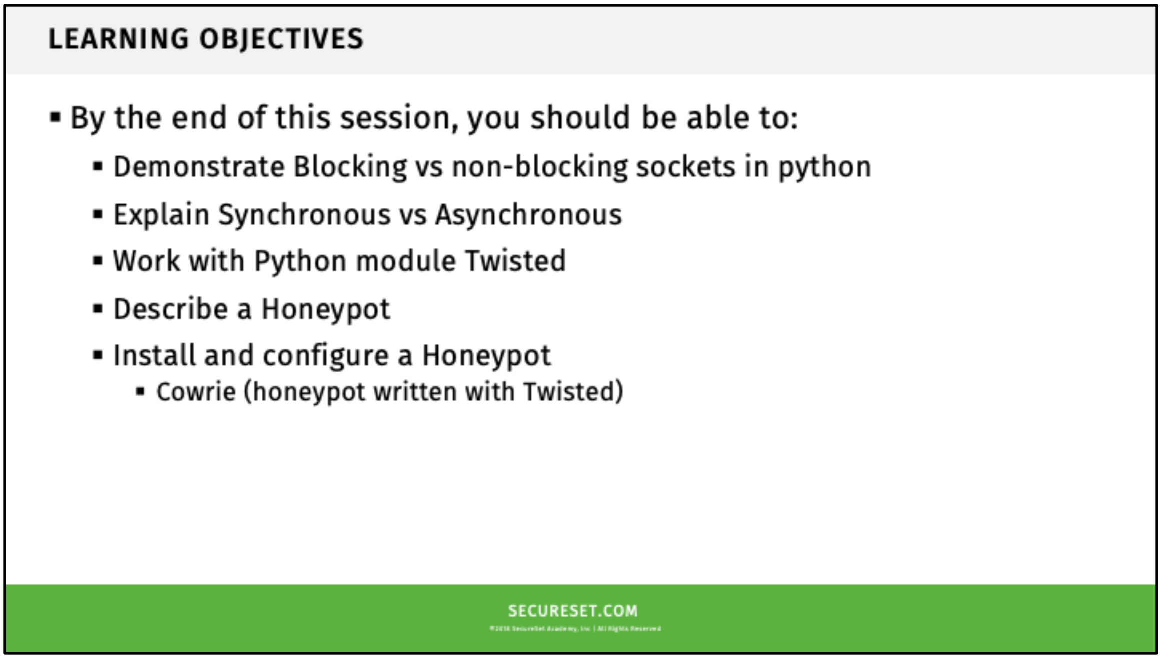
CORE

AN IMMERSIVE CYBERSECURITY
ENGINEERING PROGRAM

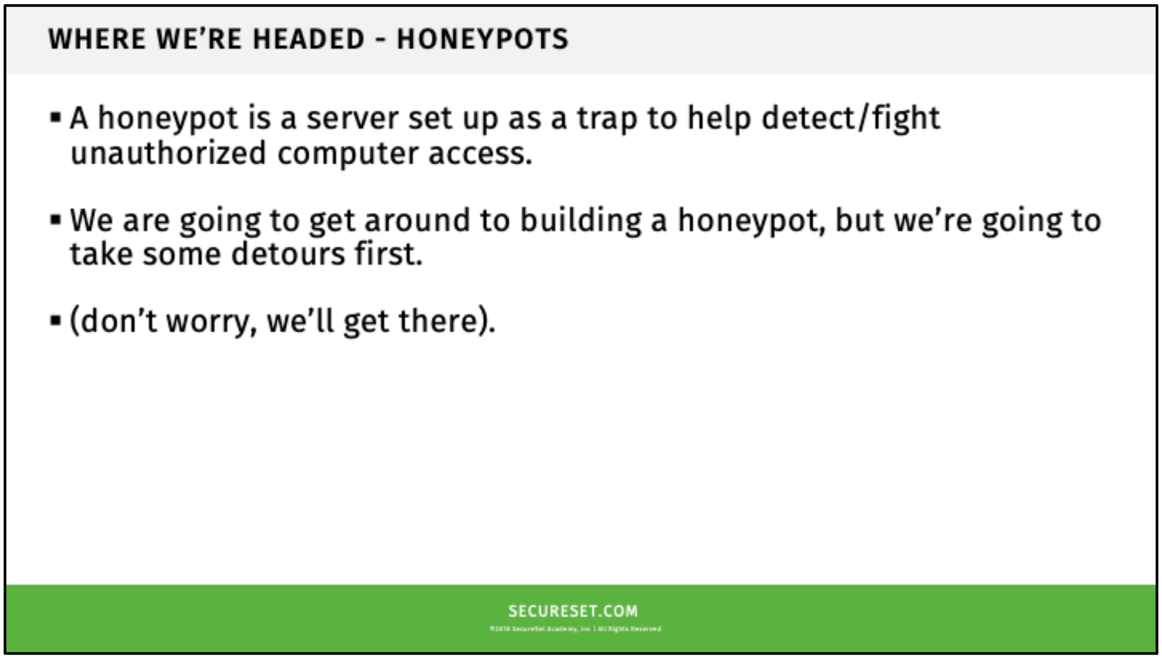
CORE NETWORK 500

Multithreaded Networking & Honeypots



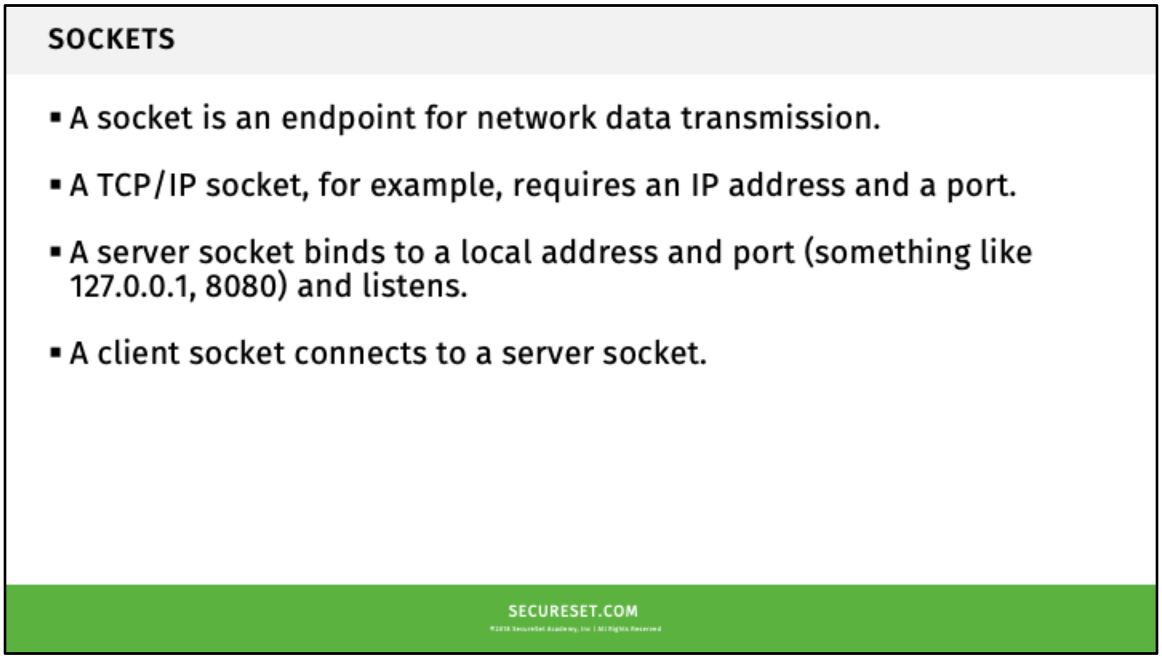
LEARNING OBJECTIVES

- By the end of this session, you should be able to:
  - Demonstrate Blocking vs non-blocking sockets in python
  - Explain Synchronous vs Asynchronous
  - Work with Python module Twisted
  - Describe a Honeypot
  - Install and configure a Honeypot
    - Cowrie (honeypot written with Twisted)



