Recreating NetCat with Python3

Jake Clary and Seth Kyker

What Is NetCat?

The network swiss army knife

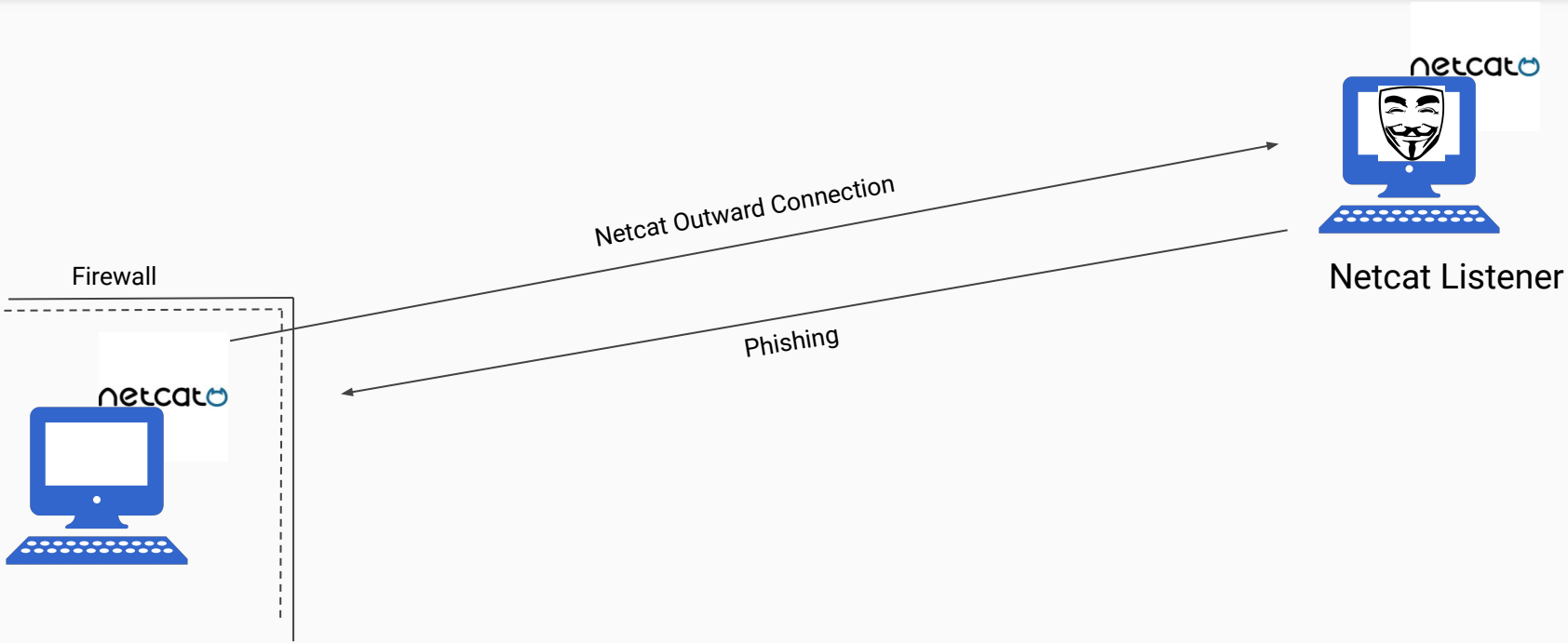
With Great Power Comes Great Responsibility

While NetCat is an extremely powerful and useful, it is also dangerous

NetCat can be used to instigate reverse shell exploits

Reverse Shells

The Network Uno Reverse Card



netcat.py

Libraries and the Execute Method

```
1   import argparse
2   import socket
3   import shlex
4   import subprocess
5   import sys
6   import textwrap
7   import threading
8
9   def execute(cmd):
10      cmd = cmd.strip()
11      if not cmd:
12          return
13      output = subprocess.check_output(shlex.split(cmd), stderr = subprocess.STDOUT)
14      return output.decode()
```

