A Deep Dive into Zebrocy's Dropper Docs - SentinelLabs

By Marco Figueroa

Published: 2021-04-19 · Archived: 2026-04-05 13:53:26 UTC

Contributor: Amitai Ben Shushan Ehrlich

Sofacy is an APT threat actor that's been around since 2008 and rose to prominence with the election hacks of 2016. Better known as FancyBear or APT28, this threat actor targets governments, military, and private organizations and has been known to engage in hack-and-leak operations. In the past couple of years, Sofacy has drastically retooled and largely evaded analysts. One of the more consistent subgroups is known as Zebrocy. Their targeting appears primarily focused on former Soviet Republics and, more recently, Asia.

In March 2021, we observed a cluster of activities targeting Kazakhstan with Delphocy – malware written in Delphi and previously associated with Zebrocy. The Word documents that were observed purport to be from a Kazakhy company named Kazchrome, a mining and metal company and one of the world's largest producers of chrome ore and ferroalloys.

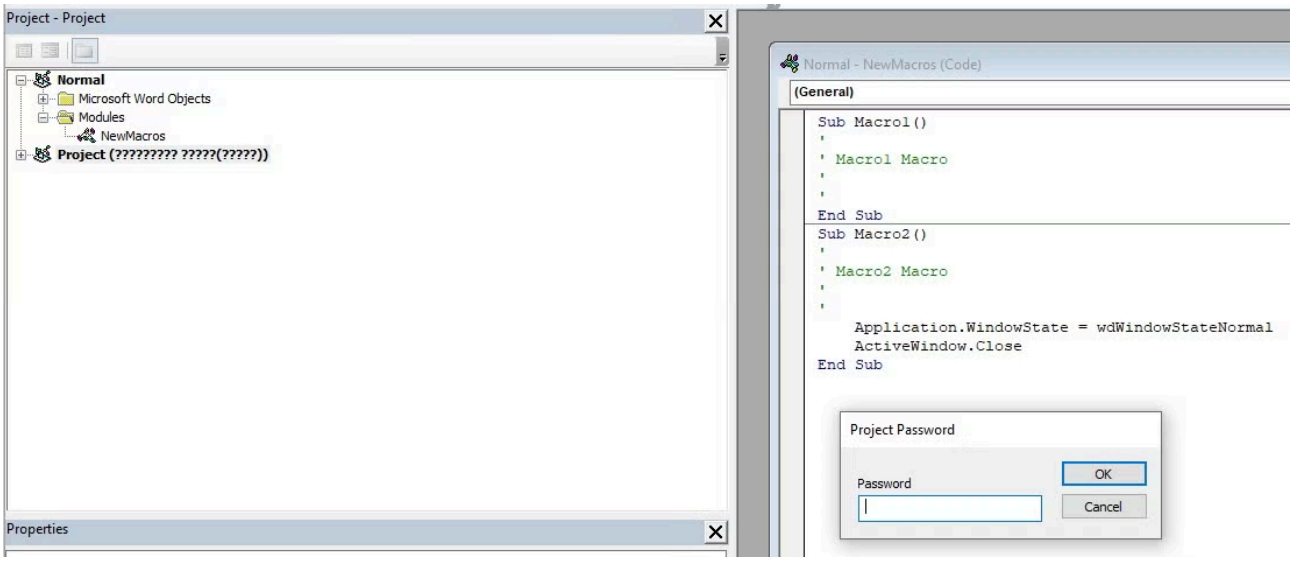
In total, we found six Delphocy Word documents that appear to be related to this cluster, all of which contain the same VBA script that drops a PE. Out of the six Word documents, two appear to be authentic uploads to VirusTotal by victims originating from Kazakhstan. The uploaded files contain what appeared to be the original filenames Авансовый отчет(новый).doc and Форма докладной (служебной) записки.doc.

In this post, we take a deep dive into these samples and share some techniques other analysts can employ to reverse engineer Delphocy dropper docs. We show how researchers can bypass password-protected macros and describe both how to decompile Delphi using IDR (Interactive Delphi Reconstructor) and how to import the saved IDC file into Ghidra using dhrake's plugin.

The results of our analysis led us to discover further Zebrocy clusters; a list of IOCs and YARA detection rules are provided to enable threat hunters to search for these and related artifacts in their environments.

Bypassing VBA Macro Password Protection

When analyzing Office documents with VBA macros, threat hunters have many different tools and techniques that do the job, but I've built a habit that I still use when I first started reversing malware to bypass password-protected macros manually.



1. Open up your favorite hex editor. I use HxD.
2. Load the Word Document.
3. Search for the following text:
    1. `CMG=`
    2. `GC=`
    3. `DPB=`
4. Add an `x` to each of them:
    1. `CMGx=`
    2. `GCx=`
    3. `DPBx=`
5. Save the file with the changes.