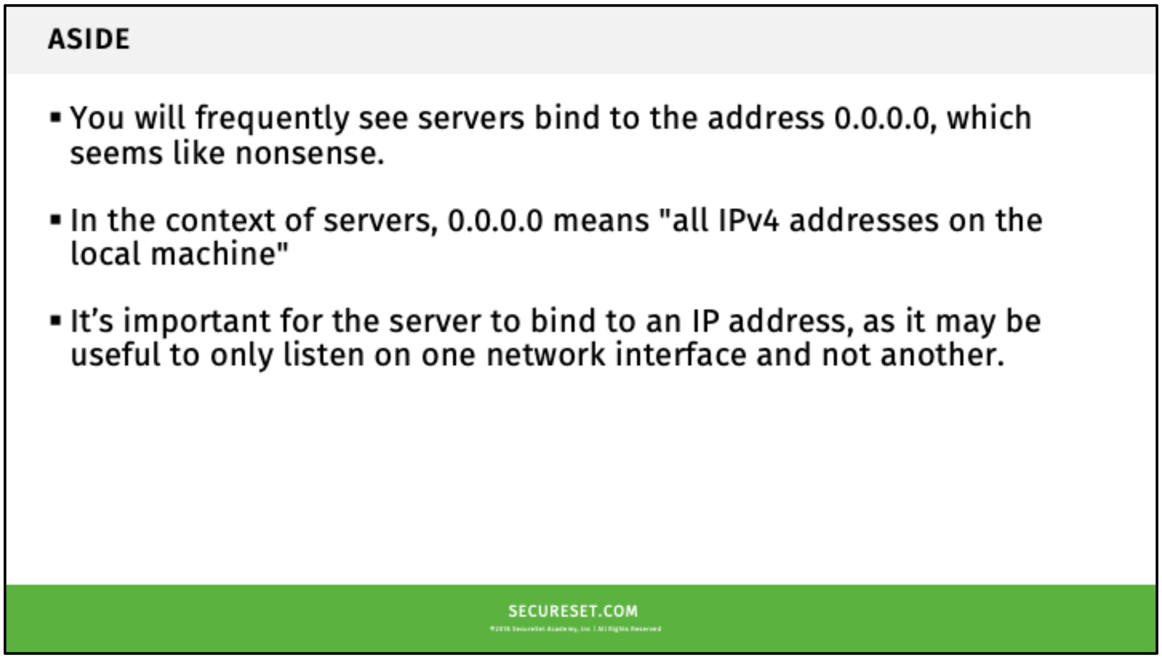
WHERE WE’RE HEADED - HONEYPOTS

- A honeypot is a server set up as a trap to help detect/fight unauthorized computer access.
- We are going to get around to building a honeypot, but we’re going to take some detours first.
- (don’t worry, we’ll get there).



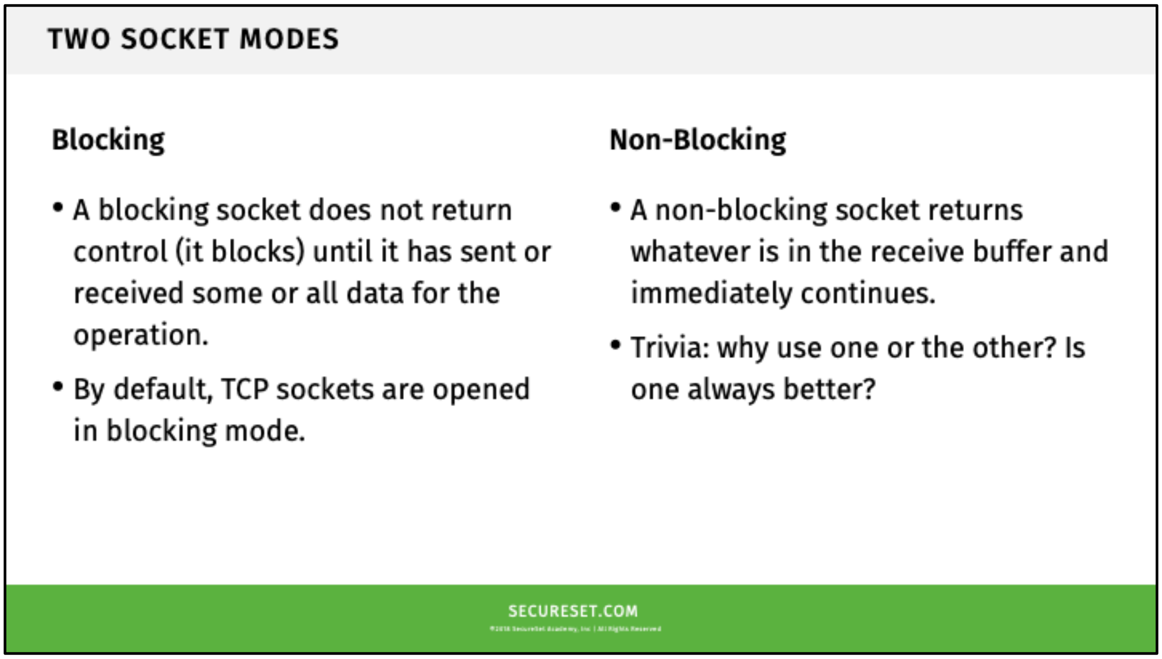
SOCKETS

- A socket is an endpoint for network data transmission.
- A TCP/IP socket, for example, requires an IP address and a port.
- A server socket binds to a local address and port (something like 127.0.0.1, 8080) and listens.
- A client socket connects to a server socket.



ASIDE

- You will frequently see servers bind to the address 0.0.0.0, which seems like nonsense.
- In the context of servers, 0.0.0.0 means "all IPv4 addresses on the local machine"
- It's important for the server to bind to an IP address, as it may be useful to only listen on one network interface and not another.



TWO SOCKET MODES

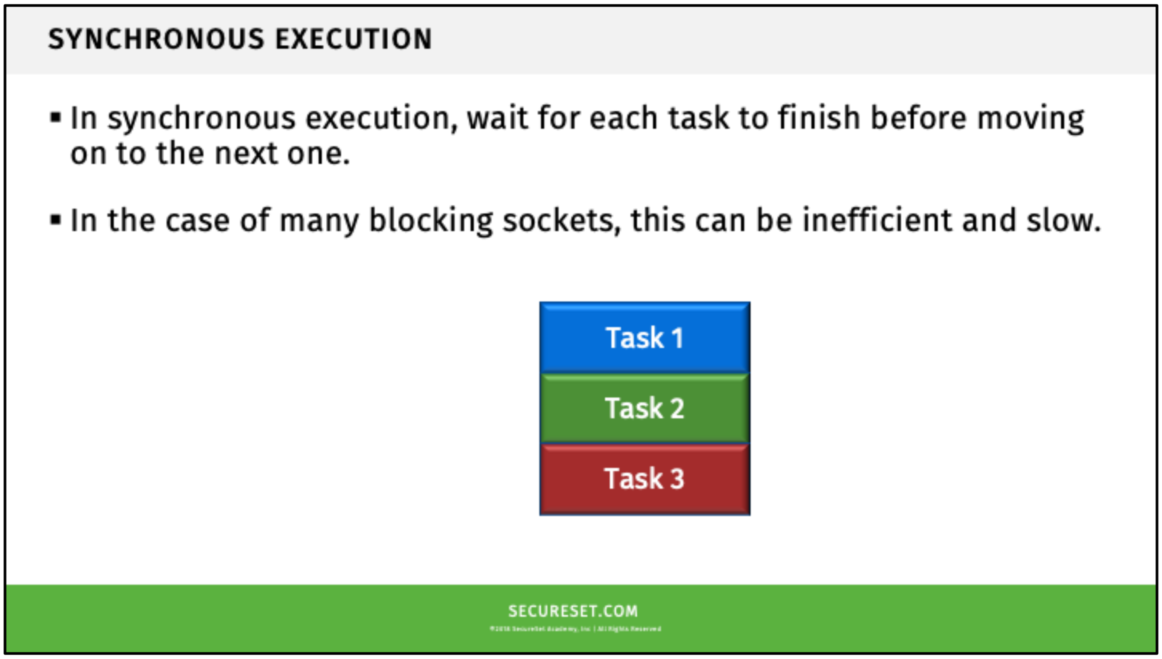
Blocking

- A blocking socket does not return control (it blocks) until it has sent or received some or all data for the operation.
- By default, TCP sockets are opened in blocking mode.

Non-Blocking

- A non-blocking socket returns whatever is in the receive buffer and immediately continues.
- Trivia: why use one or the other? Is one always better?

