Ethical Student Hackers

Automation in Python

SHEFFIELD Ethical Student Hackers
Breaking into security.

The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is VERY easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.

- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.

- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.

Code of Conduct

- Before proceeding past this point you must read and agree to our Code of Conduct - this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.
- Code of Conduct can be found at https://shefesh.com/downloads/SESH%20Code%20of%20Conduct.pdf

SHEFFIELD Ethical Student Hackers
Breaking into security.

Why Automate?

Save time in daily life!

- Speeds up repetitive, menial tasks
- Reverse engineer something tricky so you don't have to memorise the steps

Understand the task at hand

- To build a tool to solve a problem, you have to understand that problem
- Automation lets you explore the intricacies and low level parts of the problem
- Gives you skills to apply it in future when you don't have tools to hand

Replicate + improve common libraries

- Not necessarily trying to build a better sqlmap - but understand how it works
- In some cases, you CAN make minor improvements
- Picking apart someone else's code lets you see its weak points

The Basics

On Unix machines (and windows subsystems) you can:

- Setup aliases in ~/.bashrc to quickly execute common commands - for example:
  - alias gitgone=‘git add -A; git commit -m “Commit everything blindly”; git push --force origin master’
  - alias whoops=‘sudo $(history -p !!)’
  - Remember to source your bash file!
- Write bash scripts
  - Bash scripts will execute the commands inside, in order
  - They have the .sh file extension
  - A plaintext file with the "#!/bin/bash" shebang will also be interpreted this way
  - Run it with ‘bash script.sh’ or ‘./script’ if using shebang
  - See https://www.linux.com/training-tutorials/writing-simple-bash-script/
  - If writing in Windows (e.g. WSL) you might need to run dos2unix to convert EOFs

Passing Arguments to Bash

Want your bash script to do something more interesting, like taking user input? Pass it an argument or a flag!

In Unix, arguments are supplied after the call to the Unix command - it is the same when invoking a script. In the examples below, arguments are highlighted in pink and flags in blue

- ssh -i ./access.pem 127.0.0.1
- nmap 127.0.0.1 -sC -sV

Access them in your script by doing the following:

- '$x' for the xth argument
- Use 'getopts' and a case statement to get a flag
- If using both flags and positional arguments, use the '$@' variable and 'shift' instead
- More details: https://www.baeldung.com/linux/use-command-line-arguments-in-bash-script and https://pretzelhands.com/posts/command-line-flags

What about Python?

Python can do all sorts

- File manipulation with file, os, and filepath
  - f = open('/path/to/file.txt,'r')
  - for item in os.scandir ...
- Run Unix commands with the subprocess module
  - subprocess.check_call(['ssh','127.0.0.1'])
- HTTP requests with the requests library - can use requests.get, requests.put etc for each HTTP verb, and add parameters - you can save the response for later
  - response = requests.get(‘https://www.shefesh.com’)
  - requests.post('https://httpbin.org/post', data={'key':'value'})
- API calls
  - Make requests directly to an API - include your API Key as a parameter if needed
  - response = requests.get("http://api.openweathermap.org/data/2.5/forecast", params={'APPID':'API_KEY', 'q':'Seattle,US'})
- It can even take arguments! (See https://www.tutorialspoint.com/python/python_command_line_arguments.htm)

Making Sense of Response Data

When we make a GET request, we usually get a large chunk of HTML as a response. We can use BeautifulSoup - the essential web scraping library - to search through this

- Parses a response object from a request
- Allows searching and partitioning by HTML elements and IDs - elements referred to as tags

Key Methods

- tag['id'], tag.get_text(), find(id='id'), findAll('tag', {"key": "value"}), tag.parent, tag.element

Process

- What element of the page do we want? Have a look at the page structure in Developer Tools
- Make a request and save the response
- Use BeautifulSoup to find the elements you want!
- Learn more here: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

SHEFFIELD Ethical Student Hackers
Breaking into security.

Basic Beautiful Soup

Simple Scraping

- Let’s make a request to our site, using requests.get(“https://www.shefesh.com”)
- Now let’s pull out some useful stuff - we could look for <h3> elements, <img> sources or <script> tags - what function should we use to do this?
- Anything you want to see? What else might be useful?

Paginated Scraping

- Find a URL that includes a page number as a parameter, and set up a loop
- Or find all the link hrefs you want to explore on the page, and make subsequent requests (this is what a web spider does)
- Try this out at https://scrapingclub.com/

Back to Juice Shop

Enumerating baskets

- Baskets retrieved using GET request to /rest/basket/id
- In Python this looks like requests.get("https://juice-shop.herokuapp.com/rest/basket/id", headers=headers)

We'll be using my custom-built Repeater tool

- Set the payload (remember to include authorization headers)
- Tell it how to iterate
- Inject the payload into the defined position + print the responses

This functionality is similar to Burp's Repeater tool that you may have used - this is essentially what it does!

Check out the code here: https://github.com/Twigonometry/CTF-Tools/tree/master/repeater

Capture that Captcha

Juice Shop behaves… differently to a normal site

- GET requests always return the same base HTML response
- The page content is updated by a heavily obfuscated Javascript function - which I didn't fancy picking apart!
- This is an effective (yet bizarre) defence - but it means we can't use normal web scraping methods...

But we can still look at how the Captcha is delivered

- Requested by call to /rest/captcha - grab the answer from here!
- Captcha ID submitted as part of form - we can change this to one we know the answer to

Alternate solutions

- Somehow intercept the /rest/captcha request as the page loads
- Use Selenium to simulate browser activity and calculate the captcha answer on the fly

SHEFFIELD Ethical Student Hackers
Breaking into security.

Other Tools

Python Server (https://github.com/Twigonometry/CTF-Tools/blob/master/scripts/simple-python-server.py)

- Script for launching simple server
- Allows file hosting (useful for uploading files to a box!)
- Also handles POST requests (useful for stealing data!)

Password Cracking (https://github.com/Twigonometry/CTF-Tools/tree/master/password_cracker)

- Hashes are one-way, uniformly distributed, and deterministic - perfect for passwords!
- Wordlist based password cracking relies on lists of stolen passwords
  - Brute force hashes them one-by-one
  - Dictionary attacks make use of precomputed lists
- The password cracker implements brute force and dictionary attacks for several algorithms
- It could be improved in several ways - concurrency, permutations and transforms on passwords, sorting the dictionaries and using binary search, and more!

SHEFFIELD Ethical Student Hackers
Breaking into security.

Useful Links

Repositories

- https://github.com/Twigonometry/CTF-Tools
- https://github.com/Twigonometry/sesh-automation

Tools

- Practice requests - https://httpbin.org/
- Practice scraping - https://scrapingclub.com/

Courses

- https://www.linkedin.com/learning/using-python-for-automation
- https://www.udemy.com/course/automate/ (I haven't done this one, but it's popular!)

And, as always, visit https://www.shefesh.com/wiki/resources for a comprehensive list!

Upcoming Sessions

What’s up next?
www.shefesh.com/sessions

28th October - Postgrad Activity Fair

2nd November - Networking

7th-8th November - HackSheffield 6!
(see https://hacksheffield.com/)

9th November - All the Shells!

16th November - Enumeration

Any Questions?

www.shefesh.com

SHEFFIELD Ethical Student Hackers
Breaking into security.