HxD - [C:\Users\Marco\Desktop\Авансовый отчет(новый).doc]

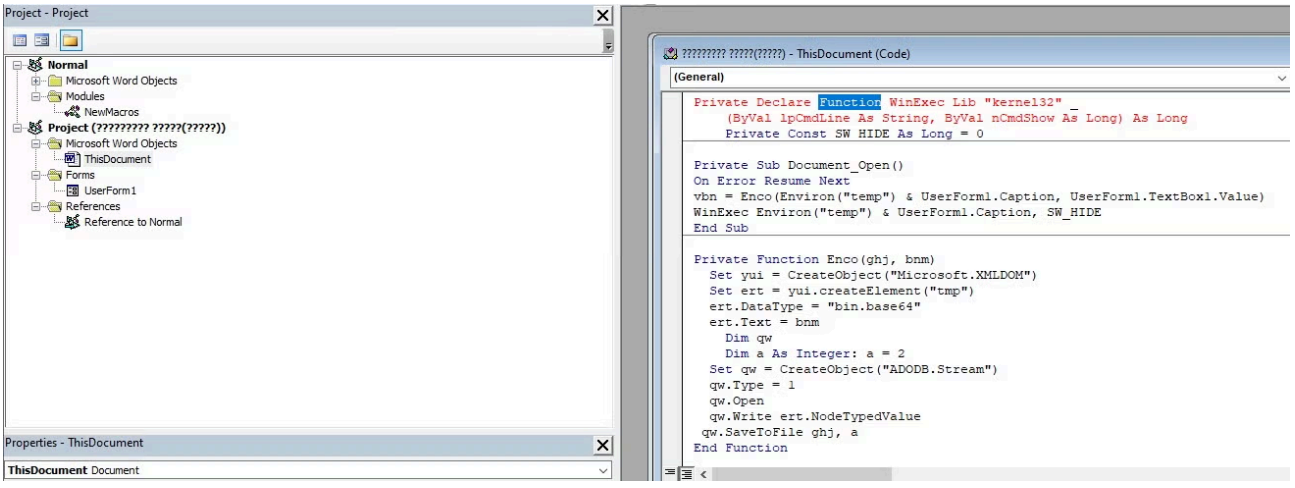
File Edit Search View Analysis Tools Window Help

16 | Windows (ANSI) | hex

Авансовый отчет(новый).doc

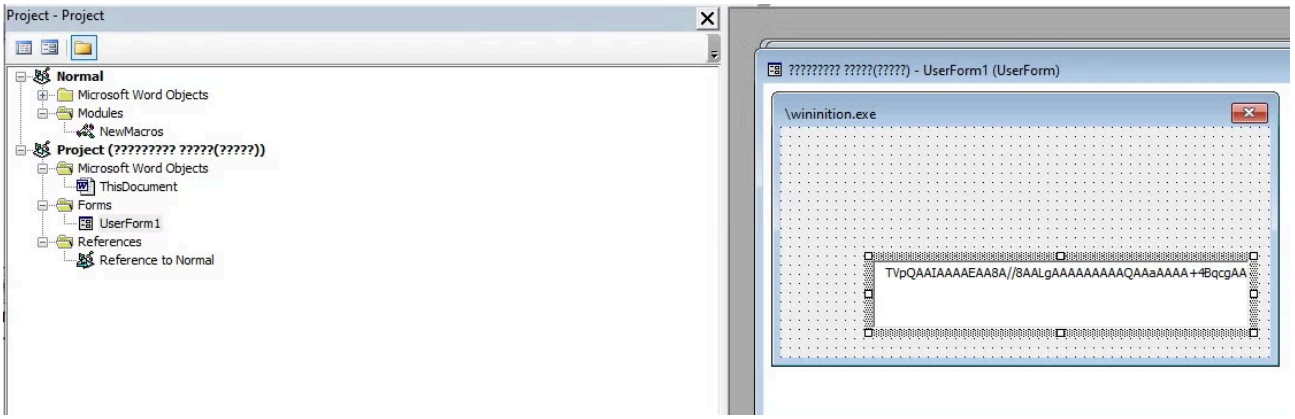
Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
004B8110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
004B8120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
004B8130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
004B8140	49	44	3D	22	7B	30	30	30	30	30	30	30	30	2D	30	30	ID="{00000000-00
004B8150	30	30	2D	30	30	30	30	2D	30	30	30	30	2D	30	30	30	00-0000-0000-000
004B8160	30	30	30	30	30	30	30	30	30	7D	22	0D	0A	44	6F	63	000000000}"..Doc
004B8170	75	6D	65	6E	74	3D	54	68	69	73	44	6F	63	75	6D	65	ument=ThisDocume
004B8180	6E	74	2F	26	48	30	30	30	30	30	30	30	30	0D	0A	42	nt/&H00000000..B
004B8190	61	73	65	43	6C	61	73	73	3D	55	73	65	72	46	6F	72	aseClass=UserFor
004B81A0	6D	31	0D	0A	48	65	6C	70	46	69	6C	65	3D	22	22	0D	m1..HelpFile="".
004B81B0	0A	4E	61	6D	65	3D	22	50	72	6F	6A	65	63	74	22	0D	.Name="Project".
004B81C0	0A	48	65	6C	70	43	6F	6E	74	65	78	74	49	44	3D	22	.HelpContextID="
004B81D0	30	22	0D	0A	56	65	72	73	69	6F	6E	43	6F	6D	70	61	0"..VersionCompa
004B81E0	74	69	62	6C	65	33	32	3D	22	33	39	33	32	32	32	30	tible32="3932220
004B81F0	30	30	22	0D	43	4D	47	78	3D	22	31	35	31	37	42	39	00".CMGx="1517B9
004B8200	35	42	43	39	46	37	43	44	46	37	43	44	46	33	44	31	5BC9F7CDF7CDF3D1
004B8210	46	33	44	31	22	0D	0A	44	50	42	3D	22	41	44	41	46	F3D1"..DPB="ADAF
004B8220	30	31	43	33	30	31	34	36	31	45	34	36	31	45	42	39	01C301461E461EB9
004B8230	45	32	34	37	31	45	36	31	36	46	30	31	44	30	36	30	E2471E616F01D060
004B8240	39	33	43	35	39	41	37	43	34	44	33	30	46	36	34	41	93C59A7C4D30F64A
004B8250	35	31	42	44	45	44	44	41	39	38	45	43	31	35	39	30	51BDEDDA98EC1590
004B8260	43	39	42	31	39	31	46	46	22	0D	47	43	78	3D	22	34	C9B191FF".GCx="4
004B8270	35	34	37	45	39	36	42	31	39	30	32	31	41	30	32	31	547E96B19021A021
004B8280	41	30	32	22	0D	0A	0D	0A	5B	48	6F	73	74	20	45	78	A02"....[Host Ex

