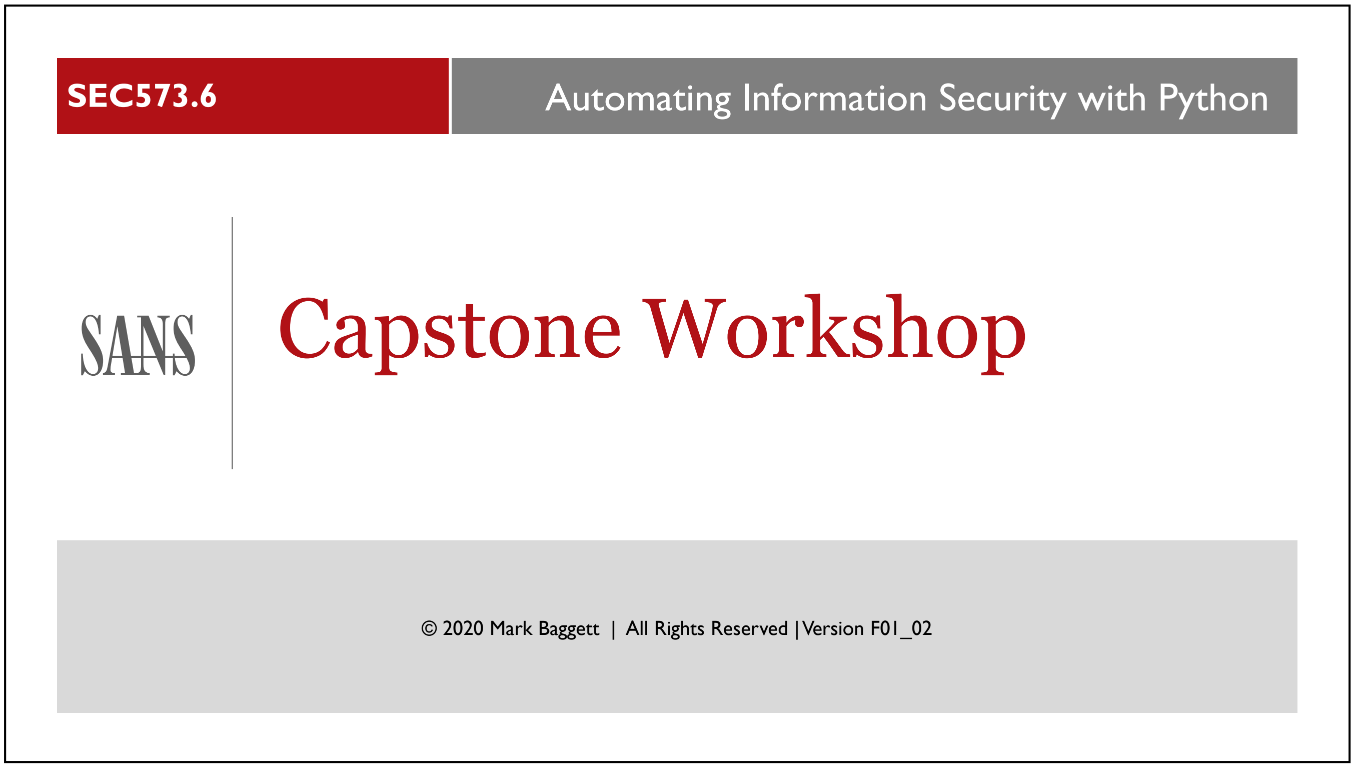
573.6
Capstone Workshop

SANS

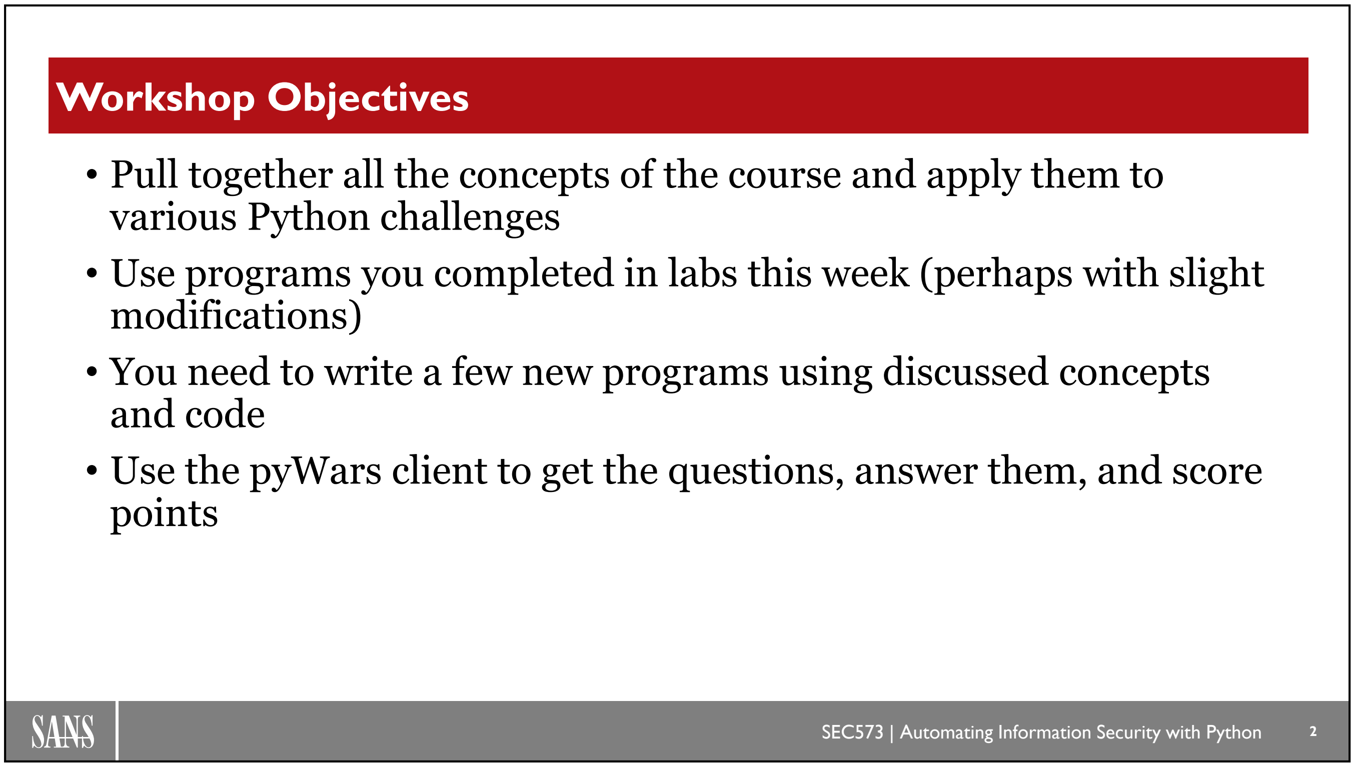


SEC573.6 Automating Information Security with Python

SANS

Capstone Workshop

Welcome to the final workshop of the SANS Automating Information Security with Python course.



