Cybersecurity
Zero to Hero
with CyberChef

Jonathan Glass
Script for the next ~40 mins

- Disclaimers
- Introduce Me/CyberChef
- Discuss the Value
- Walkthrough a Few Recipes (Small, Medium, Large)
- Advanced Use Cases (Building Custom Operations, Potential for Integration, Interacting with Active Content)
- Lessons Learned
Slide Legal made me make

• The views that I express are my own and do not necessarily represent
  • those of the Federal Reserve Bank of New York or the Federal Reserve System
  • those of the University of Richmond School of Professional and Continuing Studies
  • sound cybersecurity advice in general.

• View at your own risk
I refuse to tell you which is which
Introductions
Jonathan Glass

- **Federal Reserve (Present)**
  - Malware Analyst
  - Local and National Incident Responder
  - Forensic Analyst
- **University of Richmond School of Professional and Continuing Studies (Present)**
  - Adjunct Instructor
    - Digital Forensics
    - Malware Analysis
    - Black/Blue Hat Python
- **10 years Cybersecurity**
- **9 years USAF**
- **GCIH, GAWN, GCFA, CISSP, CEH, MODOK, MCSE, GPYC**
- **BS in InfoSec, MBA**
- **http://jon.glass**
- **email@jon.glass**
- **@GlassSec**
CyberChef

- [https://gchq.github.io/CyberChef/](https://gchq.github.io/CyberChef/)
- The Cyber Swiss Army Knife - a web app for encryption, encoding, compression and data analysis
How does it work?
Powerful Operations

- From/To Hex
- From/To Base64
- URL Encode/Decode
- Regular Expression
- XOR Brute Force
- Decode Text
- CSV to JSON
- JSON to CSV

- RC2, RC4, DES, Triple DES, AES Encrypt/Decrypt
- Bitwise operations
- HTTP request
- JPath Expression
- Strings
- Extract Filepaths
- Extract EXIF

- Zip/Unzip
- Tar/Untar
- All the Hashes
- Syntax Highlighting
- Script Beautify
- Render Image
- XKCD Random Number

- 300+ and growing!
Value of CyberChef
Value of CyberChef

• Reduces the entry threshold for Cybersecurity tasks
  • Drag and Drop operations
  • Menu of things to try
  • Web GUI

• Solid platform to demonstrate programming concepts
  • Functions, Order of operation, data types...
  • Visualize data manipulation step by step
  • Trick students into coding with RegEx
Value of CyberChef

• Serverless and Static
  • Runs client side
  • Nothing to install
  • Cross-browser compatibility

• Parses HTTP GET Parameters
  • Recipes can be bookmarked in browser with input data
  • Post URLs to Blogs with steps, comments, and input data

• Not overly difficult to customize
• Free!
REAL STATISTICS...probably not made up.
Intro to Digital Forensics
- Basic tasks easier in CyberChef but Python becomes very necessary.

Intro to Malware Analysis
- Small tasks can be combined to get big results but Python is still needed for most analysis

Black Hat Python
- Some tasks might be easier but we use Python for everything so…¯\_(ツ)_/¯

Work
- Wide variety of tasks, most are easier with Python until complexity reaches critical point where it really doesn’t matter what I am using. :D
Small Recipes Using CyberChef

Tons of value from the quick operations
Base64 Decode
Unzipping a Password Protected Zip File

Input:
- Name: $I_Parse_v1.1.zip
- Size: 19,121 bytes
- Type: application/zip
- Loaded: 100%

Output:
- 1 file(s) found
- $I_Parse.exe 81,376 bytes
Combining ‘Unzip’ and ‘From Base64’
Resolving a List of Domain Names

Recipe

- Fork
  - Split delimiter
  - Merge delimiter

- Ignore errors

DNS over HTTPS
- Resolver: https://dns.google.com/resolve
- Request Type: A
- Validate DNSSEC

JPath expression
- Query: Answer[0][name, data]
- Result delimiter

Input

- google.com
- apple.com
- jon.glass
- github.com
- thenegative.zone

Output

- "google.com." 172.217.6.238
- "apple.com." 17.172.224.47
- "jon.glass." 192.30.252.153
- "github.com." 192.30.253.113
- "thenegative.zone." 192.30.252.153
YARA? Sure!
Medium Recipe using CyberChef

Deobfuscating Emotet v4 Downloader
MUMMY SPIDER is a criminal entity linked to the core development of the malware most commonly known as Emotet or Geodo.