When opening the Word document and viewing the macro this time, you can see the script as well as the Forms. When analyzing the function, what immediately sticks out is the `ert.DataType = “bin.base64”`, showing that the UserForm1 is encoded with base64.

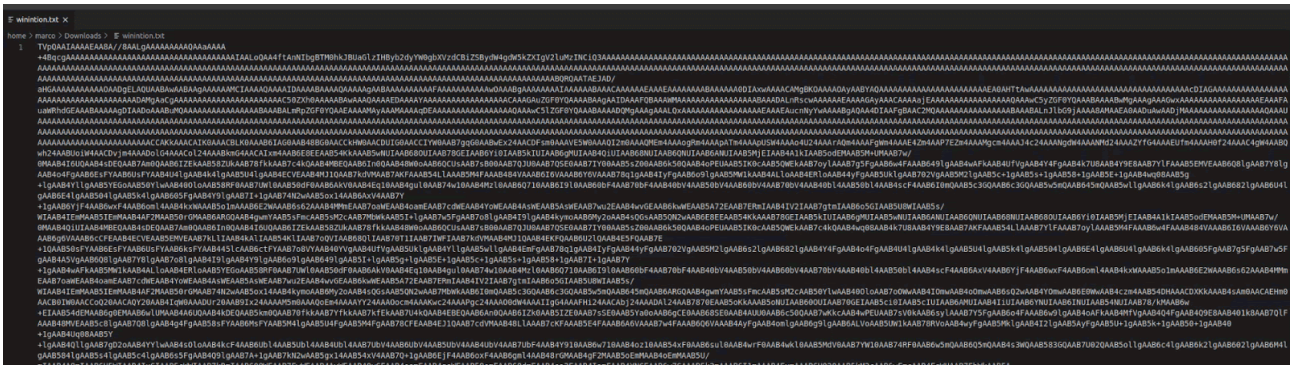


Wininition UserForm

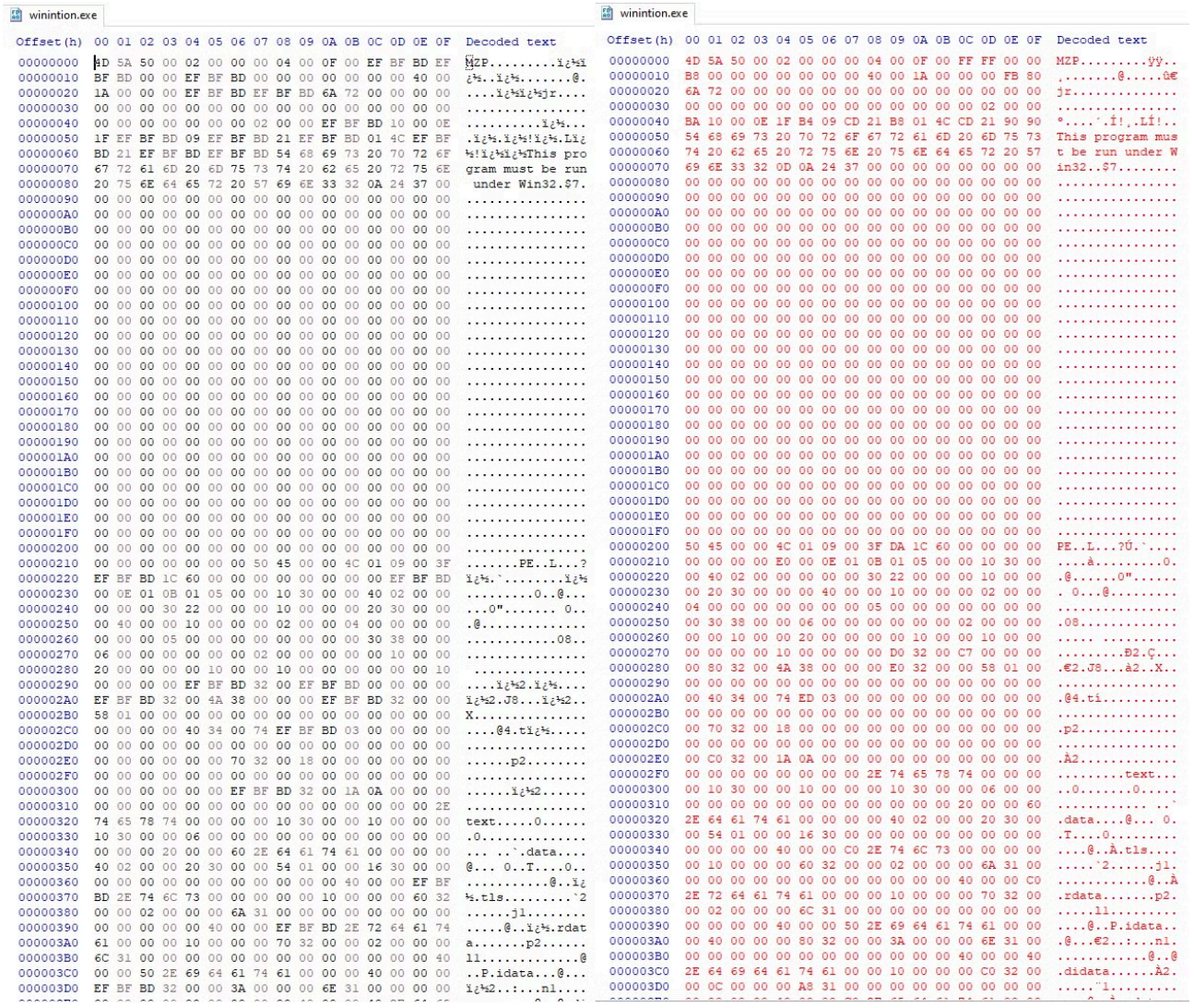