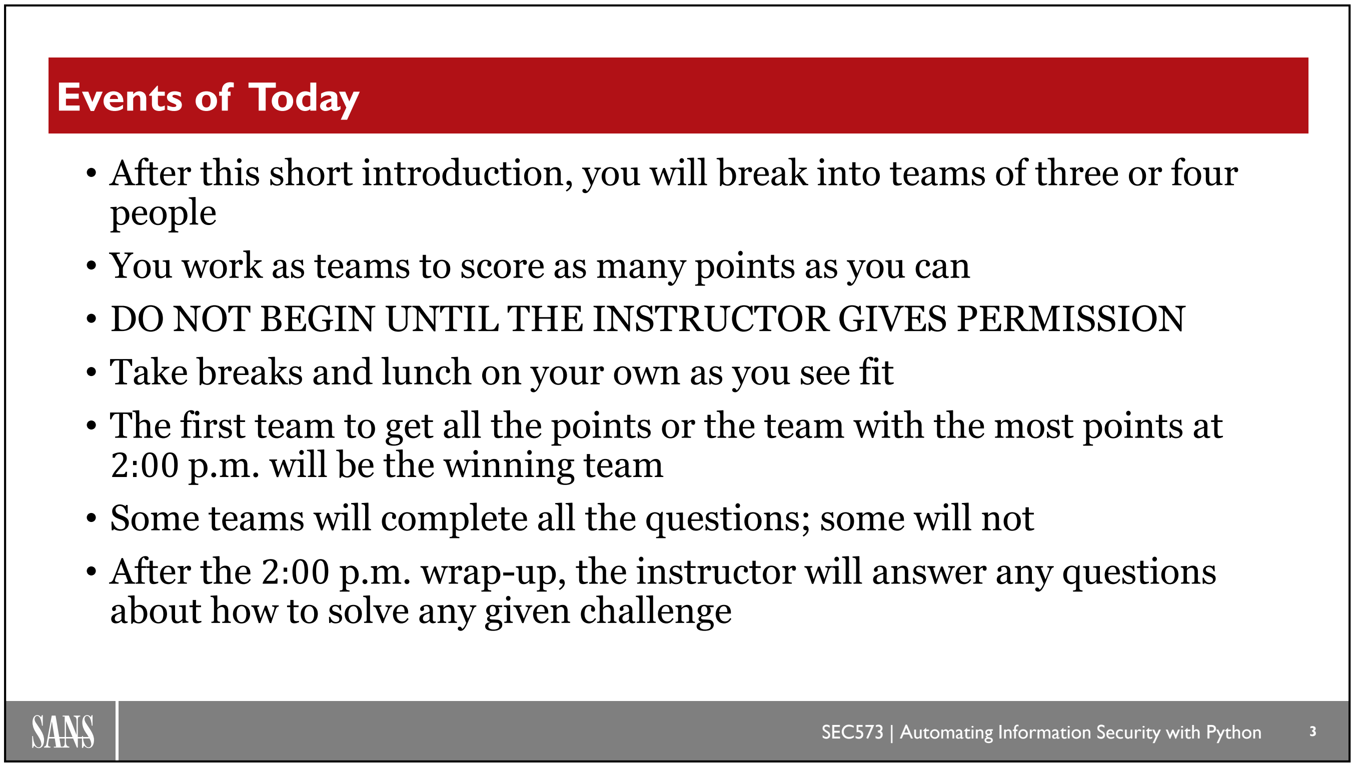
Workshop Objectives

- Pull together all the concepts of the course and apply them to various Python challenges
- Use programs you completed in labs this week (perhaps with slight modifications)
- You need to write a few new programs using discussed concepts and code
- Use the pyWars client to get the questions, answer them, and score points

SANS | SEC573 | Automating Information Security with Python 2

We have covered a lot of ground over the last five days. We discussed many key programming concepts as we developed those projects, including reading websites, filesystems, parsing data with regular expressions, incorporating third-party modules, creating executables, and other tasks that you will use frequently as security professionals.

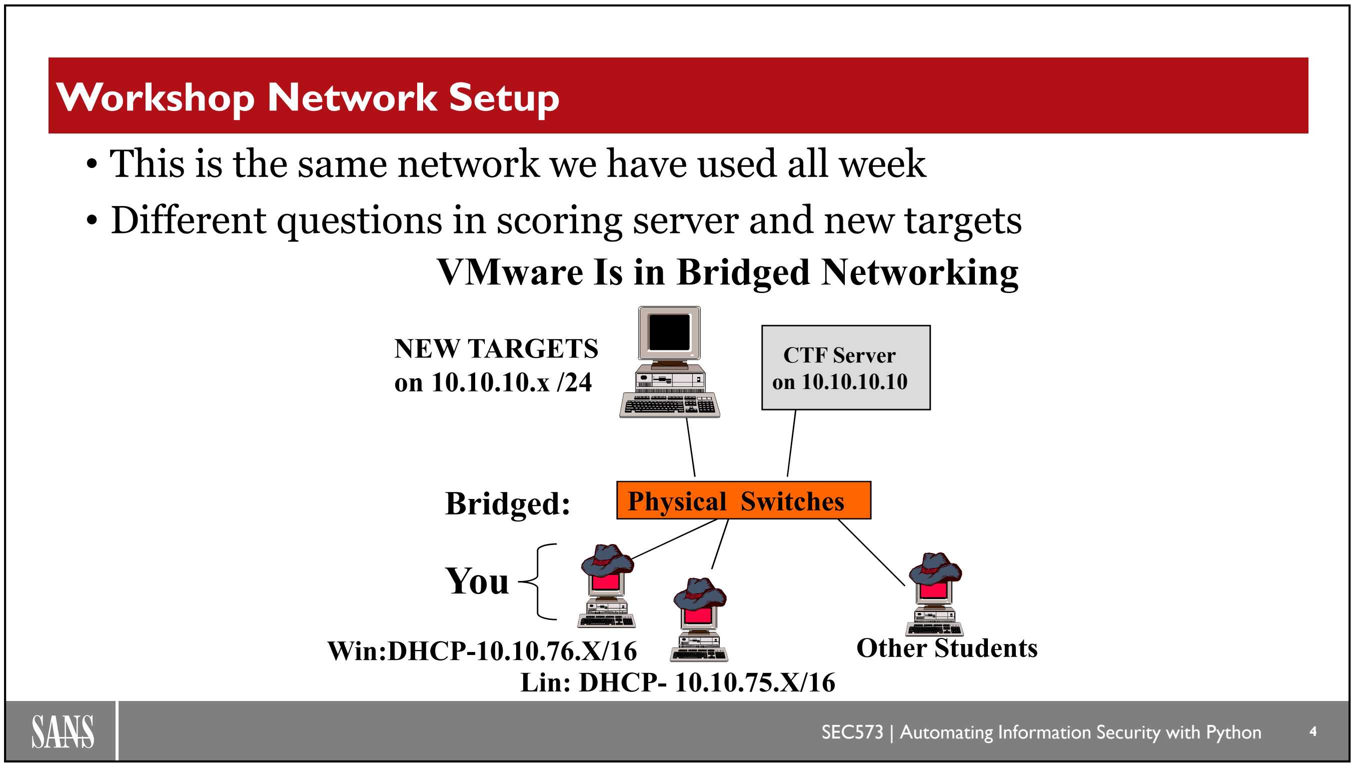
Today you put all those concepts together and apply them to challenges you will find in real-world penetration tests. You use the pyWars client to register points scored as you complete the challenges.



Events of Today

- After this short introduction, you will break into teams of three or four people
- You work as teams to score as many points as you can
- DO NOT BEGIN UNTIL THE INSTRUCTOR GIVES PERMISSION
- Take breaks and lunch on your own as you see fit
- The first team to get all the points or the team with the most points at 2:00 p.m. will be the winning team
- Some teams will complete all the questions; some will not
- After the 2:00 p.m. wrap-up, the instructor will answer any questions about how to solve any given challenge

Today's agenda is simple. We will introduce the workshop. Then, after you are given permission to begin, you will work together through the various challenges. The first team to complete all the questions will be the winner. At 2:00 p.m. today, we will have a debrief, and if no team has completed all the questions, the team with the most points at that time will be the winner. You will manage your breaks and lunch on your own.



Workshop Network Setup

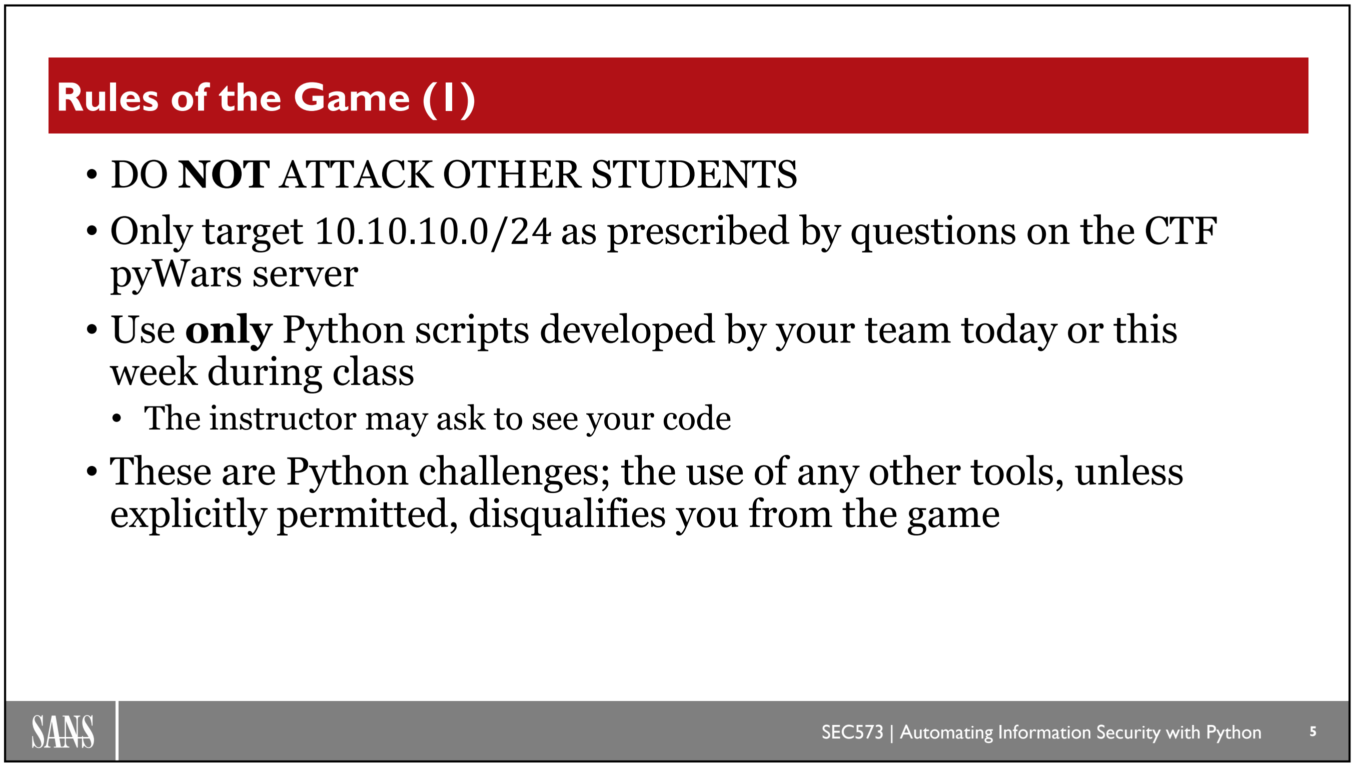
- This is the same network we have used all week
- Different questions in scoring server and new targets