SECURESET.COM



SYNCHRONOUS EXECUTION

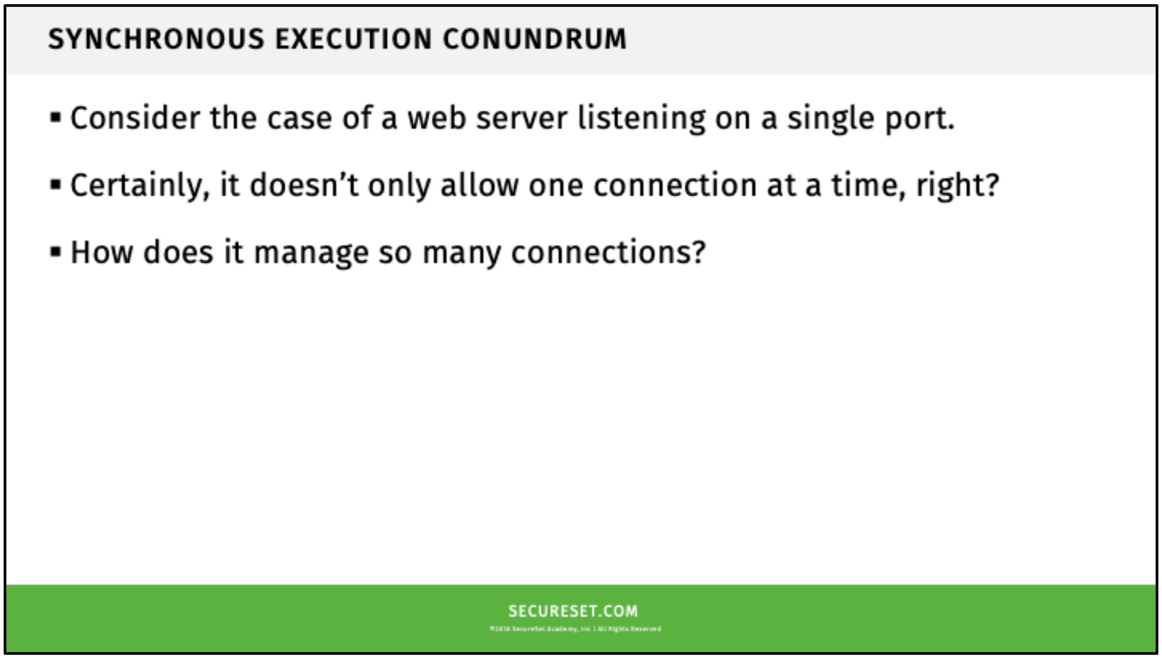
- In synchronous execution, wait for each task to finish before moving on to the next one.
- In the case of many blocking sockets, this can be inefficient and slow.

Task 1
Task 2
Task 3

SECURESET.COM

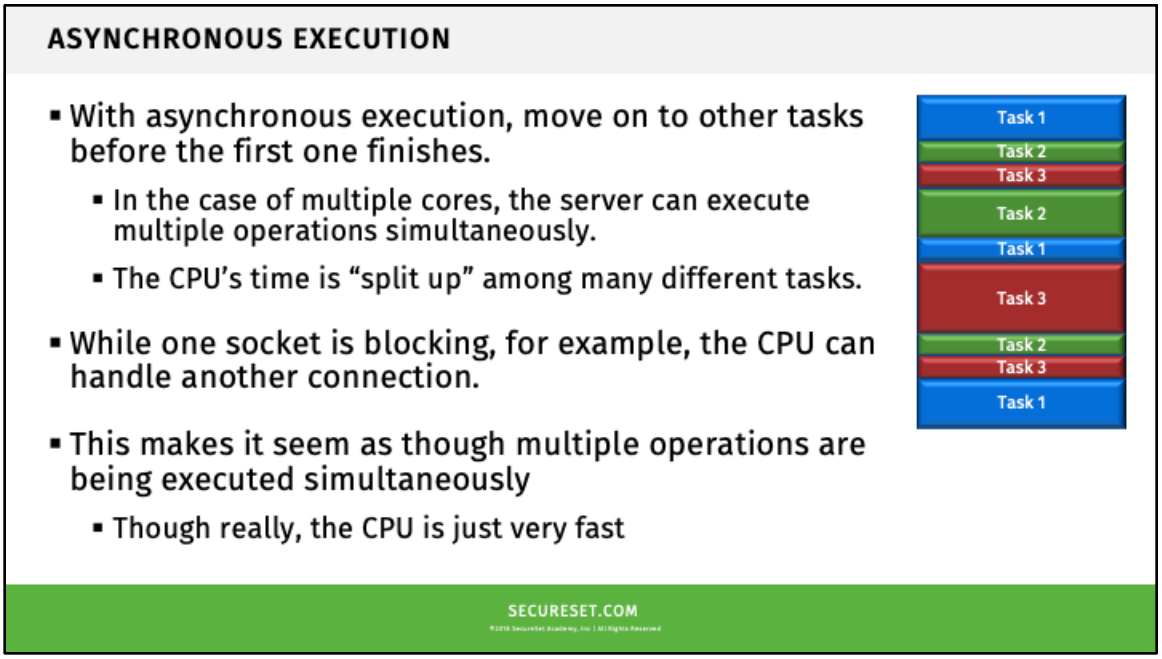
If Task one stops to wait on a condition to occur before moving on, Task 2 is, effectively, also waiting for this condition, since it cannot start until Task 1 end. Task 3 is dependent on Task 2, etc.

Batch scripts typically operate in this mode.



SYNCHRONOUS EXECUTION CONUNDRUM

- Consider the case of a web server listening on a single port.
- Certainly, it doesn’t only allow one connection at a time, right?
- How does it manage so many connections?

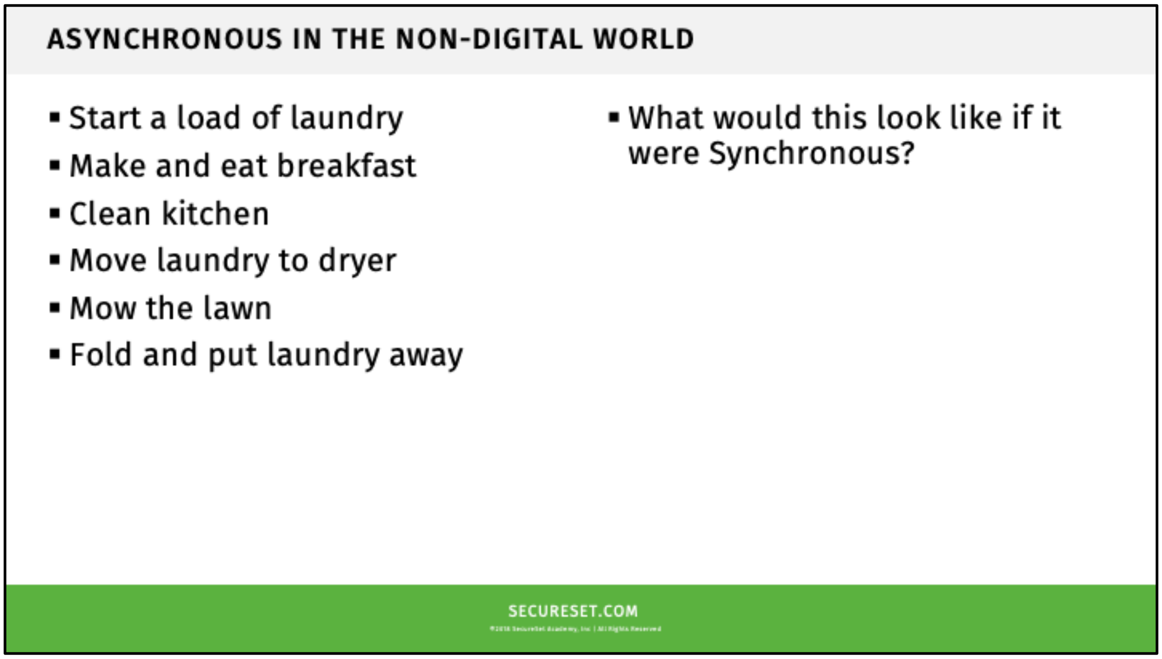


ASYNCHRONOUS EXECUTION

- With asynchronous execution, move on to other tasks before the first one finishes.
  - In the case of multiple cores, the server can execute multiple operations simultaneously.
  - The CPU’s time is “split up” among many different tasks.
- While one socket is blocking, for example, the CPU can handle another connection.
- This makes it seem as though multiple operations are being executed simultaneously
  - Though really, the CPU is just very fast