When selecting on UserForm1, the textbox reveals a `base64` encoded string; we know this because of the function we discussed above. The next step is to copy the entire string into a file so it can be decoded.



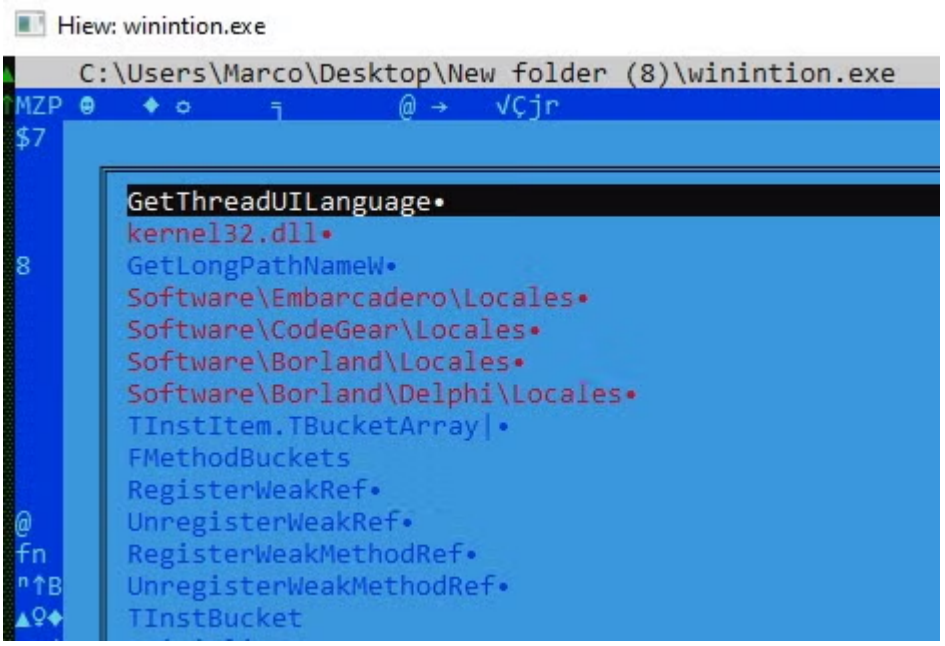
Now we decode the binary from `base64` and save it to disk as `wininition.exe`.



Following that, clean the headers using HxD, and then use PE-Bear to fix the sections headers to move to the next phase of the analysis.



