Python Basics
for
Web App Pentesters Part 2

Justin Searle
Managing Partner - UtiliSec
justin@utilisec.com

UtiliSec

Why Python

- Pre-installed on Mac and Linux
- Easy to install on Windows
- Easy to write scripts that run on all OSes
- Easy to read and collaborate
- Very complete set of standard libraries
- Many stable and powerful 3rd party libraries

Python Shell

- Using an interactive python shell
  - type “python” on your command line
  - type python commands
  - they execute when you hit enter
- Why use the shell?
  - Easy way to learn the language
  - Great way to debug portions of code
  - Nice for PoC functions and loops
- Beyond the basic shell
  - Consider ipython or IDLE after you get your feet wet
  - Provide richer functionality and productivity

Input and Output

```
print('This site is protected by SSL.')
```

Basic output

Basic input

```
answer = raw_input('Do you wish to continue? ')
```

object oriented bliss

```
if answer.lower() == 'no':
    print('Exiting the program.')
else:
    print('Continuing Program.')
```

if / then / else conditional statements. Notice the colons followed by mandatory line indentions

A Tale of Two Libraries

urllib2	**httplib**
HTTP, HTTPS, & FTP	HTTP & HTTPS
Auto Parses URI	No URI Parsing
Follows Redirections	Doesn’t Follow Redirects
Uses a Cookie Jar	Doesn’t Store Cookies
Auth: Basic & Digest	Authentication: None
Methods: GET & POST	Method: Any
Supports Proxy Servers	No Proxy Support
Auto Closes Connections	Manually Close Connection

Using httplib

Create a “connection” object

Domain only

```
import httplib

connection = httplib.HTTPConnection("secureideas.net")
connection.request("TRACE", "/index.html")

response = connection.getresponse()
payload = response.read()

print(payload)
```

Network request made here

Extract reponse

Extract payload

Using urllib2

The library that does the magic

```
import urllib2
```

This doesn't make the request, it simply packages the request

```
request = urllib2.Request('http://www.utilisec.com')
```

Don't for get the "http://"

```
response = urllib2.urlopen(request)
payload = response.read()
```

This sends the request, catches the response, and extracts out the response payload

```
print(payload)
```

POST Requests

```
import urllib2, urllib

url = 'http://whois.arin.net/ui/query.do'
data = { 'flushCache' : 'false',
           'queryinput' : '198.60.22.2'}
data = urllib.urlencode(data)
request = urllib2.Request(url, data)

response = urllib2.urlopen(request)
payload = response.read()

print(payload)
```

Add your POST data to a dictionary

Then urlencode your data
(don't forget to import urllib)

If you provide urllib2 with request data, it will assume a POST

Working with Headers

```
import urllib2

url = 'http://google.com/#q=samurai-wtf'
headers = { 'User-Agent' : 'Mozilla/5.0 (iPhone)' }
request = urllib2.Request(url, None, headers)

response = urllib2.urlopen(request)
headers = response.headers

print(headers)
```

Add your headers to a dictionary

If you are doing a GET, use None for data

Writing to a File

```
import urllib2

request = urllib2.Request('http://www.utilisec.com')

response = urllib2.urlopen(request)
payload = response.read()

with open('index.html', 'wb') as file:
    file.write(payload)
```

Try opening a file, in write and binary modes

Write the payload to the file

Filtering Responses

```
import urllib2, re
request = urllib2.Request('http://inguardians.com/info')
response = urllib2.urlopen(request)
payload = response.read()

regex = r'<dt class="title">(.*)</dt>'
results = re.findall( regex , payload )

for result in results:
    print(result)
```

Build your regex using a raw string, grouping desired text

Search payload for all instances of your regex

Loop through your results printing them

Basic Authentication

```
import urllib2

url ='http://browserspy.dk/password-ok.php'
username = 'test'
password = 'test'

password_mgr = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_mgr.add_password(None, url, username, password)
authhandler = urllib2.HTTPBasicAuthHandler(password_mgr)
opener = urllib2.build_opener(authhandler)
urllib2.install_opener(opener)

response = urllib2.urlopen(url)

payload = response.read()
print( payload )
```

Setup needed variables

Setup password manager

Add passwords

Connect handler

Build and install so all requests automatically use the password manager

Fuzzing and Brute Force

```
import urllib2, re

list = (1533095958 + i for i in range(0, 20) )

for item in list:
    url = 'http://m.facebook.com/people/a/' + str(item)
    try:
        response = urllib2.urlopen(url)
    except IOError:
        pass
    else:
        payload = response.read()
        regex = r'<strong>([^<]*)'
        match = re.search(regex, payload)
        if match:
            name = match.groups()
            site = response.geturl().split("?")[0]
            print("{0} = {1}    {2}".format(item, name[0], site) )
```

Create list of 20 Facebook IDs

Prevent missing pages from throwing an error and stopping the script

Extract name from page

Grab url and remove redirect reference

Format output

Cookie Jars

```
import urllib2, cookielib

headers = { 'User-Agent' : 'Mozilla/5.0 (iPhone)' }

cookiejar = cookielib.CookieJar()
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cookiejar))
urllib2.install_opener(opener)

request1 = urllib2.Request('http://www.google.com', None, headers)
response1 = urllib2.urlopen(request1)

request2 = urllib2.Request('http://www.google.com', None, headers)
response2 = urllib2.urlopen(request2)
payload2 = response2.read()

print(payload2)
```

Custom HTTP Methods in urllib2

```
import urllib2

method = 'TRACE'
request = urllib2.Request('http://www.secureideas.net')
request.get_method = lambda: method.upper()

response = urllib2.urlopen(request)
payload = response.read()

print(payload)
```

Multi-Threading

```
import urllib2
from Queue import *
from threading import *
num_worker_threads = 10
url = 'http://www.google.com'
def grab():
	while True:
			q.get(True,None)
			req = urllib2.Request(url)
			response = urllib2.urlopen(req)
			payload = response.read()
			print(payload)
			q.task_done()
q = Queue(10)
for i in range(num_worker_threads):
		t = Thread(target=grab)
		t.setDaemon(True)
		t.start()
q.join()
```

pyCIT

- Python Commandline Interface Templates
  - http://code.google.com/p/pycit
  - a collection of python templates for creating command line tools
  - great tool for beginners to show how to do the basics
  - saves advanced users time by providing the basic and much more
- Each templates will give you:
  - Built in command line arguments, easily modifiable
  - Provides a help page if no arguments are given
  - Tracks and displays your script version
  - Verbosity and debug functions with command line flags
  - Command line options and functions for reading and writing to files

pyCIT Templates

- Completed Templates
  - Basic file read/write access
  - Single-threaded http requests (basic wget/curl functions)
- Templates in Progress
  - Multi-threaded http requests (basic wget/curl functions)
- Planned Templates
  - Binary file read/write access with hex decode (basic xxd/hexdump functions)
  - Raw socket client and service (basic netcat functions)
  - Raw usb device access
  - Interactive CLI interface

Contact Information

Justin Searle

Managing Partner - UtiliSec

work: justin@utilisec.com

personal: justin@meeas.com

twitter: @meeas

http://code.google.com/p/pycit

http://samurai-wtf.org