The phishing campaign by MUMMY SPIDER consisted of a malicious macro-enabled Microsoft Word document sent as an email attachment.

When recipients opened the weaponized document and macros are enabled on the machine (which is quite typical), an obfuscated PowerShell command was launched.

https://www.information-age.com/ecrime-cyber-network-123482383/
Grab RegEx Operation

- I use RegEx for as much as possible
- You should too
Too Many Carets(^)!!!

We should remove the DOS Obfuscation to get a better picture of this mess.
#1 Sets an environment variable filled with #2 mostly gibberish.
#3 loops backward from the end and grabs every 4th character.
More RegEx to Capture the Obfuscated Code
Reverse The Obfuscated Code

Reverse

By

Character

Output

pIqmo8RXrEZpeY60r05s45sMh negI1l87sKkJ3nc
KfoSwoFH3h0K0mpmcF21u jmv'uVlqJfZJ2XCMsW0'T11; dEKsWgVkJU8vS60WzLZ'ZuT'010'h00qC0DvmtHcK6:TB/Wb/No/1Yw9g0Tw0OfWfZ
sc.axmUpD0o8ifKvdXK4i01r mfj16p0rYcVhryUqj8foXLINoNah.e.66cU7oZ4m7QwU0/159dvzarReAHhuETyB7tb0R54pgJehDd
xtwrttbcbqnp1c: UXy/g9/m FdrnRB81XcYma6Ro5Inj99GorSxW9jS'ZnPhR8ekes80rdavenar.G
Ketz5dzwDu0jz, PpAvGxq563P/37H0LSC/mzAhV0vLtPwToY
pk0R: 3q4y/pTy/n8vecq77s2nX0i07rlUv77crg9.KksfsPG0nI2/ jts5ueDyzfjwGnB5g0RSdHDzFdt8tbrhpxV0: v1/mZp/FepfYBv0uXq
GnEeE bsITQTyXbezyOksdo1z0kFexx0, ptyCAb5NMC/ E2uWjyyipu9v-
a1oa&usdrumwvRip1ntPy/ncLsJ8qKka5f08syc7dbkLdALF6dIG@esHezitrlWtcMFPx6/I1/rk4/KJqefYqsGF
keBCrTna1NjmZbaU5uShLZe67cnTsaPZ0mi kpgnoH6f1 .7q0rdublSN/cbUQ oSvbi6AE
WwhM4k3Hc K'mv,b.f959yWpp0ClBxJq8n75W(M/-it'wZq0v'Zm) 'Z5'vH1y:gPLsyySKKBcRcsFn63j q5P(yJaIkJc8Bauyurs.00ctBssEli
muYr.m8zoI0QOPFC.L7z9poan7t0rBhLxIjz:O: OUT0: T103G4nerekJtkcL1IGweBkm4usQ9P99r9aMKetkP3hDhT (Sw9) E21 +rok 'LHN
azjizNPnPah37AE, 55HeWvfxh1YePX3C'GM8'uYy; Vwv$0 m0qKazLPRYTK K di8=KUDNAnaZe0y1y3E5- d50laubDYjYWHEmD5cLW87t TU 0
-0-hukvcUlo5y9mjqf onv3'3bTwmdhsfKUxXsAa70hlky21yg.EHxsqRlMcG618XkomrtGxvteExsptGY'6XlY; Uy8BjquHrgrswr3cvAY
YvW8=4b JKnHg6er9gwwr-4v7J5srbxn5emhymEwKGCokuEtxy kjS-6yZycEoET7xmHam x0' nNevd0evdvn
ospdmdgbQSM.qushaxtnXARpNiezAtZ UjmVen'MS/k; JTEfewe6oM 9rncubeeSnNarBzEoE1r6rg (dhus051Mdp7nppN1i 8 shu5Ib5YnGx
rhNj0idg9B3iaQVRdWQ) lp(ez tpeyrtnYyowd(04504A4ERVwuml0HYw.v800XvpufkeF
enD9X(9LdX'XgbbgeLOE6P8STKh6 C'IUN, VgJ2sWMqRncsO011F, wkU0BXc)n;CN0;R75H0kOcnJna5hYpak dfsdAbsFeWXenEOFdgok(tC7x) dQW;
GOIC518FYv (AUCnP QGmhea9hYcK, foTMYJfJ Rs4a9pYjtuNRRfsQa7 NRh+1GeFvxsqsQ Fxz2bT0bHR0N5Q9) BOF
aMm(Wv5$1bEB8S1sCvLc6c3.B16OFPf96Aeucp711x(Cur)DMX; kve5D0FgH93a5cGxt3p.KoDtr1cGKVpkz7e649 Cvu= vuren
5PClAh1; kboBHsMkmQKboCpHp.LhhWbsrZx5Hwmaa8jl (dw bUB0Jqhpapia0Y0To. VjRjA
eAubsSmxjoxNoKyu0vnN3sDrDswSvbFhQogT4DntmyjBv60h; jvbo8Wthucjos5ecv2r.g1s4UCa3p7yGoe0CoBt50x8ef4DgjXpl2HB2e
cuM{M4sqqSS5XRST n1wi}ZP:w52sKDe6EAAepXrjPuxbc-GWBPZizwr04 opj1jcmi0eT7xnsisjP1e YmTspHPSL4c4Rn E
09; wrrBgbFrso10eHwBa5RSKx47a)I8j;jwBekcezwa897iz71crN9uhP(h2G) y6n) amap Gp4 Wj a65s JAs 2myXk6f zv a0n n u
VCX 15T RfN XWH d2D JDT hjd
More RegEx!