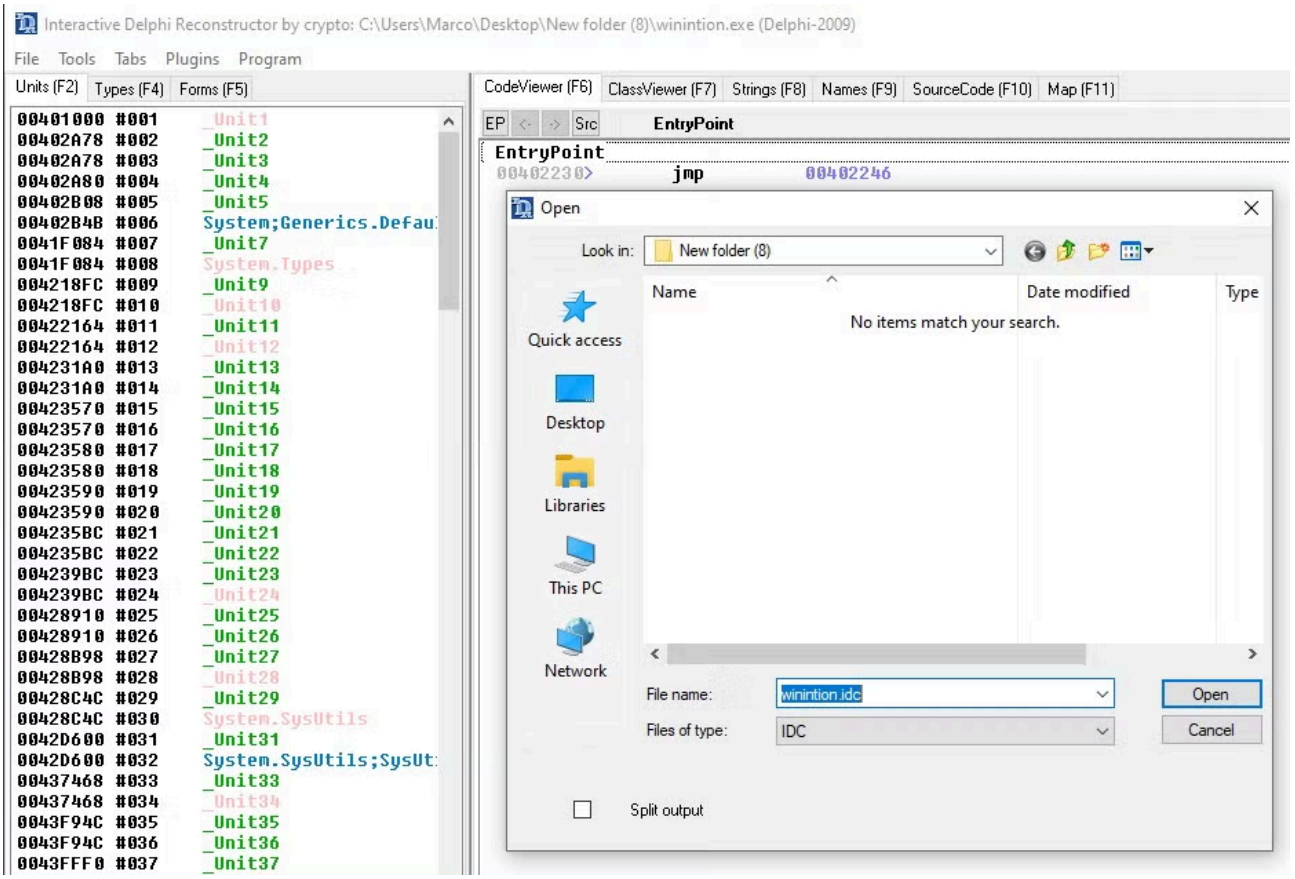
When triaging a binary, the go-to tool is Hiew to investigate and look for clues for a deeper understanding. With `wininition`, I notice the Embarcadero string, which means that this binary was written in Delphi. When reversing Delphi binaries I've always used IDR (Interactive Delphi Reconstructor). IDR is a decompiler of executable files and dynamic libraries (DLL) written in Delphi.