SECURESET.COM

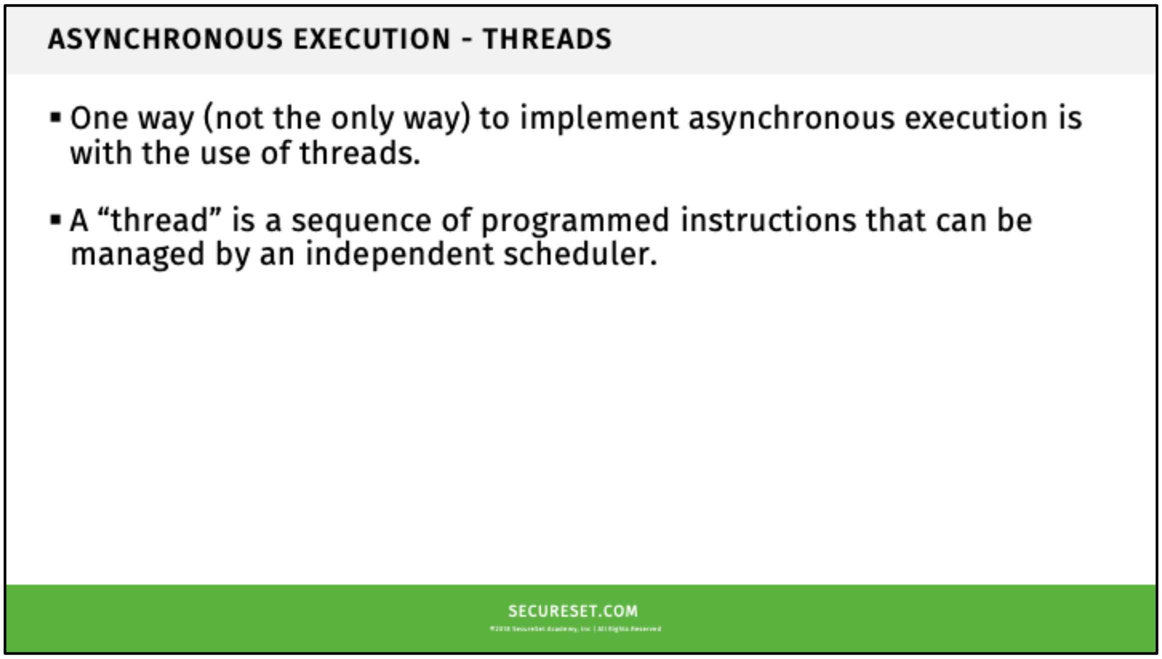
This diagram is a single thread, asynchronous visualization



ASYNCHRONOUS IN THE NON-DIGITAL WORLD

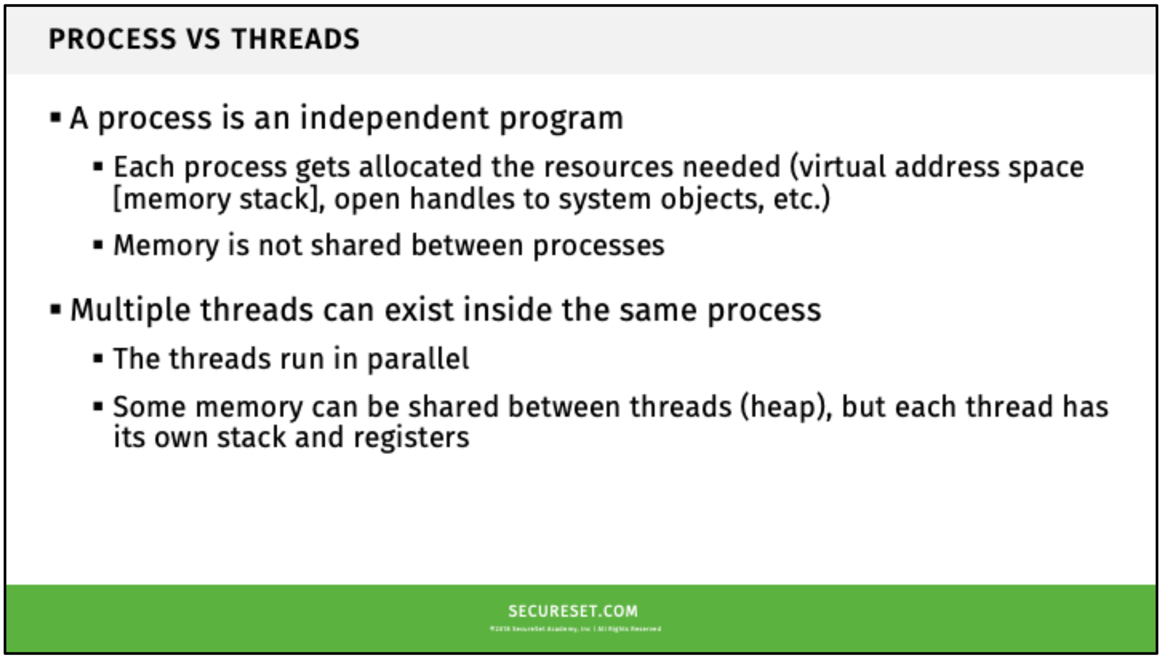
- Start a load of laundry
- Make and eat breakfast
- Clean kitchen
- Move laundry to dryer
- Mow the lawn
- Fold and put laundry away

- What would this look like if it were Synchronous?



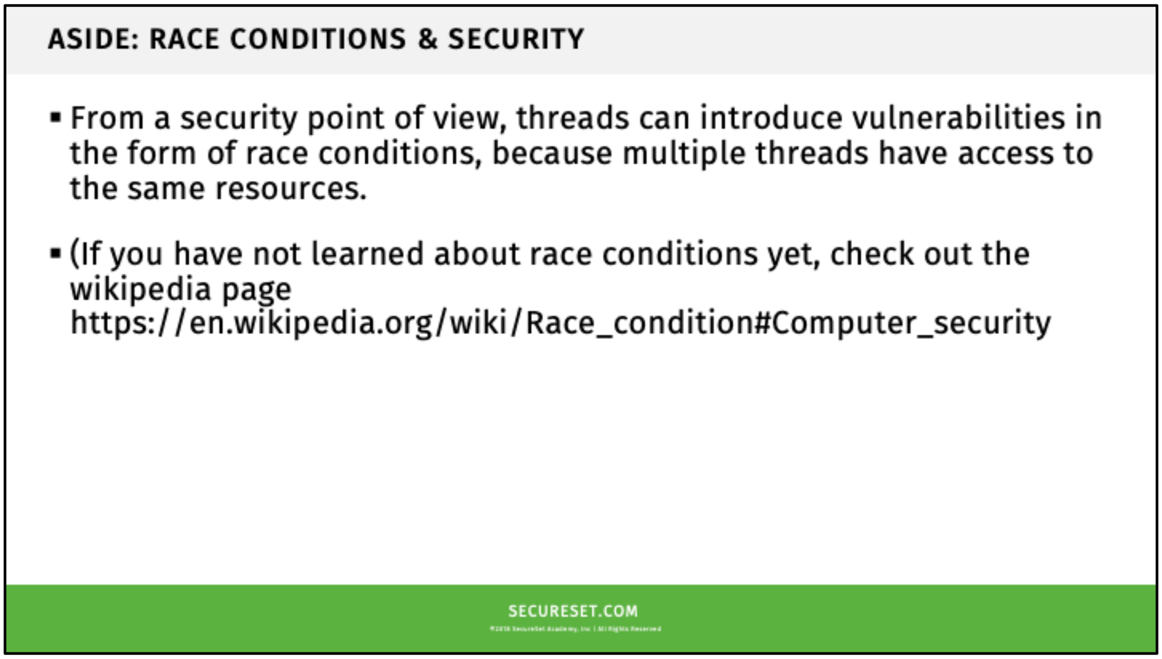
ASYNCHRONOUS EXECUTION - THREADS

- One way (not the only way) to implement asynchronous execution is with the use of threads.
- A “thread” is a sequence of programmed instructions that can be managed by an independent scheduler.



