),

“IBM X-Force ongoing collection was created to provide you with the latest IoCs and intel on the Phorpiex Botnet Extortion activity found on our environment. The collection will be updated automatically whenever new findings have been determined. Please note that the IoCs within this collection are actionable and hence can be used for blockings to protect your environment. Due to this, it may happen that specific IoCs get removed from the collection in a dynamic time range when no further evidence is available to support the maliciousness of the indicator.”

“The Phorpiex botnet was initially known as a botnet using the IRC protocol for its operations before switching to a modular architecture. The distribution takes place by exploit kits and through support from other malware families. With this success its operators leverage now operations as Malware-as-a-Service (MaaS). This offering allows other cyber gangs to use the botnet infrastructure for their malicious intentions. However, a big part of its daily threats are the so called "sextortion" spam campaigns. Sextortion in spam campaigns refers to the coercion of victims to pay a certain amount of money within a given time frame under threat of publishing intimate pictures or videos of the victim. This is known to be a fake threat since there is no known case so far where the threat actors proved to possess such material.”

IBM X-Force Exchange provides a collection of reports containing IP addresses from which the activity of this botnet has been recently observed. At the time of writing the present blog, the list of the IP addresses can be deduced from information downloaded from IBM X-Force exchange. Namely, a file can be downloaded from the aforementioned link by choosing the "Export" option from the menu next to the "I'm affected" button. Choosing the STIX 2.0 format will result in a downloaded file which is in a standard JSON format from which the IP addresses can be deduced and processed further.

2. Distilling IP addresses from a STIX 2.0 format report

Our downloaded STIX 2.0 file has been saved under the name xfe-collection_76265914d081e79d158260bf5385a9da.json . Then get the list of IPs with PowerShell this way, in the directory where the downloaded file resides, we do the following conversion:

```
$data=Get-Content ".\xfe-collection_76265914d081e79d158260bf5385a9da.json" |
    ConvertFrom-Json
$data.objects.pattern |
    foreach {$_ | Select-String -Pattern '(\d{1,3}\.){3}\d{1,3}'} |
    foreach {$_.Matches.Value} > ips.csv
```

The first command converts the entries in the collection into a PowerShell object. The second line gets the pattern field of the JSON which contains also the IP address in a string. Then we get the IPv4 address with regular expression matching. The resulting ips.csv file has one IP address each line. In the next step we will develop a script that supplements this file with passive DNS information using the Reverse IP API.

3. A script to extend the list of IPs with reverse passive DNS data

Our goal of extending a list of IP addresses with reverse passive DNS data can be accomplished using the PowerShell cmdlet with the following code (recommended to save e.g. under the name ExtendIPCsvWithReversePDNS.ps1):

```
#Extends a list of IPs with Reverse IP/DNS lookup results
# The input list does not contain a header and contains valid IPs
#Usage:
# 1. set the environment variable APIKEY to your API key:
#    PS C:\Users\User\WorkingDirectory> $APIKey="YOUR_API_KEY"
# 2. Once done, you can run it like this:
#   PS C:\Users\User\WorkingDirectory> .\ExtendIPCsvWithReversePDNS.ps1
'.\inputfile.csv' '.\outputfile.csv'
# Note: if the output file exists and is not empty, the results will be appended.
```

```
[CmdletBinding()]
param(
    [string] $InputFile,
    [string] $OutputFile
)

$BaseUrl = "https://reverse-ip.whoisxmlapi.com/api/v1?apiKey=" + $ENV:APIKEY +"&ip="

Function Convert-FromUnixDate ($UnixDate) {

[timezone]::CurrentTimeZone.ToUniversalTime(([datetime]'1/1/1970').AddSeconds($UnixDate))
}

Import-Csv -Header "IP" -Path $InputFile | ForEach-Object {
Write-Host $_.IP
$URI = $BaseUrl + $_.IP
#Need this to be visible in the catch branch
$IP = $_.IP
$IPData = [PSCustomObject]@{
IP = $IP
}
try{
$APIResponse = Invoke-WebRequest -Uri $URI -UseBasicParsing
$k=0
$Result = ConvertFrom-Json $APIResponse.Content
foreach($row in $Result.result) {
$first_seen = Convert-FromUnixDate $row.first_seen
$last_visit = Convert-FromUnixDate $row.last_visit
Write-Host $k $row.name $first_seen $last_visit
$field1 = "name_" + [string]$k
$field2 = "first_seen_" + [string]$k
$field3 = "last_visit_" + [string]$k
$IPData | Add-Member -MemberType NoteProperty -Name $field1 -Value $row.name
```

```
$IPData | Add-Member -MemberType NoteProperty -Name $field2 -Value $first_seen
$IPData | Add-Member -MemberType NoteProperty -Name $field3 -Value $last_visit
$k += 1}
for($l=$k; $l -lt 300; $l++){
$field1 = "name_" + [string]$l
$field2 = "first_seen_" + [string]$l
$field3 = "last_visit_" + [string]$l
$IPData |
    Add-Member -MemberType NoteProperty -Name $field1 -Value " "
$IPData |
    Add-Member -MemberType NoteProperty -Name $field2 -Value " "
$IPData |
    Add-Member -MemberType NoteProperty -Name $field3 -Value " "
}
}
catch{
Write-Host "Ran into an issue: $($PSItem.ToString())"
for($l=0; $l -lt 300; $l++){
$field1 = "name_" + [string]$l
$field2 = "first_seen_" + [string]$l
$field3 = "last_visit_" + [string]$l
$IPData |
    Add-Member -MemberType NoteProperty -Name $field1 -Value " "
$IPData |
    Add-Member -MemberType NoteProperty -Name $field2 -Value " "
$IPData |
    Add-Member -MemberType NoteProperty -Name $field3 -Value " "
}
}
$IPData |
    Export-Csv -Append -NoTypeInformation -Encoding UTF8 $OutputFile
}
```