“(.).”

Capture the first character, skip 3, repeat
A Pattern Emerges!!!
Now we have deobfuscated code for the downloader!!!
One final round of RegEx and using the Split Operator...
We have FIVE glorious C2 addresses for the price of one!

http://www.vladimirfilin.com/VzBE7R
http://nimsnowshera.edu.pk/D
http://sinonc.cn/uz6
http://forestbooks.cn/wp-admin/sFfyyqdF
http://eskrimadecampo.ru/UVAwk
Structure from chaos in 8 Drag and Drop operations

http://www.vladimirfilin.com/VzBE7R
http://nimsnowshera.edu.pk/D
http://sinonc.cn/uz6
http://forestbooks.cn/wp-admin/sFfyqdf
http://eskrimadecamparo.ru/UVAwk
Not only easy but, repeatable!

- Recipes can be saved to local storage for reuse, be given as gifts, or exchanged for beer.
- Data Links can be stored as bookmarks
Large Recipes using CyberChef

Building a Parser for Windows Recycle Bin Metadata
Parsing Windows Recycle Bin Metadata

- Why? The simplest forensic artifact I can think of

<table>
<thead>
<tr>
<th>Prior to Windows 10</th>
<th>Offset</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td></td>
<td>Header</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>File Size</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td></td>
<td>Deleted Timestamp</td>
</tr>
<tr>
<td>24</td>
<td>520</td>
<td></td>
<td>File Name</td>
</tr>
</tbody>
</table>

$I$ structure prior to Win 10

<table>
<thead>
<tr>
<th>Windows 10</th>
<th>Offset</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td></td>
<td>Header</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>File Size</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td></td>
<td>Deleted Timestamp</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td></td>
<td>File Name Length</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td>var File Name</td>
</tr>
</tbody>
</table>

Windows 10 $I$ structure
Using Native CyberChef to Parser Recycle Bin Meta

The idea is fairly simple as far parsers go.
The Reality Ends Up being ~24 Steps

Conditional_Jump('^(\x01|\x02)',true,'Error',10)
Find_/_Replace({'option':'Regex','string':'^(\x02.{23})(....)'},'$1',false,false,false,false)
Subsection('^.{24}{.*}',true,true,false)
Decode_text('UTF16LE (1200)')
Find_/_Replace({'option':'Regex','string':'^(.*).'},'\nDeleted File Path: $1',false,false,false,false)
Merge()
Subsection('^.{16}{.*}',false,true,false)
Swap_endianness('Raw',8,true)
To_Hex('None')
Windows_Filetime_to_UNIX_Timestamp('Seconds (s)','Hex')
From_UNIX_Timestamp('Seconds (s)')
Find_/_Replace({'option':'Regex','string':'^(.* UTC)'},'\nFile Deletion Time: $1',true,false,true,false)
Subsection('^.{8}(.{8})',true,true,false)
To_Hex('None')
Swap_endianness('Hex',8,true)
From_Base(16)
Find_/_Replace({'option':'Regex','string':'^.{8}$'},'******** WINDOWS RECYCLE BIN METADATA ********',true,false,false,false)
Find_/_Replace({'option':'Regex','string':'^.*$'},'This doesn\'t look like a Recycle Bin file to me ',true,false,true,false)
Find_/_Replace({'option':'Regex','string':'^.*$'},'This doesn\'t look like a Recycle Bin file to me ',true,false,true,false)
Conditional Jump