Main Function

```
103  if __name__ == "__main__":
104  # Argparser used to provide arguments to create a command line interface
105      parser = argparse.ArgumentParser(
106          description = 'BHP Net Tool',
107          formatter_class = argparse.RawDescriptionHelpFormatter,
108
109  # Display example usage whenever the user uses the --help switch
110          epilog = textwrap.dedent('''Example:
111              netcat.py -t 192.16.1.108 -p 5555 -l -c #command shell
112              netcat.py -t 192.16.1.108 -p 5555 -l -u = mytest.test #upload file
113              netcat.py -t 192.16.1.108 -p 5555 -l -e = \"cat /etc/passwd\" #execute command
114              echo 'ABC' | ./netcat.py -t 192.168.1.108 -p 135 #echo text to server port 135
115              netcat.py -t 192.168.1.108 -p 5555 #connect to server
116          '''))
117
118  # Six parsed arguments to specify how the program reacts
119      parser.add_argument('-c', '--command', action = 'store_true', help = 'command shell')
120      parser.add_argument('-e', '--execute', help = 'execute specified command')
121      parser.add_argument('-l', '--listen', action = 'store_true', help = 'listen')
122      parser.add_argument('-p', '--port', type = int, default = 5555, help = 'specified port')
123      parser.add_argument('-t', '--target', default = '192.168.1.203', help = 'specified IP')
124      parser.add_argument('-u', '--upload', help = 'upload file')
125
126  # Netcat is set up as a listener
127      args = parser.parse_args()
128      if args.listen:
129          buffer = ''
130      else:
131          buffer = sys.stdin.read()
132
133      nc = NetCat(args, buffer.encode())
134      nc.run()
```

The NetCat Class

Initialization and the Run Method

```
16  class NetCat:
17  # Initialize NetCat object and creates the socket object
18      def __init__(self, args, buffer = None):
19          self.args = args
20          self.buffer = buffer
21          self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
22          self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
23
24  # Sets up a listener if the listen method is called and calls the send method otherwise
25      def run(self):
26          if self.args.listen:
27              self.listen()
28          else:
29              self.send()
```

The Send Method

```
31      def send(self):
32  # Connects to target port and sends any buffers to the target if buffer returns as true
33          self.socket.connect((self.args.target, self.args.port))
34          if self.buffer:
35              self.socket.send(self.buffer)
36
37  # Loops through as data is received
38          try:
39              while True:
40                  recv_len = 1
41                  response = ''
42                  while recv_len:
43                      data = self.socket.recv(4096)
44                      recv_len = len(data)
45                      response += data.decode()
46                      if recv_len < 4096:
47                          break
48  # Prints response data, pause to receive and then send interactive input
49                  if response:
50                      print(response)
51                      buffer = input('> ')
52                      buffer += '\n'
53                      self.socket.send(buffer.encode())
54  # Ctrl+C to quit
55          except KeyboardInterrupt:
56              print('User terminated.')
57              self.socket.close()
58              sys.exit()
```

The Listen Method

```
60      def listen(self):
61  # Binds the target and the port
62          self.socket.bind((self.args.target, self.args.port))
63          self.socket.listen(5)
64
65  # Listens on a loop
66          while True:
67              clientSocket, _ = self.socket.accept()
68  # Any connected sockets are passed off to the handle method
69              clientThread = threading.Thread(
70                  target = self.handle, args = (clientSocket,)
71              )
72              clientThread.start()
```