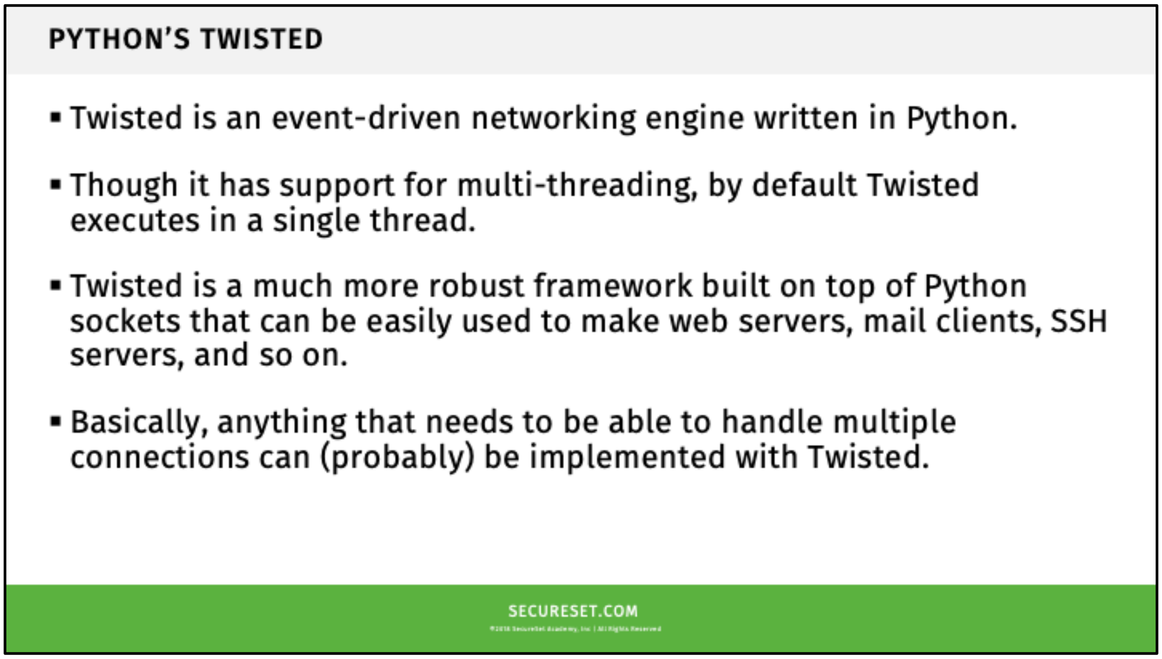
PROCESS VS THREADS

- A process is an independent program
  - Each process gets allocated the resources needed (virtual address space [memory stack], open handles to system objects, etc.)
  - Memory is not shared between processes
- Multiple threads can exist inside the same process
  - The threads run in parallel
  - Some memory can be shared between threads (heap), but each thread has its own stack and registers



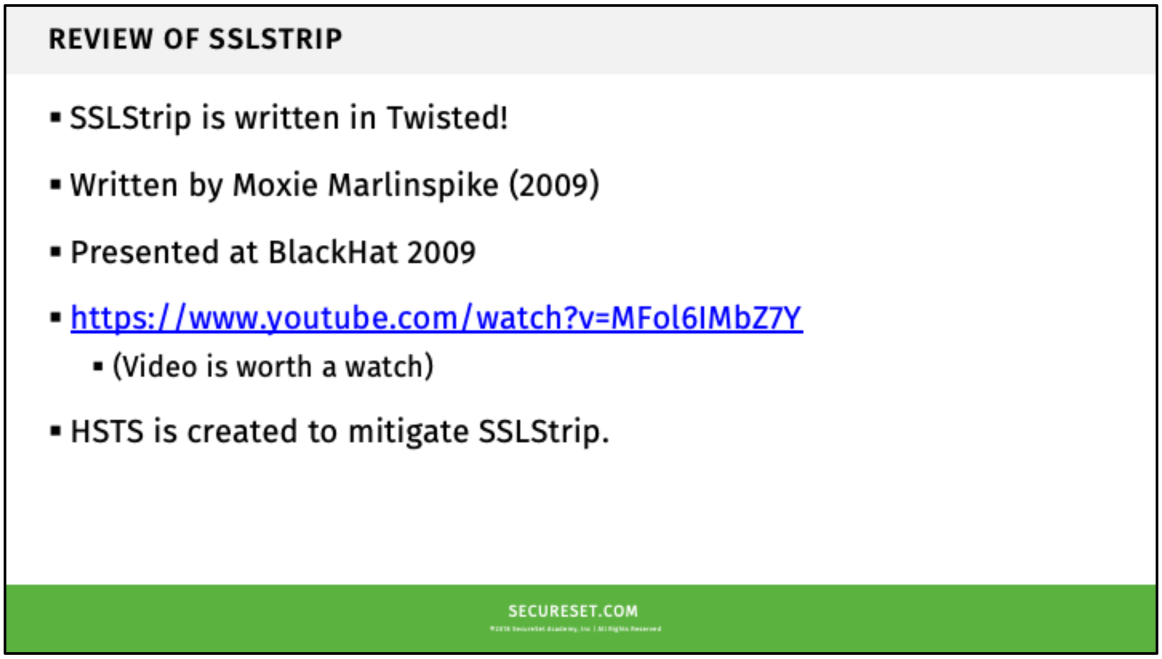
ASIDE: RACE CONDITIONS & SECURITY

- From a security point of view, threads can introduce vulnerabilities in the form of race conditions, because multiple threads have access to the same resources.
- (If you have not learned about race conditions yet, check out the wikipedia page https://en.wikipedia.org/wiki/Race_condition#Computer_security



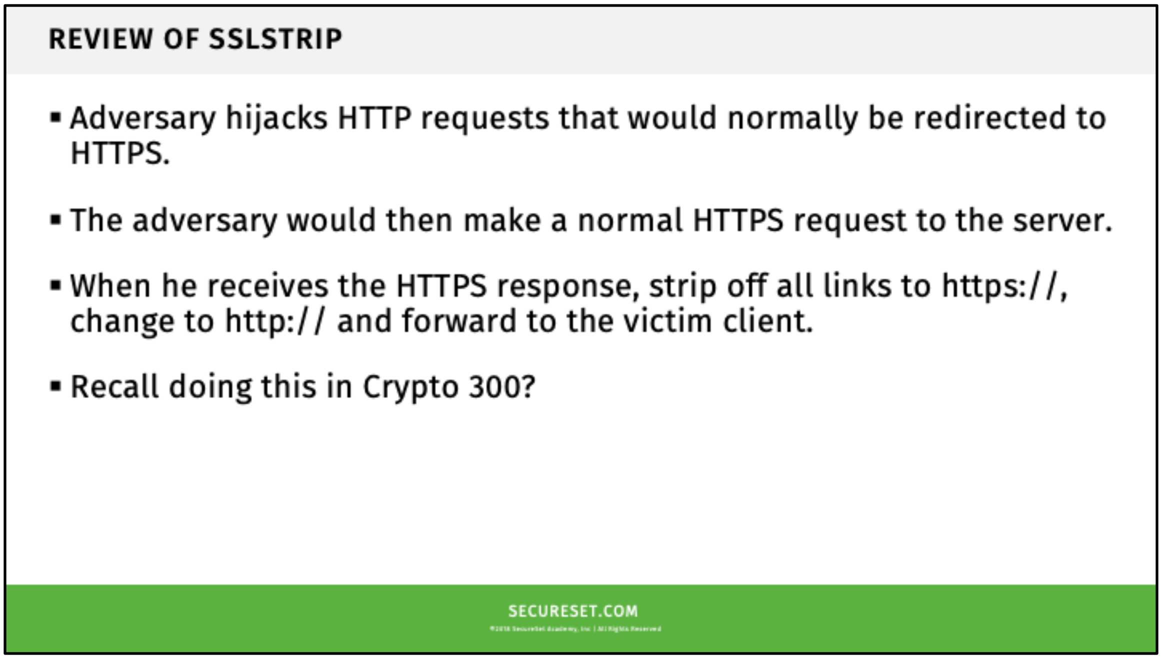
PYTHON'S TWISTED

- Twisted is an event-driven networking engine written in Python.
- Though it has support for multi-threading, by default Twisted executes in a single thread.
- Twisted is a much more robust framework built on top of Python sockets that can be easily used to make web servers, mail clients, SSH servers, and so on.
- Basically, anything that needs to be able to handle multiple connections can (probably) be implemented with Twisted.