The script contains usage instructions in the first few comment lines to be self-consistent; we shall

describe the way of invoking it later. As of its operation, we define two positional arguments: the input file which is a list of IPs one per line, and the output file the result will be appended to. We store the base URL of the API in $BaseUrl. The function Convert-FromUnicDate serves the goal of converting times returned as Epoch by the API to datetimes.

The main loop gets the IPs through a pipeline. For each IP we will store the record in $IPData. The API call and its processing is in a try-catch to handle the case when something goes wrong. We invoke the API with Invoke-WebRequest and parse the resulting JSON with ConvertFrom-Json. We want each IP to be a row in the output csv file, so we loop through the result field of the API result which is a list of results and map each record's name, first_seen, and last_visit to a next field with an ordinal number. As Export-Csv cannot deal with a variable number of feeds per row at the moment, we fill the remaining fields with a value of a single space as a placeholder. We do the same in the catch branch to have an empty line if something went wrong, after printing an error message to the console. (Note that the API returns up to 300 records in a single call; if the given IP has more, it will provide us with 300 arbitrary records when called as above, hence we generate lines with 300 result triplets here. We refer to the API documentation for a description of how to get all the records.) Finally the resulting IPData object is appended to the output csv file.

To use the script the environment variable $APIKey is to be set to your actual API key, i.e.

```
$APIKey="YOUR_API_KEY"
```

(The string YOUR_API_KEY above is to be substituted with your API key which is available after registering at https://reverse-ip.whoisxmlapi.com/api/signup , or your account page after registration.) Then we can run the script just as described in the comment at the beginning. Using it for the file ips.csv we prepared previously,

```
PS C:\Users\User\WorkingDirectory> .\ExtendIPCsvWithReversePDNS.ps1 '.\ips.csv' '.\ips_result.csv'
```

results in the file ips_result.csv that can be imported to an office spreadsheet or viewed as text. It contains a header line and a line for each IP like this:

"91.232.140.99","91.232.140.99.ip.rudna-net.pl","1/4/2019 8:32:10 PM","7/16/2021 8:09:19 AM"," "," "

We have omitted the lines as they are very long.

4. Results and conclusion

We do not publish the data file here as it contains malicious IPs and it is rather large. We summarize our findings instead. Looking at the results in detail it is apparent that many of these IPs were not present in the passive DNS database. A direct DNS lookup has confirmed that most of these indeed do not have a reverse DNS record. Even those IPs which have reverse records typically resolve to automatically generated names given by cable or mobile network providers to dynamic IPs.

From this, one can conclude that the bot activity is, according to the presented reverse passive DNS analysis of the IP address collection, based on machines of individual subscribers at various providers.

As the passive DNS data contain datetimes, and the initial STIX 2.0 file contains dates, too, correlating the dates, the domain names which were really affected could be found. Based on the results, a warning could be sent to the impacted providers, who could then analyze their logs and inform their clients who became a part of this botnet typically as a consequence of the activity of some exploit kit they were targeted by.

As a summary, we have described also in technical details how one can extend a list of IPs with supplemental information using the Reverse IP/DNS API. Our demonstration has used real-life data and resulted in consequences that can be useful directly. Visit https://www.whoisxmlapi.com/ for an API key to reproduce the above results yourself or do similar analyses. It is also worth looking at WhoisXML API's other powerful cybersecurity tools.