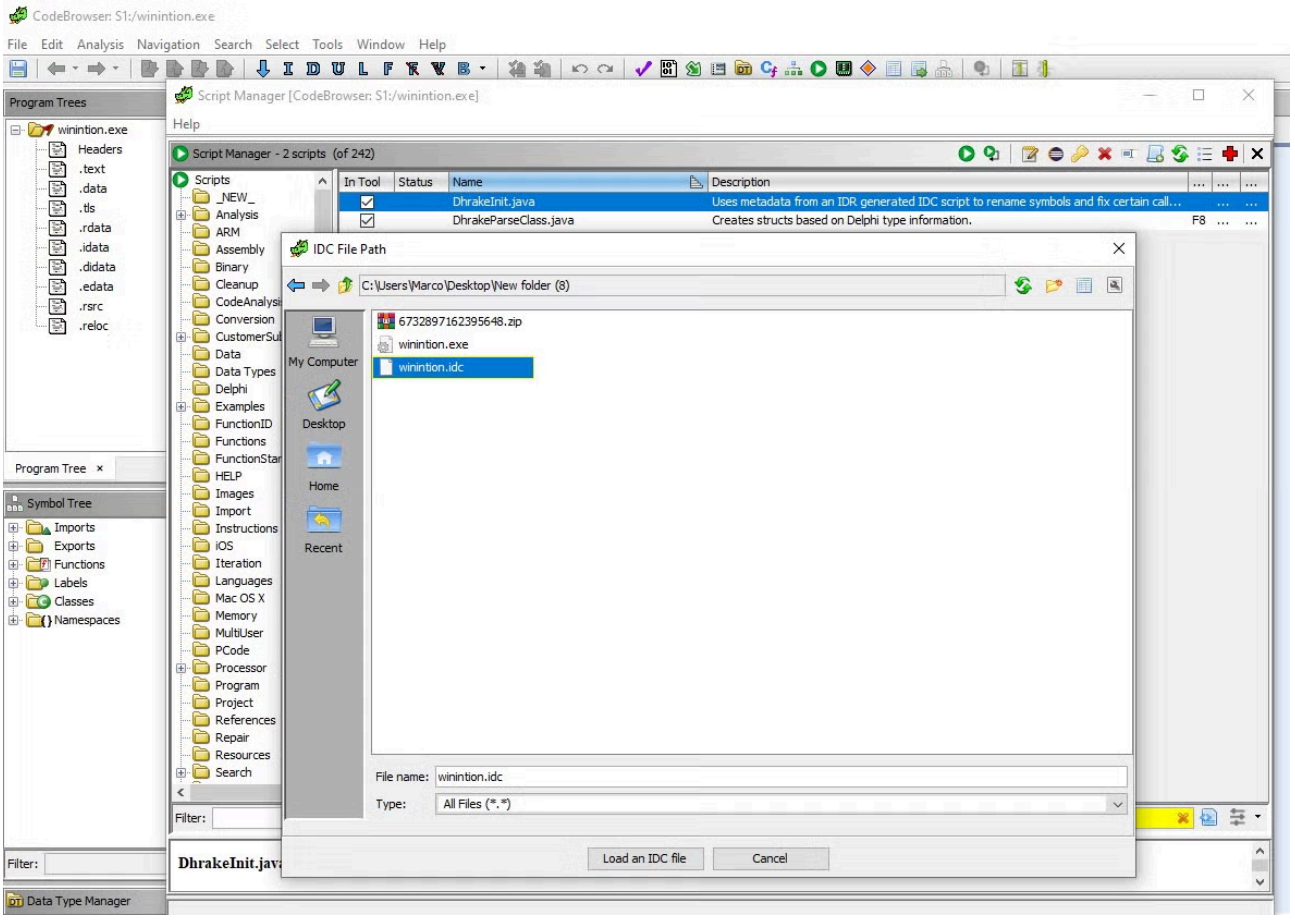


Reversing Delphi Binaries with Ghidra and dhrake

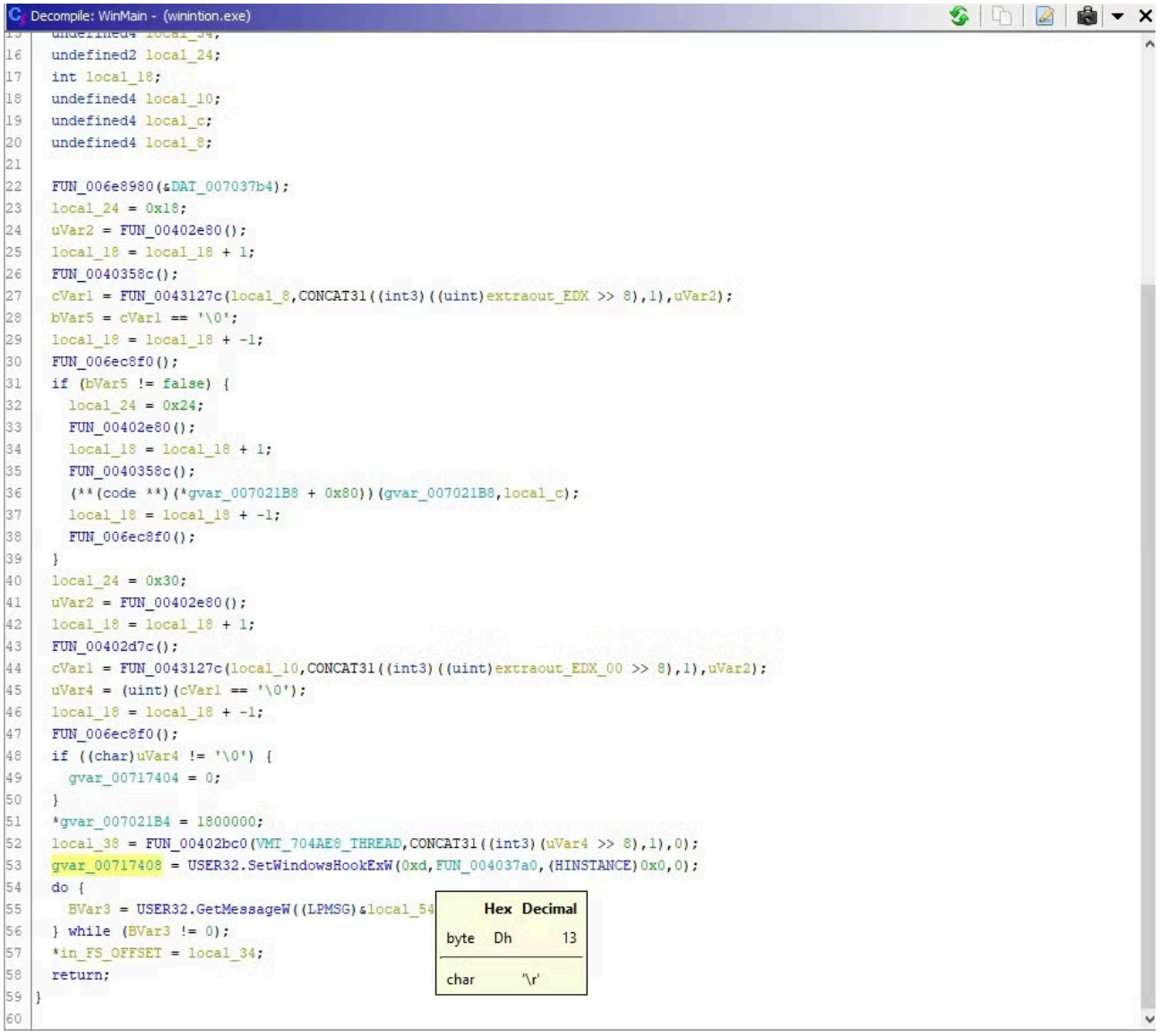
When searching for the latest developments with IDR, I came across a fantastic plugin for Ghidra, a collection of scripts for reverse engineering Delphi binaires in Ghidra using IDR's output to IDC. It was published over a year ago, but it is a gem if threat hunters are using Ghidra.