VMware Is in Bridged Networking

SANS SEC573 | Automating Information Security with Python 4

Our network setup is the same as it has been for the past week. The scoring server at 10.10.10.10 will be replaced by a CTF server with a different set of questions. There are also going to be additional targets on the network in the 10.10.10.0/24 network range. We will discuss these targets and their role in the game in a few minutes.

The IP addresses you currently have been assigned will work properly, and no network reconfiguration will be required to play the game.



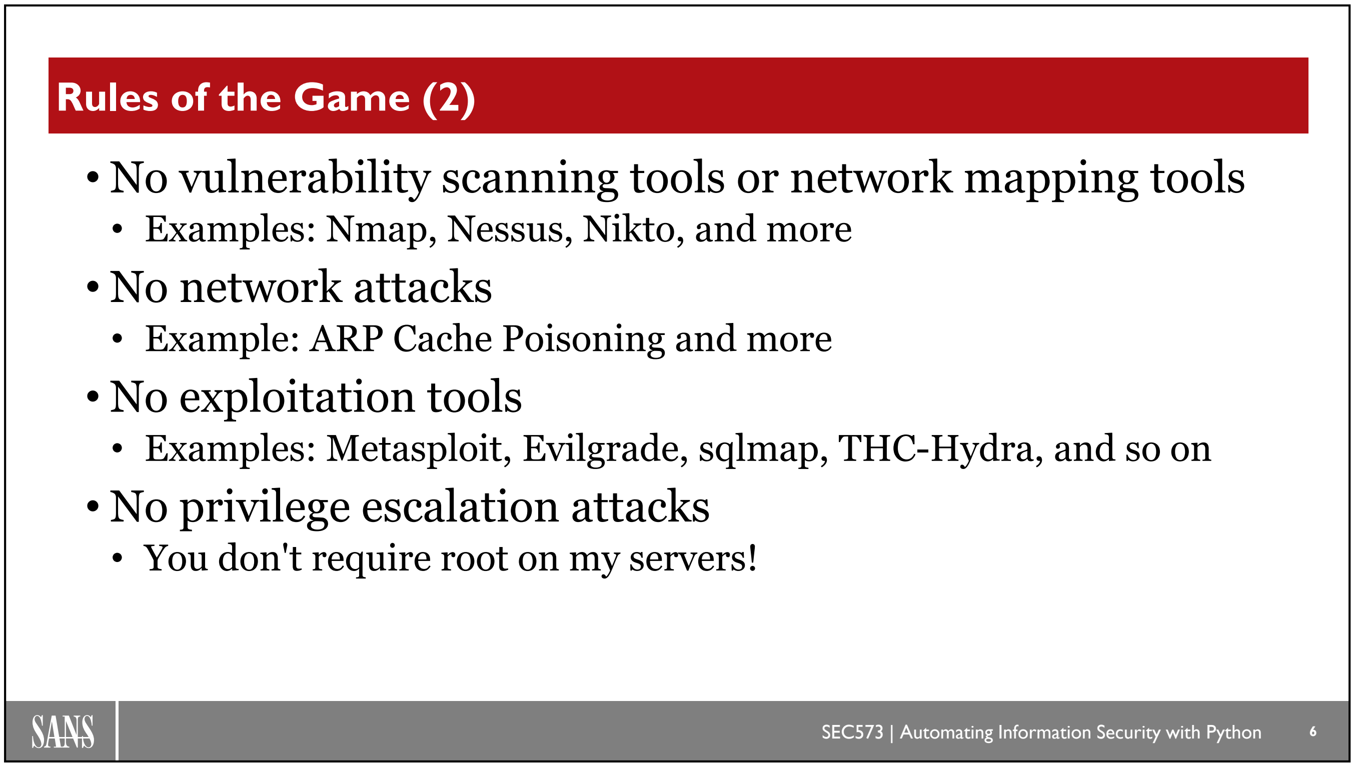
Rules of the Game (1)

- DO **NOT** ATTACK OTHER STUDENTS
- Only target 10.10.10.0/24 as prescribed by questions on the CTF pyWars server
- Use **only** Python scripts developed by your team today or this week during class
  - The instructor may ask to see your code
- These are Python challenges; the use of any other tools, unless explicitly permitted, disqualifies you from the game

Here are your rules of engagement. Be sure to carefully follow these rules to win the game. First, do not attack any other students. You should be targeting only hosts on the 10.10.10.0/24 network and only when told to do so by the questions on the CTF pyWars server. Second, this is a Python coding challenge. Use programs that your team has written over the last week, or write some new programs today, to complete the challenges. Do not use any programs that have been written by someone else. Do not use other open-source tools. For example, you may be tempted to try and use THC-Hydra to solve a password guessing challenge or sqlmap to extract SQL data from a vulnerable target. Don't do that. Instead, write your own tool to do the job.

Do not attempt to hack the scoring server or interact with the pyWars server with anything other than the pyWars client unless explicitly told to in a challenge.

The instructor may ask to see your code during the workshop.

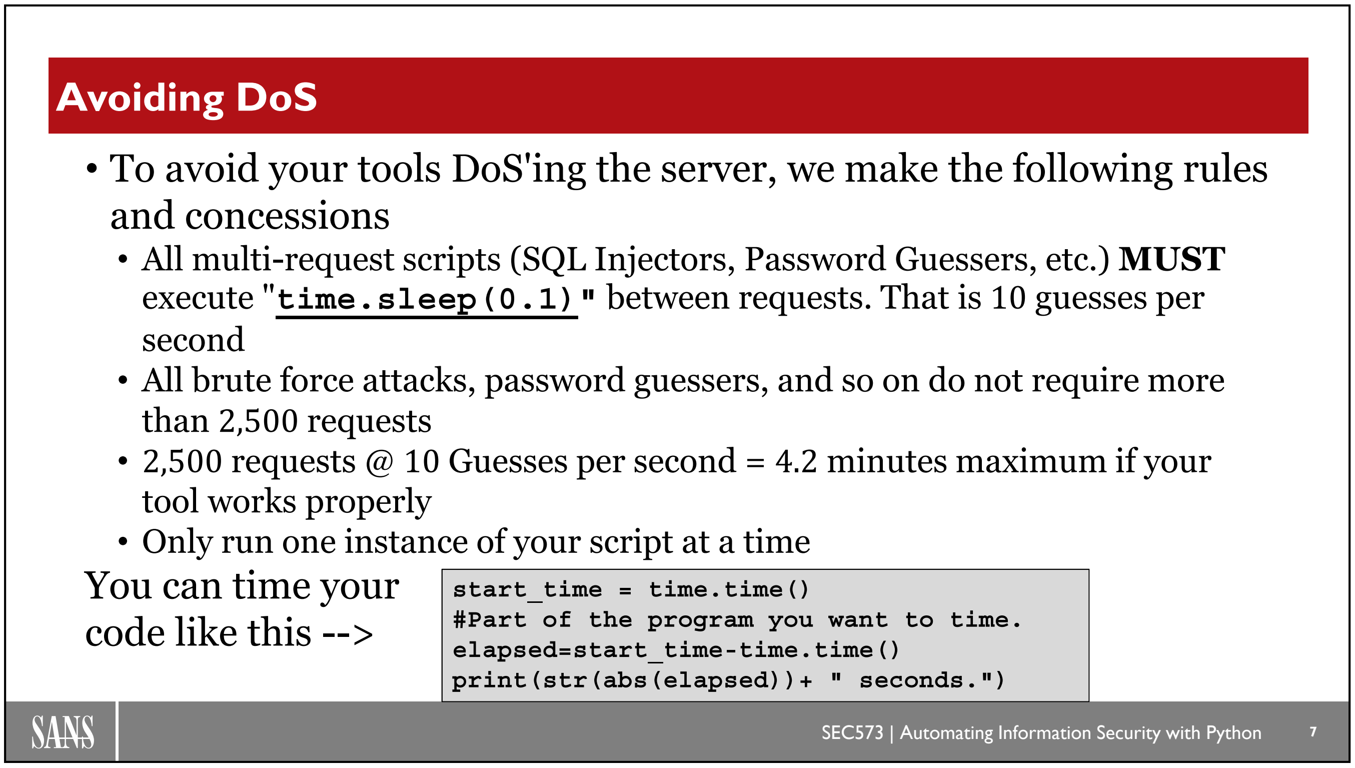


Rules of the Game (2)