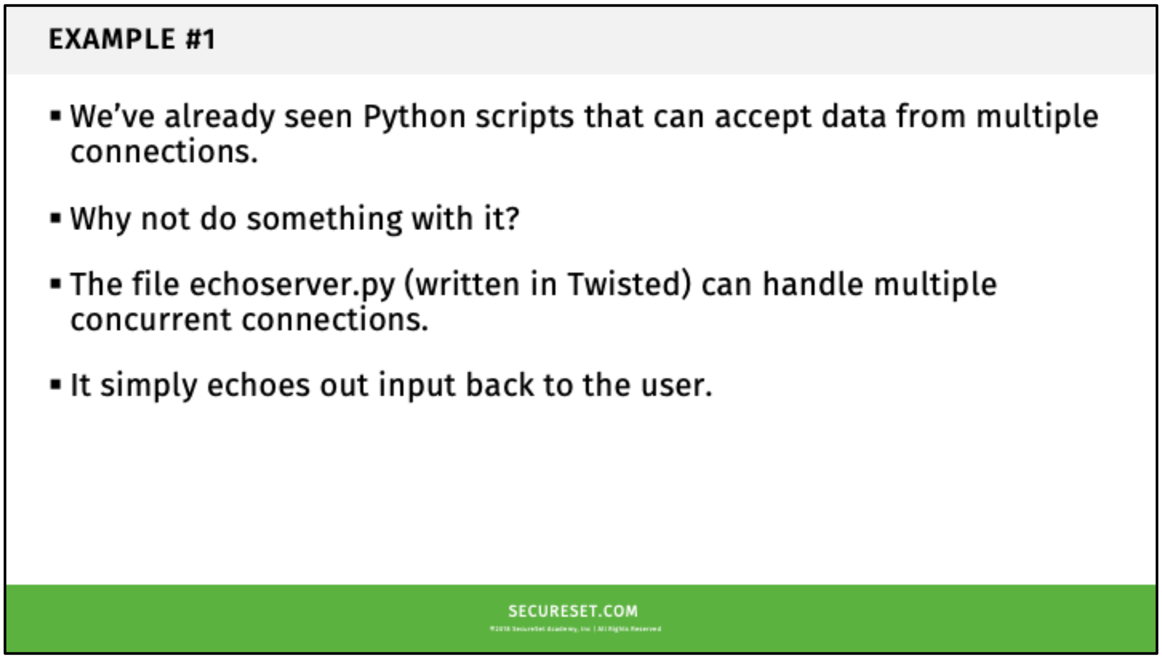
REVIEW OF SSLSTRIP

- SSLStrip is written in Twisted!
- Written by Moxie Marlinspike (2009)
- Presented at BlackHat 2009
- https://www.youtube.com/watch?v=MFol6IMbZ7Y
  - (Video is worth a watch)
- HSTS is created to mitigate SSLStrip.



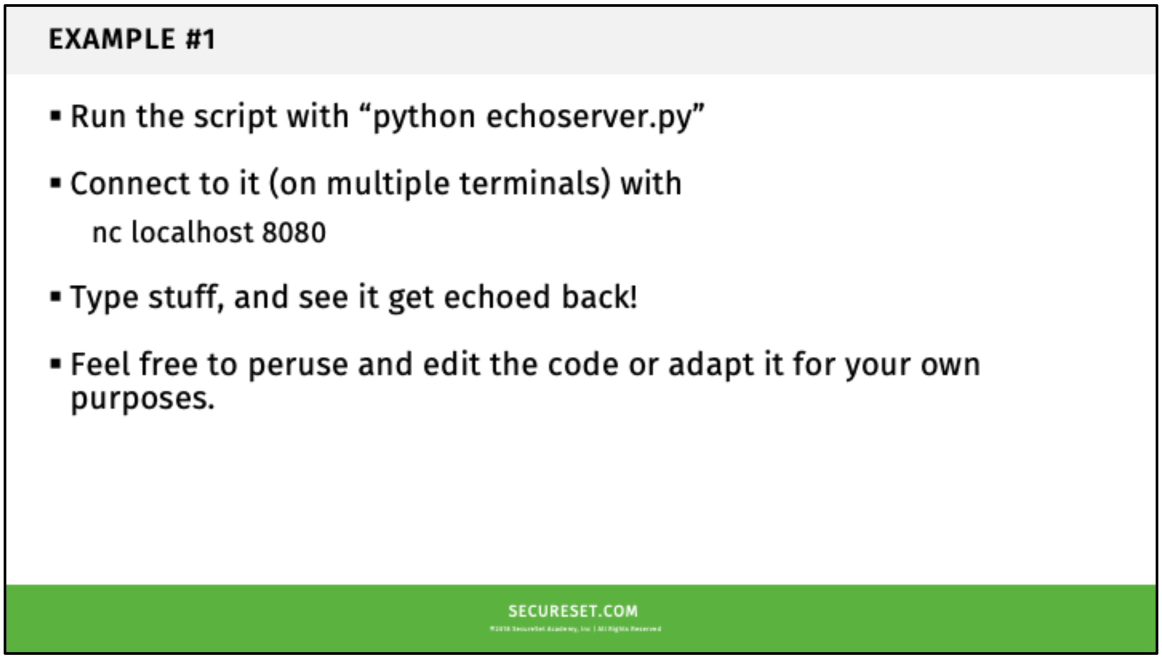
REVIEW OF SSLSTRIP

- Adversary hijacks HTTP requests that would normally be redirected to HTTPS.
- The adversary would then make a normal HTTPS request to the server.
- When he receives the HTTPS response, strip off all links to https://, change to http:// and forward to the victim client.
- Recall doing this in Crypto 300?



EXAMPLE #1

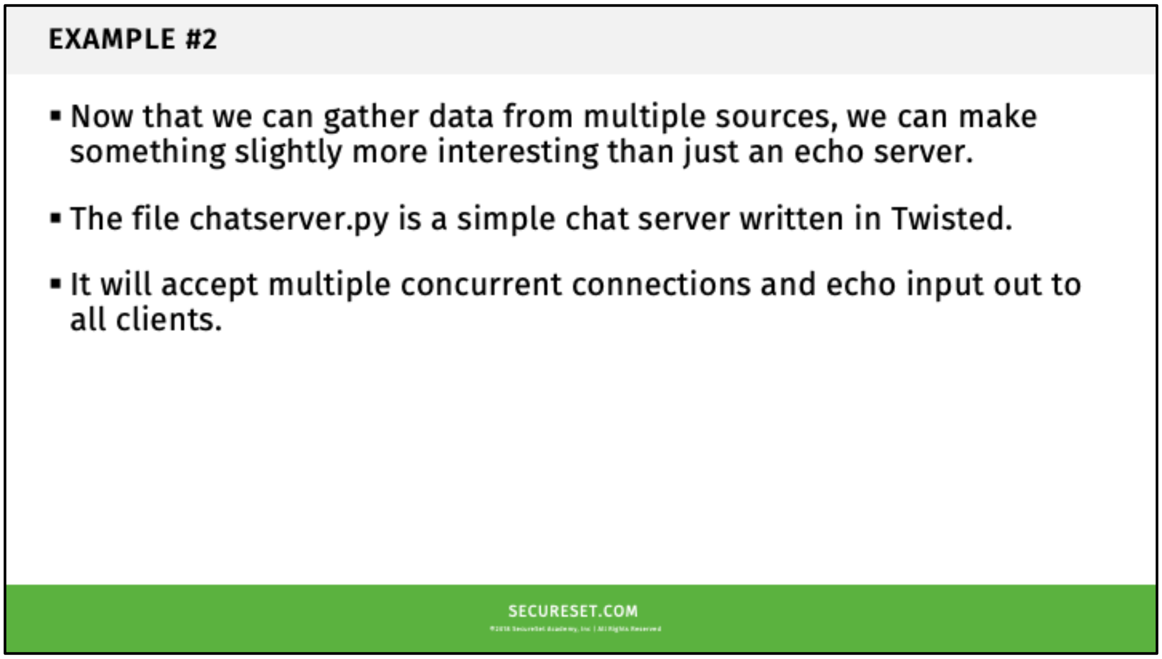
- We've already seen Python scripts that can accept data from multiple connections.
- Why not do something with it?
- The file echoserver.py (written in Twisted) can handle multiple concurrent connections.
- It simply echoes out input back to the user.