dhrake allows you to import the IDC file from IDR into Ghidra. This will import the Symbol names, function signatures and create structs for Delphi classes. This plugin extracts and applies the Delphi symbols from the IDC file, which is generated by IDR, and attempts to find cases where Ghidra has incorrectly determined the entry point of a function. If you've never imported a plugin to Ghidra please read this post. I've saved the IDC to a selected folder. I then install the plugin in Ghidra and run the script it prompts for the IDC file and then load it!


Interactive Delphi Reconstructor by crypto: C:\Users\Marco\Desktop\New folder (8)\winintion.exe (Delphi-2009)
File Tools Tabs Plugins Program
Units (F2) Types (F4) Forms (F5)
CodeViewer (F6) ClassViewer (F7) Strings (F8) Names (F9) SourceCode (F10) Map (F11)
EntryPoint
00402230> jmp 00402246
Open
Look in: New folder (8)
Name Date modified Type
No items match your search.
Quick access
Desktop
Libraries
This PC
Network
File name: winintion.idc
Files of type: IDC
Open
Cancel
Split output


CodeBrowser: S1:/winintion.exe
Script Manager [CodeBrowser: S1:/winintion.exe]
Program Trees
Script Manager - 2 scripts (of 242)
DhrakeInit.java
DhrakeParseClass.java
IDC File Path
6732897162395648.zip
winintion.exe
winintion.idc
File name: winintion.idc
Type: All Files (*.*)
Load an IDC file
Cancel

In the `wininition` binary, the first function WinMain has `SetWindowsHookExW` function, which is a hook procedure to monitor a system for certain types of events. The hook procedures low-level keyboard input events is `WH_KEYBOARD_LL`, which is the number 13 in the parameter. This hook is a mechanism that intercepts keystroke events. All the events are then saved to a log file to be sent to a C2.



The C2 is obfuscated using hex that can be converted to ascii:

```
68747470733A2F2F7777772E786268702E636F6D2F646F6D696E617267726561746173696E6F6479737365792F77702D636F6E74656E7
```

```
hxxps://www.xbhp[.]com/dominargreatasianodyssey/wp-content/plugins/akismet/style.php
```

```
68747470733A2F2F7777772E63346373612E6F72672F696E636C756465732F736F75726365732F66656C696D732E706870
```

```
hxxps://www.c4csa[.]org/includes/sources/felims.php
```

Note: These appear to be compromised domains.

Conclusion

Analysis of these documents led us to find other Zebrocy clusters. As Zebrocy continues to evolve its scope, organizations must have the proper visibilities and detection capabilities to find this threat actor. We hope the techniques discussed in this post will be useful to other researchers in analyzing Delphocy dropper docs in particular, and documents with password-protected macros in general.