- No vulnerability scanning tools or network mapping tools
  - Examples: Nmap, Nessus, Nikto, and more
- No network attacks
  - Example: ARP Cache Poisoning and more
- No exploitation tools
  - Examples: Metasploit, Evilgrade, sqlmap, THC-Hydra, and so on
- No privilege escalation attacks
  - You don't require root on my servers!

SANS SEC573 | Automating Information Security with Python 6

Remember, it is a Python programming challenge. You will not have a need for any vulnerability scanners, network mapping tools, exploits or exploitation frameworks, or privilege escalation attacks. Solve the challenges with your Python programming skills.



Avoiding DoS

- To avoid your tools DoS'ing the server, we make the following rules and concessions
  - All multi-request scripts (SQL Injectors, Password Guessers, etc.) **MUST** execute "`time.sleep(0.1)`" between requests. That is 10 guesses per second
  - All brute force attacks, password guessers, and so on do not require more than 2,500 requests
  - 2,500 requests @ 10 Guesses per second = 4.2 minutes maximum if your tool works properly
  - Only run one instance of your script at a time

You can time your code like this -->

```
start_time = time.time()
#Part of the program you want to time.
elapsed=start_time-time.time()
print(str(abs(elapsed))+ " seconds.")
```

SANS SEC573 | Automating Information Security with Python 7

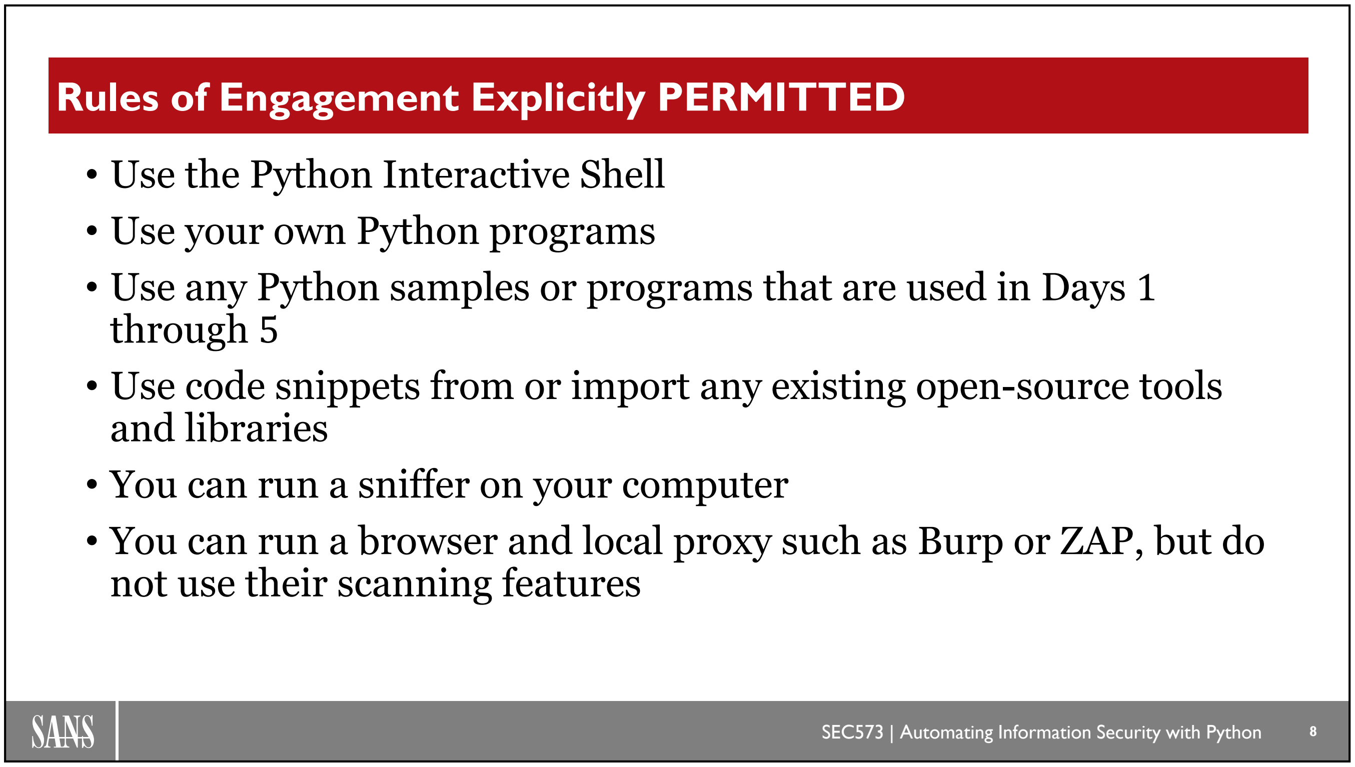
To avoid all these password guessers causing a denial of service on the target servers, we will establish the following rule and concession.

Rule: All multi-request scripts that need to repeatedly make requests to the servers must sleep 0.1 seconds. To have your script sleep for 0.1 seconds, call time.sleep(0.1). This will limit the number of requests you can make to only 10 per second.

Concession: If your program is written properly, it will not require more than 2,500 requests to successfully guess or brute force the solution to the challenge. So, at 10 guesses per second, it will not take more than 4.2 minutes to find the solution. If your script runs for more than 5 minutes, your program isn't working properly. Don't allow your programs to continue to attack the servers for more than 5 minutes.

Although not required, you can use the following code to time your code:

```
start_time = time.time()
#Place the timer notification IN the loop that is running
elapsed=start_time-time.time()
print("Elapsed time is "+ str(abs(elapsed))+ " seconds.")
```



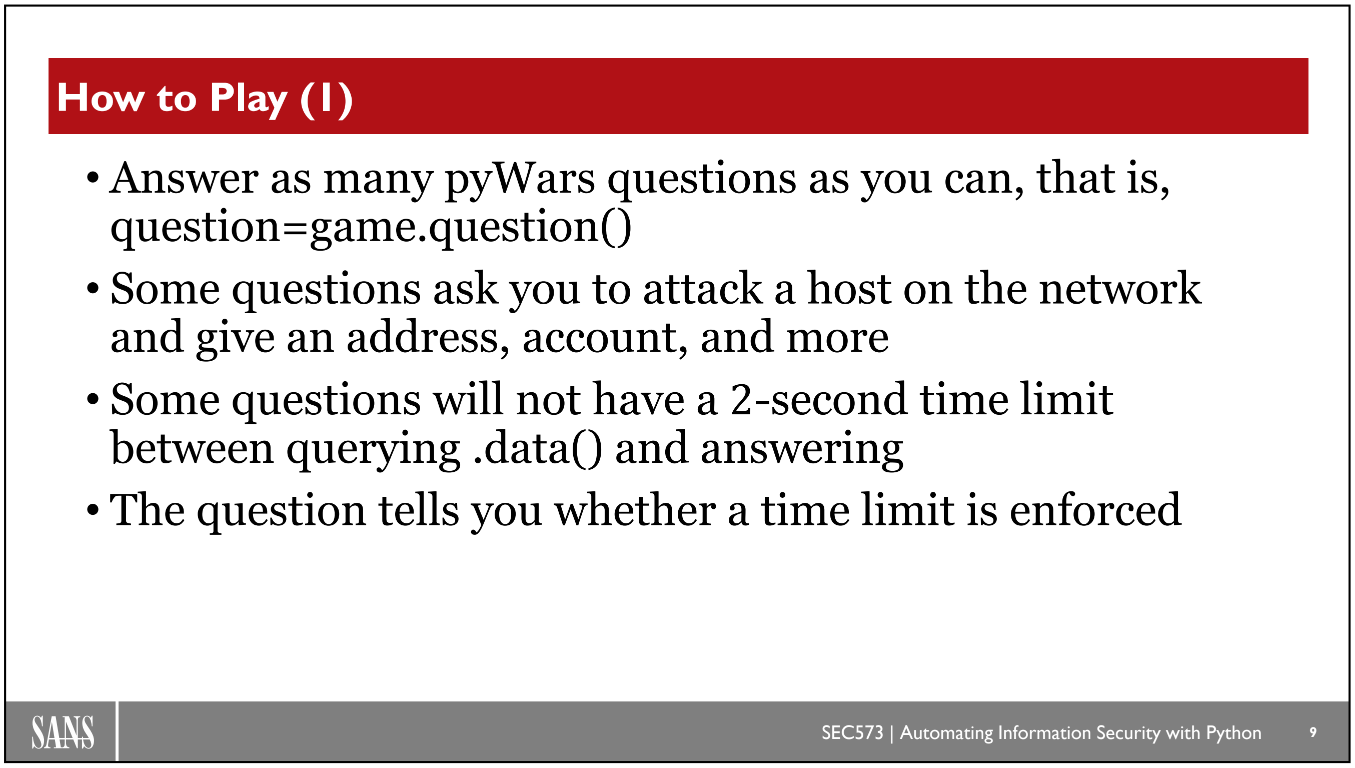
Rules of Engagement Explicitly PERMITTED