EXAMPLE #1

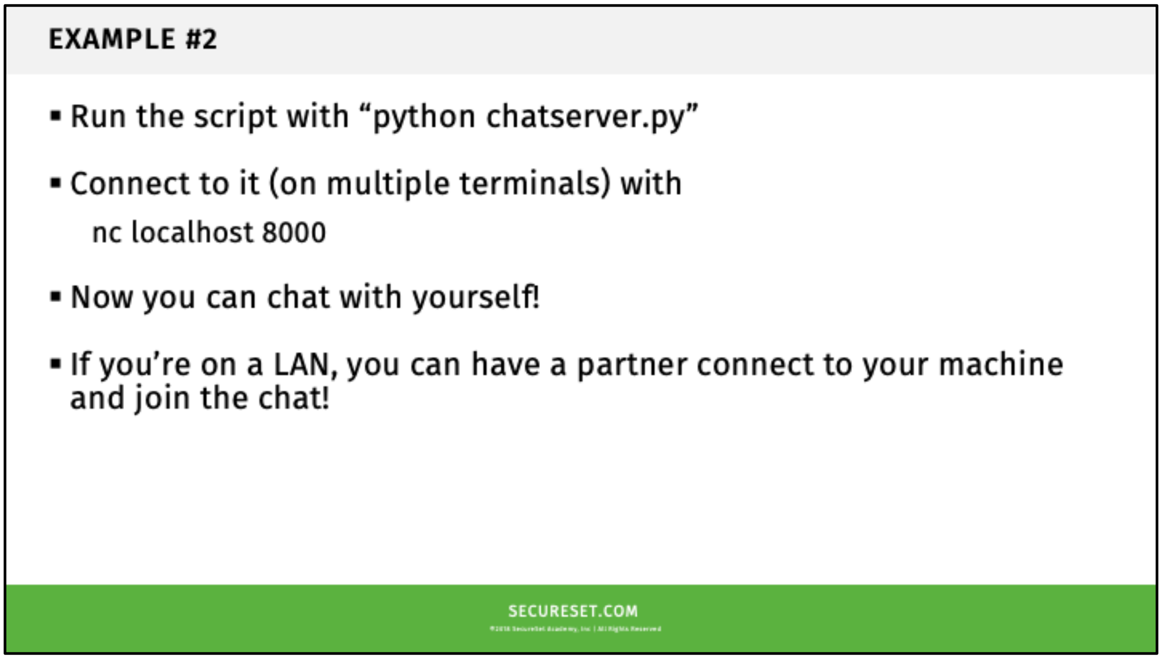
- Run the script with “python echoserver.py”
- Connect to it (on multiple terminals) with

  ```
  nc localhost 8080
  ```
- Type stuff, and see it get echoed back!
- Feel free to peruse and edit the code or adapt it for your own purposes.



EXAMPLE #2

- Now that we can gather data from multiple sources, we can make something slightly more interesting than just an echo server.
- The file chatserver.py is a simple chat server written in Twisted.
- It will accept multiple concurrent connections and echo input out to all clients.

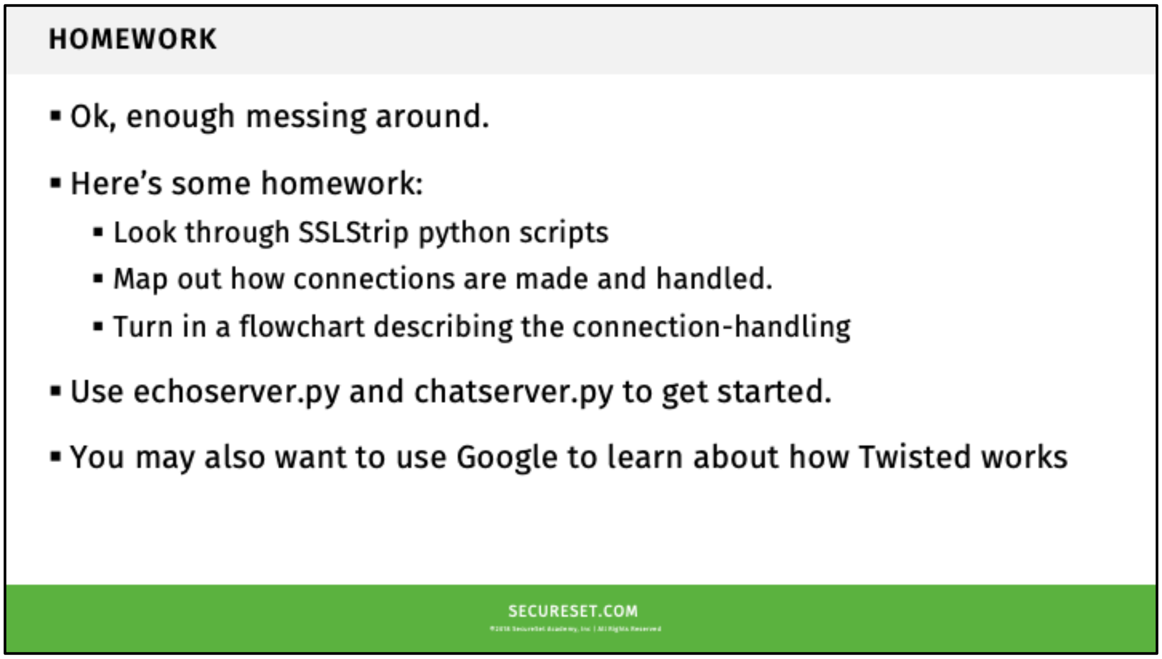


EXAMPLE #2

- Run the script with “python chatserver.py”
- Connect to it (on multiple terminals) with

```
nc localhost 8000
```

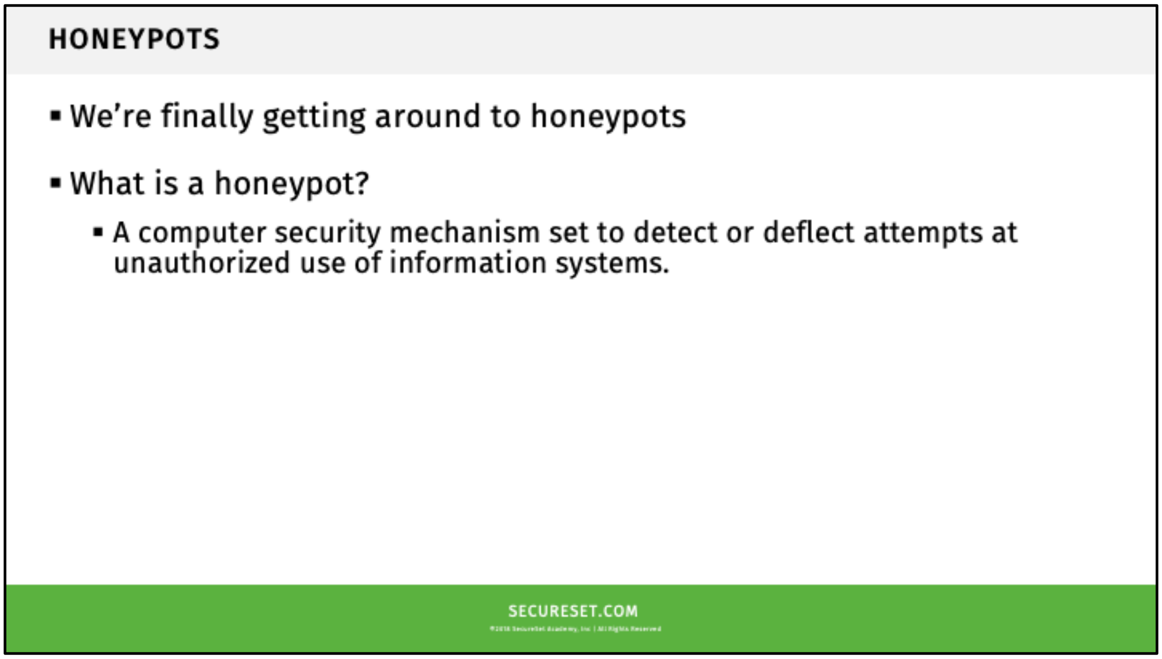
- Now you can chat with yourself!
- If you’re on a LAN, you can have a partner connect to your machine and join the chat!



