Python Basics
for
Web App Pentesters
Part 1

Justin Searle
Managing Partner – UtiliSec
Certified Instructor – SANS Institute
justin@utilisec.com
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
print('This site is protected by SSL.')

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

if answer.lower() == 'no':
    print('Exiting the program.')
else:
    print('Continuing Program.')
# A Tale of Two Libraries

<table>
<thead>
<tr>
<th>urllib2</th>
<th>httplib</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP, HTTPS, &amp; FTP</td>
<td>HTTP &amp; HTTPS</td>
</tr>
<tr>
<td>Auto Parses URI</td>
<td>No URI Parsing</td>
</tr>
<tr>
<td>Follows Redirections</td>
<td>Doesn’t Follow Redirects</td>
</tr>
<tr>
<td>Uses a Cookie Jar</td>
<td>Doesn’t Store Cookies</td>
</tr>
<tr>
<td>Auth: Basic &amp; Digest</td>
<td>Authentication: None</td>
</tr>
<tr>
<td>Methods: GET &amp; POST</td>
<td>Method: Any</td>
</tr>
<tr>
<td>Supports Proxy Servers</td>
<td>No Proxy Support</td>
</tr>
<tr>
<td>Auto Closes Connections</td>
<td>Manually Close Connection</td>
</tr>
</tbody>
</table>
Using httplib

```python
import httplib

connection = httplib.HTTPConnection("secureideas.net")
connection.request("TRACE", "/index.html")
response = connection.getresponse()
payload = response.read()
print(payload)
```

Create a “connection” object

Domain only

Network request made here

Extract payload

Extract response
Using urllib2

The library that does the magic

```python
import urllib2

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

response = urllib2.urlopen(request)

payload = response.read()

print(payload)
```

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

Don't forget the "http://"

This sends the request, catches the response, and extracts out the response payload
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
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)
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)
Filtering Responses

```python
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

```python
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**
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) )
New SANS Course

SEC573: Python for Penetration Testers

- 5 Day Hands-on Class
  - Day 1: variables, operators, statements, introspection
  - Day 2: lists, loops, tuples, dicts, debugger, sys, files
  - Day 3: sockets, exceptions, metasploit, AV, IDS, SQLi
  - Day 4: fuzzing web apps, network recon, scapy, pcaps
  - Day 5: capstone and capture the flag

- First time in US: Washington DC, June 2013
- First time in AsiaPAC: Singapore, Feb 2014
- http://www.sans.org/course/python-for-pen-testers
Contact Information

Justin Searle
Managing Partner - UtiliSec

work: justin@utilisec.com
personal: justin@meeas.com
twitter: @meeas

http://samurai-wtf.org