- Use the Python Interactive Shell
- Use your own Python programs
- Use any Python samples or programs that are used in Days 1 through 5
- Use code snippets from or import any existing open-source tools and libraries
- You can run a sniffer on your computer
- You can run a browser and local proxy such as Burp or ZAP, but do not use their scanning features

SANS SEC573 | Automating Information Security with Python 8

You definitely should use your own Python scripts and the Python interactive shell. You are welcome to use any program samples, code samples that were provided to you in the books, and labs this week. Any tools that you developed, including your password guesser, SQL injection tool, backdoor, and recon tools, are all fair game. You can borrow snippets of code or import any existing open-source tools or libraries that you'd like.

For non-Python-related tools, you may find running a sniffer or local proxy application useful.

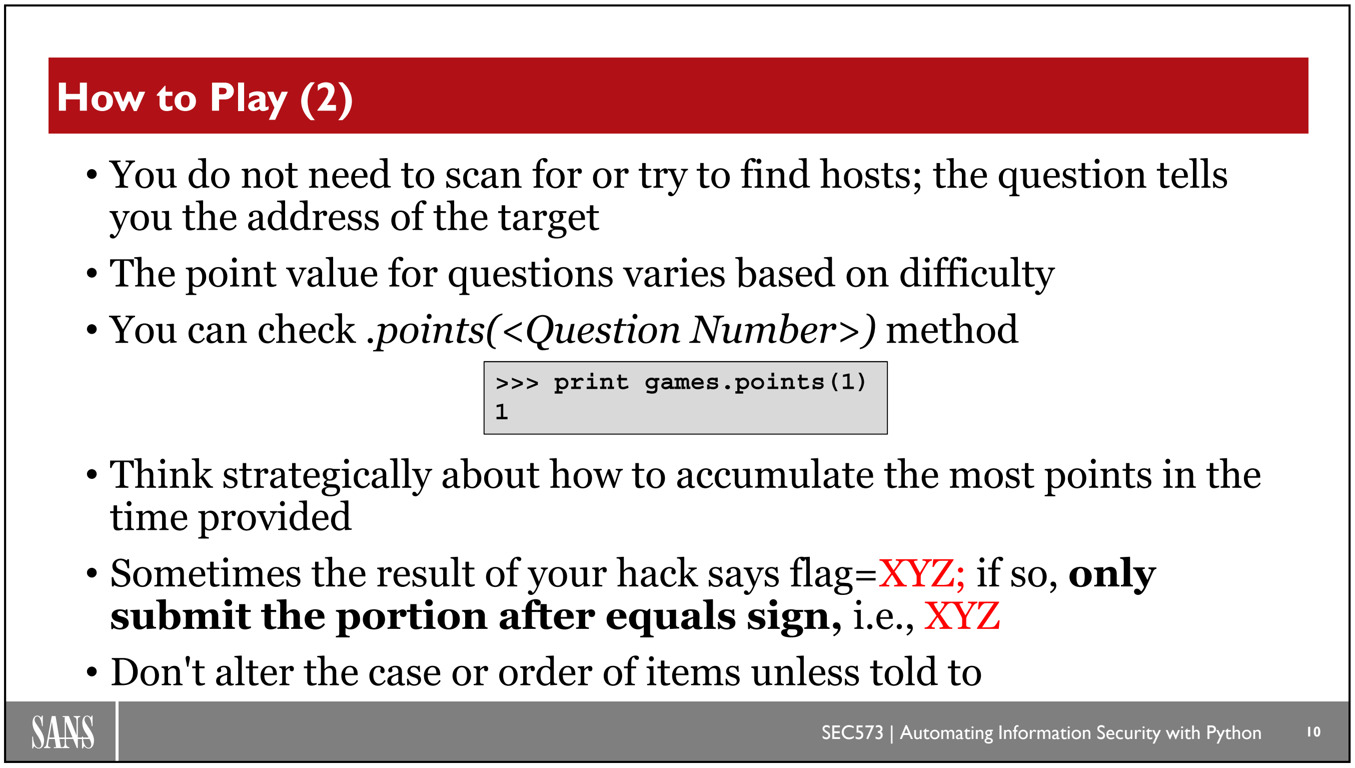


How to Play (1)

- Answer as many pyWars questions as you can, that is, question=game.question()
- Some questions ask you to attack a host on the network and give an address, account, and more
- Some questions will not have a 2-second time limit between querying .data() and answering
- The question tells you whether a time limit is enforced

SANS SEC573 | Automating Information Security with Python 9

Your goal is to answer as many questions as you can. Use pyWars to retrieve the questions and the associated data, and answer as many questions as possible in the time permitted. This is similar to playing pyWars just as you have all week, with a few minor differences. The first difference is the fact that there are additional targets on the network. Questions direct you to interact with and manipulate those targets to solve a challenge. Another difference is that some questions that target those external hosts do not require that you submit the answer within 2 seconds of querying the data element. If the time limit is not enforced, that will be stated in the question.



How to Play (2)

- You do not need to scan for or try to find hosts; the question tells you the address of the target
- The point value for questions varies based on difficulty
- You can check *.points(<Question Number>)* method

```
>>> print games.points(1)
1
```

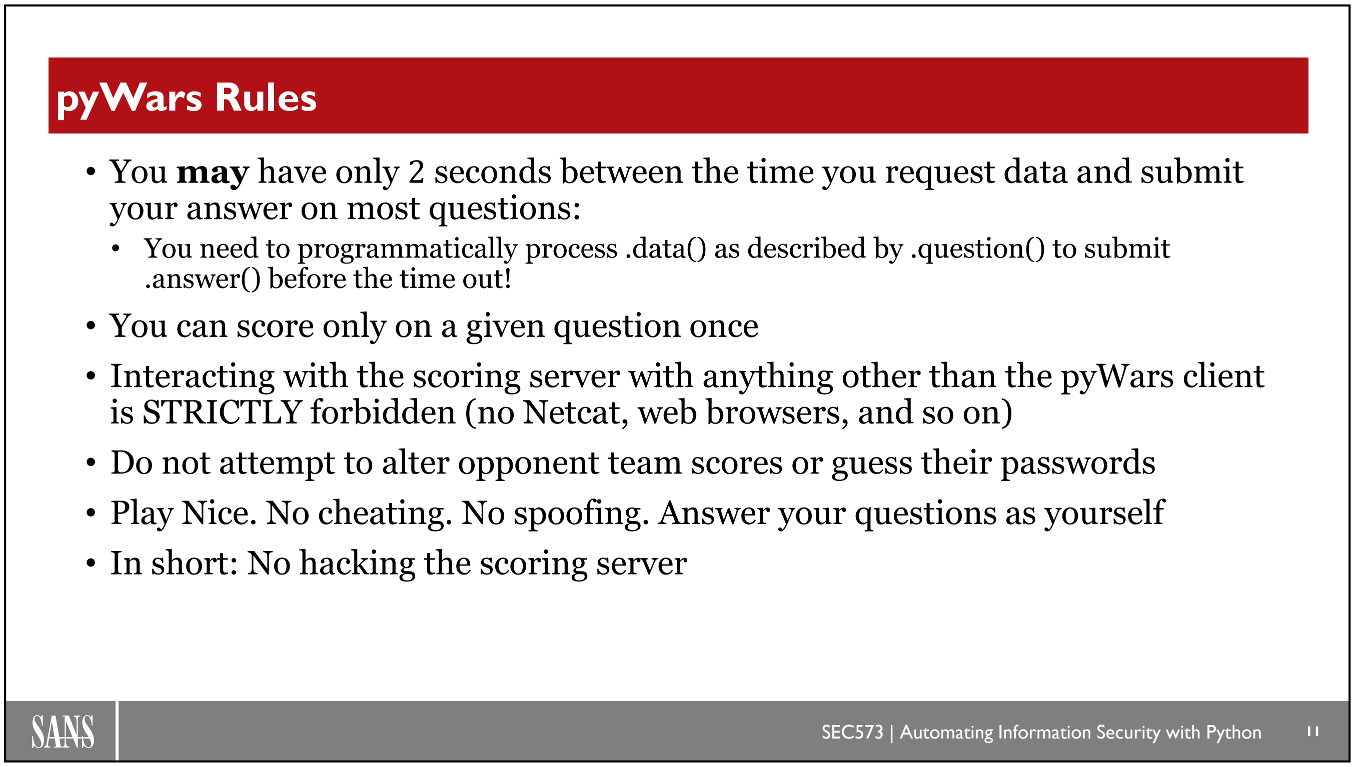
- Think strategically about how to accumulate the most points in the time provided
- Sometimes the result of your hack says flag=XYZ; if so, **only submit the portion after equals sign,** i.e., XYZ
- Don't alter the case or order of items unless told to