HOMEWORK

- Ok, enough messing around.
- Here’s some homework:
  - Look through SSLStrip python scripts
  - Map out how connections are made and handled.
  - Turn in a flowchart describing the connection-handling
- Use echoserver.py and chatserver.py to get started.
- You may also want to use Google to learn about how Twisted works

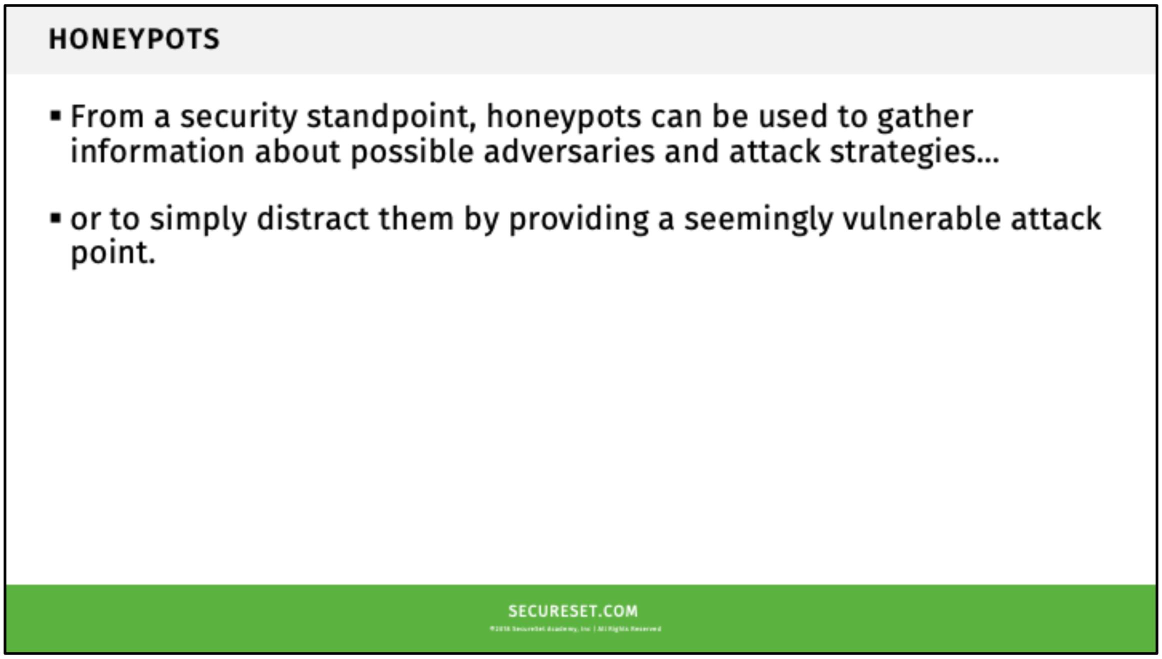
HONEYPOTS!



HONEYPOTS

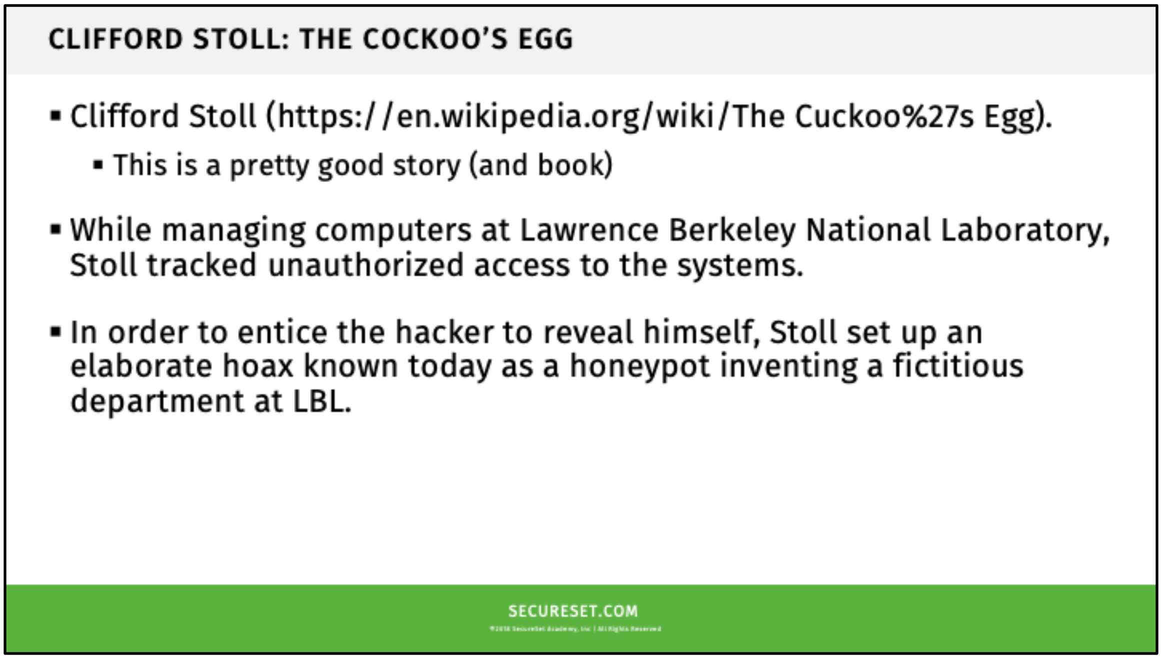
- We’re finally getting around to honeypots
- What is a honeypot?
  - A computer security mechanism set to detect or deflect attempts at unauthorized use of information systems.

SECURESET.COM



HONEYPOTS

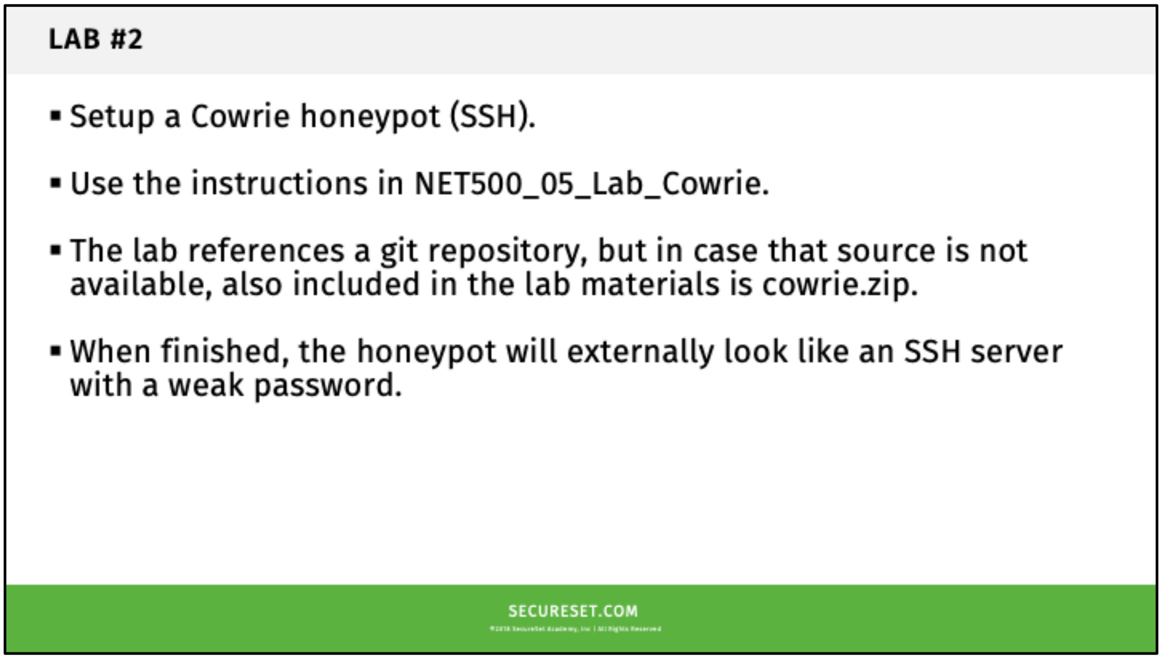
- From a security standpoint, honeypots can be used to gather information about possible adversaries and attack strategies…
- or to simply distract them by providing a seemingly vulnerable attack point.



CLIFFORD STOLL: THE COCKOO’S EGG

- Clifford Stoll (https://en.wikipedia.org/wiki/The Cuckoo%27s Egg).
  - This is a pretty good story (and book)
- While managing computers at Lawrence Berkeley National Laboratory, Stoll tracked unauthorized access to the systems.
- In order to entice the hacker to reveal himself, Stoll set up an elaborate hoax known today as a honeypot inventing a fictitious department at LBL.

LBL = Lawrence Berkeley Lab



LAB #2

- Setup a Cowrie honeypot (SSH).
- Use the instructions in NET500_05_Lab_Cowrie.
- The lab references a git repository, but in case that source is not available, also included in the lab materials is cowrie.zip.
- When finished, the honeypot will externally look like an SSH server with a weak password.