Match (regex)
^\x01\x02

Invert match
Label name Error

Maximum jumps (if jumping ... 10

Find / Replace

Find
^\x02\{23\}\(.\)

Replace
$1

Global match
Case insensitive
Multiline matching
Dot matches all

Subsection

Section (regex)
^\{24\}\(.\)

Output

********** WINDOWS RECYCLE BIN METADATA **********
Deleted File Size: 13012 bytes
File Deletion Time: Tue 8 January 2019 02:31:28 UTC
Deleted File Path: C:\Users\Username\Desktop\http_20190102_122044.txt

Name: $1EOEO.txt
Size: 130 bytes
Type: text/plain
Loaded: 100%
Advanced Use Cases

Building Custom Operations
Potential for Integration
Interacting with Active Content
Before you sell your soul to JavaScript...

• Rolling your operations can be really helpful but...
  • How good is your JavaScript *writing* really?
  • If you are going to be coding to do DFIR work, you probably should just be using Python
    • Better Community support
    • Better memory management
    • Better Syntax

• Now that you have been cautioned...
  • LET’S LOOK AT AN EXAMPLE I DID JUST TO PROVE IT COULD BE DONE!
Coding Time!

- A Windows RecBin Parser in JavaScript

- Features:
  - Converting Windows FILETIME object to Date
  - Converts UTF-16LE File Path to UTF-8
  - Converts LE File size to decimal

- Overall, not horrible.
  - Probably could be written better if I am being honest but it works
Output is fairly clean...

Parsed File Metadata:

- Name: $IEEEO.txt
- Size: 130 bytes
- Type: text/plain
- Loaded: 100%

Output:

- Deleted File Size: 13012
- Deleted File Path: C:\Users\Username\Desktop\http_20190102_122044.txt.
Potential for Integration
How to get CyberChef to talk to VirusTotal... at your own risk

• Download CyberChef

• Open Chrome with web protections turned off
  • “--disable-web-security”

• HTTP Request Operation

• https://stackoverflow.com/questions/3102819/disable-same-origin-policy-in-chrome
Download Samples
VirusTotal Query Reports

HTTP request
Method
GET

URL
https://www.virustotal.com/vtapi/v2/file/rept...

Headers

Mode
Cross-Origin Resource Share

JPath expression
Query
$.scans..result

Result delimiter

Output
"Backdoor/W32.Bladabindi.264704"
"I-Worm.Mawanella!"
"TrojWare.Win32.TrojanDropper.Dexel.A@6k1yft"
"Trojan.Diple.Win32.79656"
"Trojan.Siggen6.57104"
Interacting with Live Content

—

Not for the faint of heart
Recommended for Sandboxes Only
Hand to Hand Combat with Malicious JavaScript
Lessons Learned

Tips and Tricks
Not ideal for everything

- Memory management being what it is, don’t be surprised if a large file knocks it over.
- Don’t parse a whole $MFT
- Don’t parse a whole memory dump
- Take Bytes, Drop Bytes, and RegEx can help make the data more manageable but they aren’t miracle workers.
- Use the right tool for the right job.
Use The Comment Field Like Notepad

- Helps to not have to switch back and forth to take notes.
- Comments do not effect the operation but can be saved into the Recipe!
- Comment Early and Comment Often
Mind Meld with Your Friends!

https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')Gunzip()JavaScript_Beautify('%5C%5Ct','Auto'true)Find/_/_Replace(%7B'option':'Regex','string':'%5C%5C%5C%20%5C%5C%2B%20%5C%5C%5C','%7D',',true,false)&
Turn off “Auto Bake” unless you need it.

- Auto Bake runs the recipe whenever anything changes in the input or the recipe.
- Can cause issues when designing steps.
The Value

Practical Applications
- Data Manipulation
- Deobfuscation Malware
- Forensic Artifact Parsing

Advanced Use Cases
- Building Custom Operations
- Potential for Integration
- Interacting with Active Content

Lessons Learned