SANS | SEC573 | Automating Information Security with Python 10

You need to interact only with the other targets in the 10.10.10.0/24 range when you are directed to do so by a question. You do not need to scan the network to find these hosts.

The point value awarded for completing a question varies from question to question. The amount of points awarded is directly proportional to the difficulty of the question. There will be more easy questions than difficult questions, so you should think strategically about how to accumulate the most points in the time allotted.

Sometimes when you solve a challenge, the result will simply return the answer that you must submit to the pyWars server. Other times the page will say something like "flag=ABCDEF..." If the flag says "flag=<some answer>", then you should submit only the part after the equals sign. For example, if you solve the challenge and a webpage contains "flag=SUBMITME", then you should submit "SUBMITME" as the answer.



pyWars Rules

- You **may** have only 2 seconds between the time you request data and submit your answer on most questions:
  - You need to programmatically process .data() as described by .question() to submit .answer() before the time out!
- You can score only on a given question once
- Interacting with the scoring server with anything other than the pyWars client is STRICTLY forbidden (no Netcat, web browsers, and so on)
- Do not attempt to alter opponent team scores or guess their passwords
- Play Nice. No cheating. No spoofing. Answer your questions as yourself
- In short: No hacking the scoring server

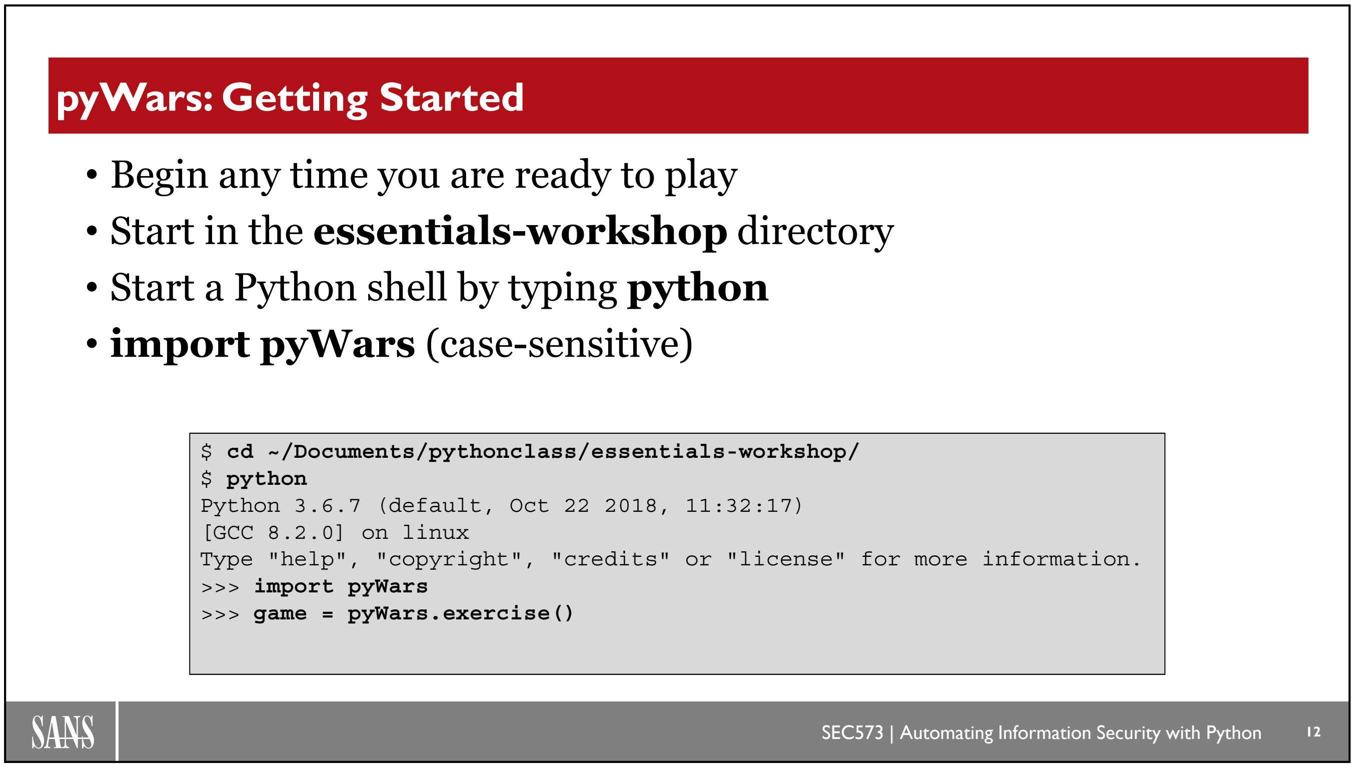
SANS SEC573 | Automating Information Security with Python 11

After you have queried the data associated with a question, by calling games.data(#), you have only 2 seconds to submit the correct answer. Some questions will not enforce this 2-second time out. If the time out is not enforced, the question will state that fact. This means that you will not have time to read the data, figure out the answer in your head, and then manually type the answer back in. You will need to automatically process the information returned by games.data(#) and automatically submit the answer.

You can submit each answer only once. Technically, you can submit an answer more than once, but you only get points for it once.

Do not interact with the scoring server with ANYTHING other than the pyWars client. NO NETCAT! NO WEB BROWSERS!

There are several additional rules, but it all comes down to doing the right thing. Don't hack anyone or my servers. Play the game using the tools provided and have fun.



pyWars: Getting Started

- Begin any time you are ready to play
- Start in the **essentials-workshop** directory
- Start a Python shell by typing **python**
- **import pyWars** (case-sensitive)

```
$ cd ~/Documents/pythonclass/essentials-workshop/
$ python
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyWars
>>> game = pyWars.exercise()
```

SANS SEC573 | Automating Information Security with Python 12

You can begin playing pyWars whenever you are ready. I'll introduce it to everyone now so that advanced programmers can jump right in. If some of this seems too advanced, that is okay. Just come back to this section when you are ready to begin playing.

Playing pyWars is simple. First, you need to check into the directory containing the pyWars module:

```
$ cd ~/Documents/pythonclass/essentials-workshop/
```

Then start Python by typing **python** and pressing **Enter**.