Indicators of Compromise

Word Documents

SHA256
3b548a851fb889d3cc84243eb8ce9cbf8a857c7d725a24408934c0d8342d5811
1dd03c4ea4d630a59f73e053d705185e27e2e2545dd9caedb26a824ac5d11466
1e8261104cbe4e09c19af7910f83e9545fd435483f24f60ec70c3186b98603cc
c213b60a63da80f960e7a7344f478eb1b72cee89fd0145361a088478c51b2c0e
2bf088955007b4f47fe9187affe65fffea234ff16596313a74958a7c85129172
d9e7325f266eda94bfa8b8938de7b7957734041a055b49b94af0627bd119c51c
SHA1
fc0b7ad2ae9347d6d2ababe2947ffb9f7cc73030
71b4b9f105de94090fc36d9226faaa1db6d9f3d1
6a8f63c4491adcf2cf7f76cd1481c5647615a6c9
a3ecf1fdc1206e9d3061530fa91775cf3d97f788
ae01ca2cf0dc07abb3a7bef9930e38c9212975d5
66b39f4fd1dd51c2f548330e5818f732dad0aa28

VBA

SHA256
a442135c04dd2c9cbf26b2a85264d31a5ac4ec5d2069a7b63bc14b64a6dd82b7
SHA1
6ec4eb883752b70db134ac0f4e0d5b4a77196184

Wininition

SHA256
ee7cfc55a49b2e9825a393a94b0baad18ef5bfced67531382e572ef8a9ecda4b
SHA1
afbdb13d8f620d0a5599cbc7a7d9ce8001ee32f1

URLs

hxxps://www.xbhp[.]com/dominargreatasianodyssey/wp-content/plugins/akismet/style.php
hxxps://www.c4csa[.]org/includes/sources/felims.php

Yara Rules

```
rule apt_RU_delphocy_encStrings {
  meta:
    desc = "Hex strings in Delphocy drops"
    author = "JAG-S @ SentinelLabs"
    version = "1.0"
    TLP = "White"
    last_modified = "04.09.2021"
    hash0 = "ee7cfc55a49b2e9825a393a94b0baad18ef5bfced67531382e572ef8a9ecda4b"
    hash1 = "07b2d21f4ef077ccf16935e44864b96fa039f2e88c73b518930b6048f6baad74"

  strings:
    $enc_keylogger2 = "5B4241434B53504143455D" ascii wide
    $enc_keylogger3 = "5B5441425D" ascii wide
    $enc_keylogger4 = "5B53484946545D" ascii wide
    $enc_keylogger5 = "5B434F4E54524F4C5D" ascii wide
    $enc_keylogger6 = "5B4553434150455D" ascii wide
    $enc_keylogger7 = "5B454E445D" ascii wide
    $enc_keylogger8 = "5B484F4D455D" ascii wide
    $enc_keylogger9 = "5B4C4546545D" ascii wide
    $enc_keylogger10 = "5B55505D" ascii wide
    $enc_keylogger11 = "5B52494748545D" ascii wide
    $enc_keylogger12 = "5B444F574E5D" ascii wide
    $enc_keylogger13 = "5B434150534C4F434B5D" ascii wide
    $cnc1 = "68747470733A2F2F7777772E786268702E636F6D2F646F6D696E617267726561746173696616E6F647973736
    $cnc2 = "68747470733A2F2F7777772E63346373612E6F72672F696E636C756465732F736F75726365732F66656C696

  condition:
    uint16(0) == 0x5a4d and (any of ($cnc*) or all of ($enc_keylogger*))
}
```

```
rule apt_RU_Delphocy_Maldocs {
  meta:
    desc = "Delphocy dropper docs"
    author = "JAG-S @ SentinelLabs"
    version = "1.0"
    TLP = "White"
    last_modified = "04.09.2021"
    hash1 = "3b548a851fb889d3cc84243eb8ce9cbf8a857c7d725a24408934c0d8342d5811"
    hash2 = "c213b60a63da80f960e7a7344f478eb1b72cee89fd0145361a088478c51b2c0e"
    hash3 = "d9e7325f266eda94bfa8b8938de7b7957734041a055b49b94af0627bd119c51c"
```

```
    hash4 = "1e8261104cbe4e09c19af7910f83e9545fd435483f24f60ec70c3186b98603cc"

  strings:
    $required1 = "_VBA_PROJECT" ascii wide
    $required2 = "Normal.dotm" ascii wide
    $required3 = "bin.base64" ascii wide
    $required4 = "ADODB.Stream$" ascii wide
    $author1 = "Dinara Tanmurzina" ascii wide
    $author2 = "Hewlett-Packard Company" ascii wide
    $specific = "Caption         =   "wininition.exe"" ascii wide
    $builder1 = "Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} UserForm1" ascii wide
    $builder2 = "{02330CFE-305D-431C-93AC-29735EB37575}{33D6B9D9-9757-485A-89F4-4F27E5959B10}" ascii
    $builder3 = "VersionCompatible32="393222000"" ascii wide
    $builder4 = "CMG="1517B95BC9F7CDF7CDF3D1F3D1"" ascii wide
    $builder5 = "DPB="ADAF01C301461E461EB9E2471E616F01D06093C59A7C4D30F64A51BDEDDA98EC1590C9B191FF""
    $builder6 = "GC="4547E96B19021A021A02"" ascii wide

  condition:
    uint32(0) == 0xE011CFD0 and all of ($required*) and (all of ($author*) or $specific or 5 of ($bu
}
```

Source: https://labs.sentinelone.com/a-deep-dive-into-zebrocys-dropper-docs/