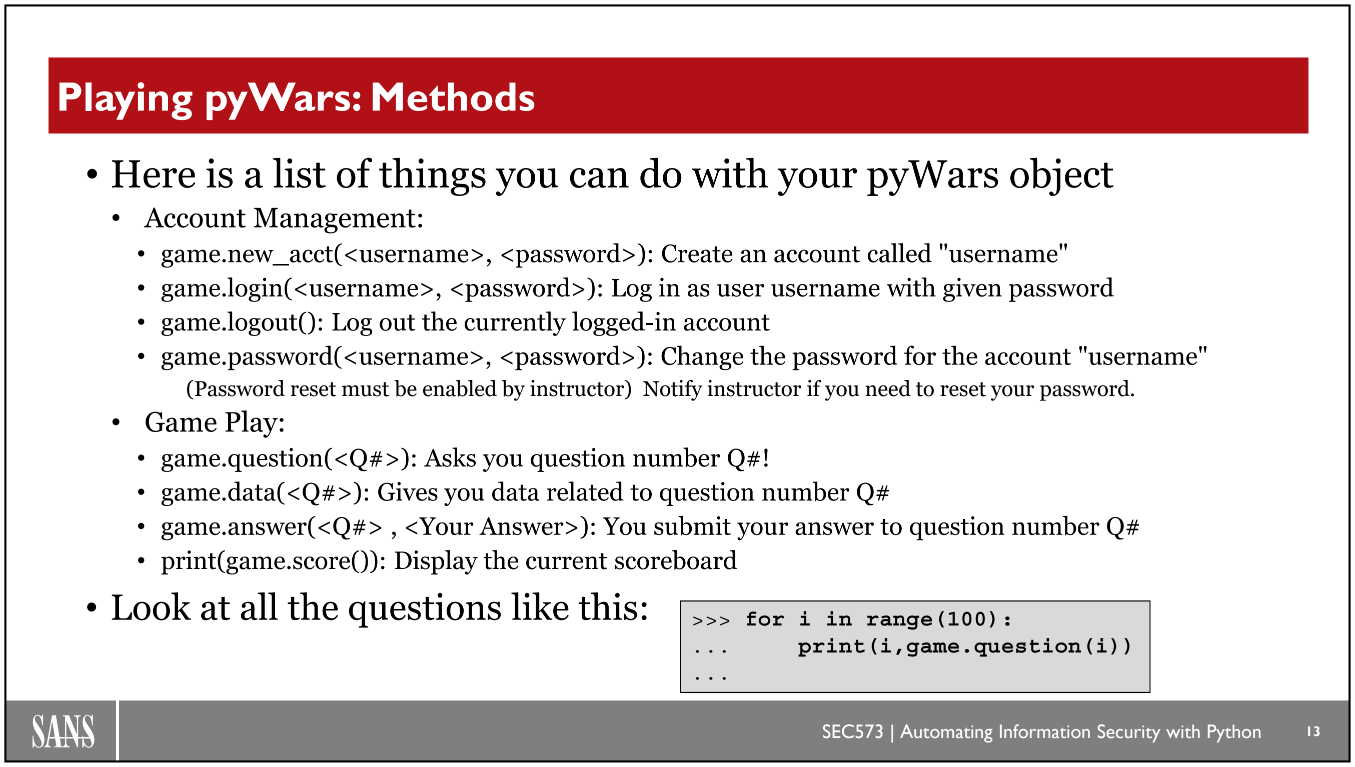
```
$ python
```

Python starts and you will get the interactive python prompt ">>>". Then import the pyWars module like this:

```
>>>  import pyWars
```

Last, we create a pyWars object in memory that we can use to interact with the server and store it in a variable. In this case, the variable is named "game", but it can be any name that you like!

```
>>>  game = pyWars.exercise()
```



Playing pyWars: Methods

- Here is a list of things you can do with your pyWars object
  - Account Management:
    - game.new_acct(<username>, <password>): Create an account called "username"
    - game.login(<username>, <password>): Log in as user username with given password
    - game.logout(): Log out the currently logged-in account
    - game.password(<username>, <password>): Change the password for the account "username"
      (Password reset must be enabled by instructor) Notify instructor if you need to reset your password.
  - Game Play:
    - game.question(<Q#>): Asks you question number Q#!
    - game.data(<Q#>): Gives you data related to question number Q#
    - game.answer(<Q#> , <Your Answer>): You submit your answer to question number Q#
    - print(game.score()): Display the current scoreboard
- Look at all the questions like this:

```
>>> for i in range(100):
...     print(i,game.question(i))
...
```

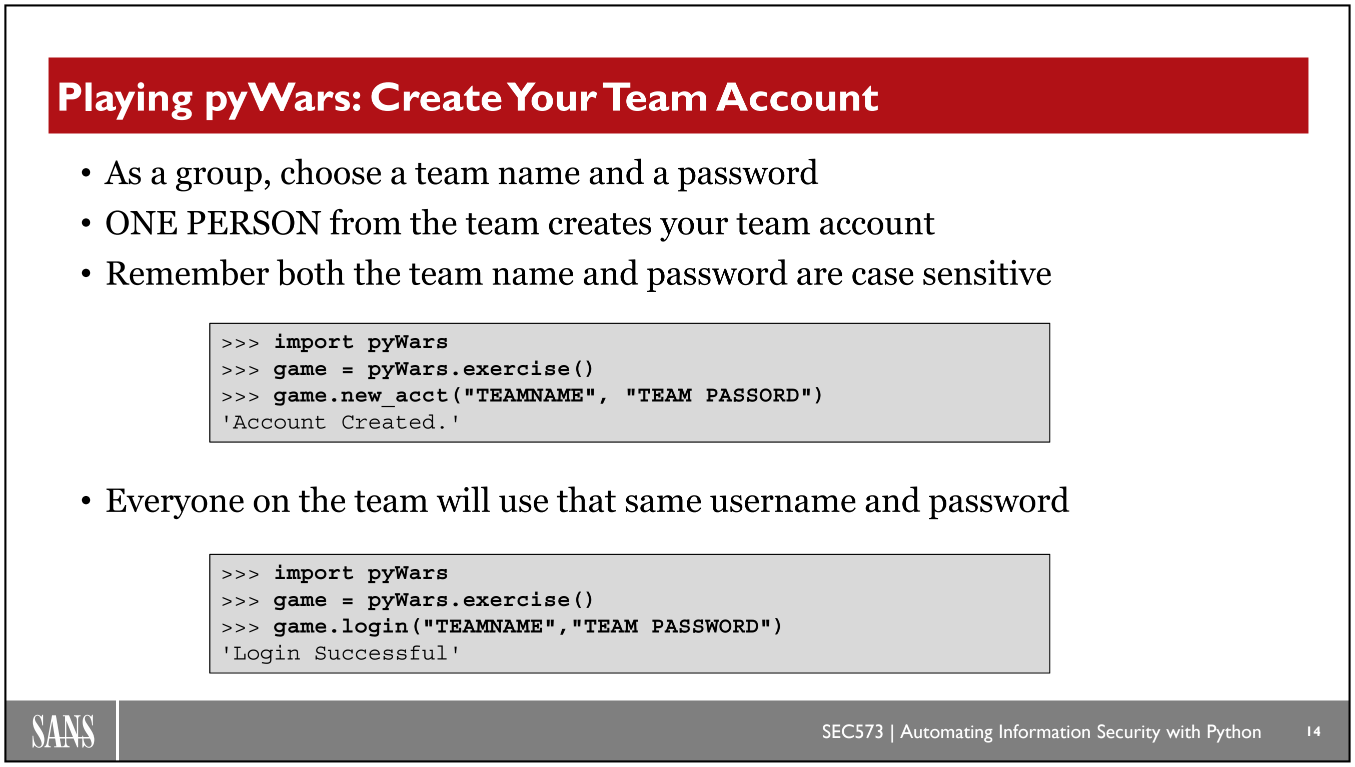
SANS SEC573 | Automating Information Security with Python 13

Once you have a pyWars exercise object in memory, you can call its various methods to perform actions against the pyWars server. For example, you will use .new_acct(), .login(), .logout(), and, if necessary, .password() to manage your account on the server. You will use new_acct() once to create an account for yourself on the server. Once your account is created, you do not need to use this method again. Then you can use login() and logout() to use that account.

Once you have a logged-in session, you can call .question(), .data(), .answer(), and .score() to interact with the server. Question takes in a question number and gives you back the text for that question. Every question will ask you to manipulate some data in some way. To get the associated data, you call .data and give it the question number that you want the data for. After you have manipulated the data, you submit it back to the server as your answer. The answer() method takes to arguments separated by a comma. The first is the question number you are submitting an answer to, and the second is your answer containing the manipulated data.

You can also print the score to see how you are doing by executing the command 'print('game.score())'.

If you would like to see a complete list of all the questions, I provide you with a 'for loop' here that you can use. Don’t be concerned about not understanding that command yet. We will discuss for loops in detail later.



Playing pyWars: Create Your Team Account

- As a group, choose a team name and a password
- ONE PERSON from the team creates your team account
- Remember both the team name and password are case sensitive

```
>>> import pyWars
>>> game = pyWars.exercise()
>>> game.new_acct("TEAMNAME", "TEAM PASSORD")
'Account Created.'
```

- Everyone on the team will use that same username and password

```
>>> import pyWars
>>> game = pyWars.exercise()
>>> game.login("TEAMNAME","TEAM PASSWORD")
'Login Successful'
```

SANS SEC573 | Automating Information Security with Python 14

After importing the module, you will create an instance of a pyWars Exercise object. Then create an account for the TEAM

```
>>> game=pyWars.exercise()
```

This creates a new variable named "game" that will hold our pyWars object that we can use to interact with the server. Now we can use that object to create a new account on the server for you to use.

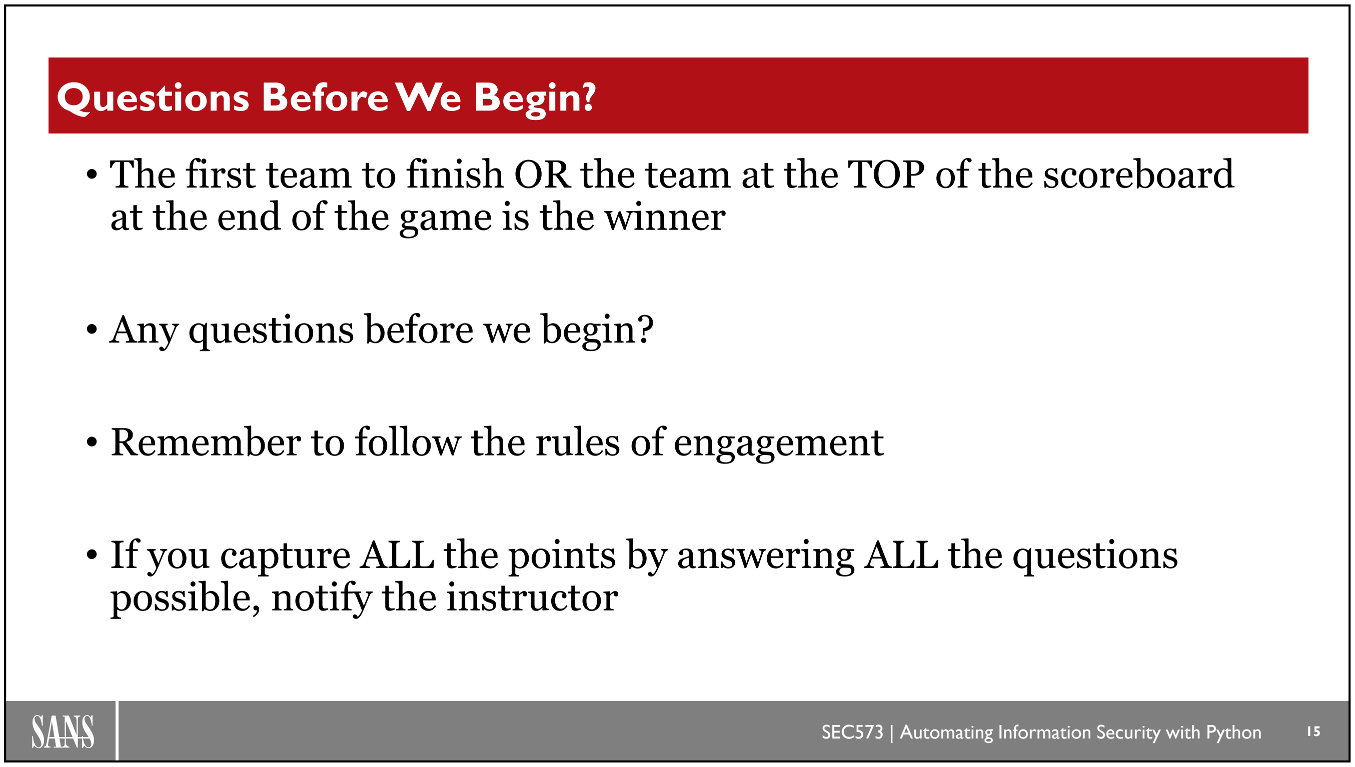
```
>>> game.new_acct("TEAMNAME", "TEAM PASSWORD")
```

Now everyone on your team can use that username and password to log in to the server and play. To log in, you call the .login() method and pass your username and password.

```
>>> game.login("username","password")
```

That's it! You are ready to play! You can also logout() of the server when you're not using it. That is always a good security practice. Additionally, if you forget your password, you can use .password to choose another password, but this feature can only be used after the instructor flags your account for password reset on the server.

It is worth noting that when you call either new_acct() or login(), the client remembers the username and password you used so you can just use login() without passing any username and password from that point forward. This provides you with a little more protection from shoulder surfing classmates.



Questions Before We Begin?

- The first team to finish OR the team at the TOP of the scoreboard at the end of the game is the winner
- Any questions before we begin?
- Remember to follow the rules of engagement
- If you capture ALL the points by answering ALL the questions possible, notify the instructor

Are there any questions before we begin?

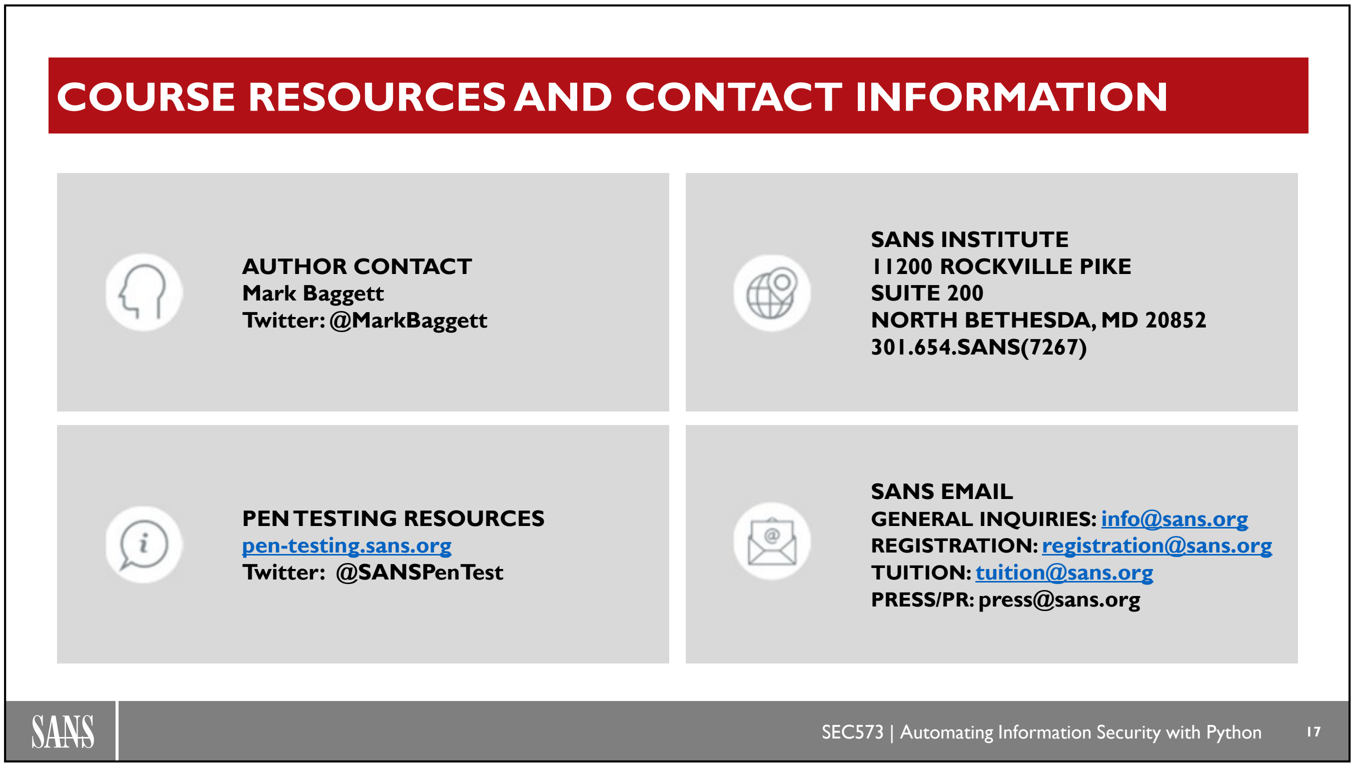
You Have Permission to Begin

GO!

SANS

SEC573 | Automating Information Security with Python 16

You may begin.



COURSE RESOURCES AND CONTACT INFORMATION

AUTHOR CONTACT
Mark Baggett
Twitter: @MarkBaggett

SANS INSTITUTE
11200 ROCKVILLE PIKE
SUITE 200
NORTH BETHESDA, MD 20852
301.654.SANS(7267)

PEN TESTING RESOURCES
pen-testing.sans.org
Twitter: @SANSPenTest

SANS EMAIL
GENERAL INQUIRIES: info@sans.org
REGISTRATION: registration@sans.org
TUITION: tuition@sans.org
PRESS/PR: press@sans.org

SANS | SEC573 | Automating Information Security with Python 17

This page intentionally left blank.

This page intentionally left blank.

Index