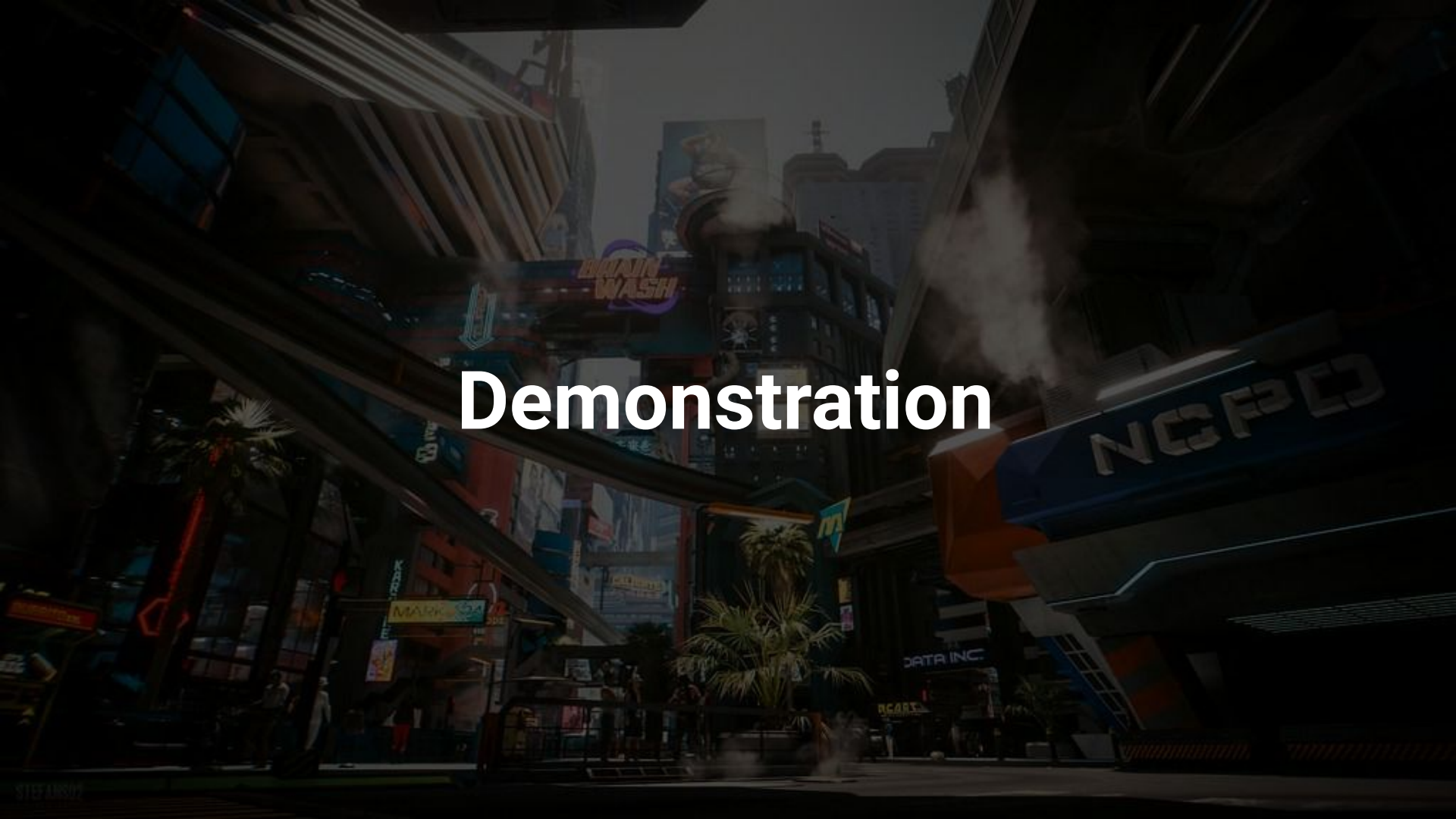
The Handle Method

```python
    def handle(self, clientSocket):
# Passes executes to the execute function and outputs on the socket
        if self.args.execute:
            output = execute(self.args.execute)
            clientSocket.send(output.encode())

# Loops to listen for content then write content to specified file
        elif self.args.upload:
            fileBuffer = b''
            while True:
                data = clientSocket.recv(4096)
                if data:
                    fileBuffer += data
                else:
                    break

            with open(self.args.upload, 'wb') as f:
                f.write(fileBuffer)
            message = f'Saved file {self.args.upload}'
            clientSocket.send(message.encode())

# Sends prompt to sender and loops until command strings return
        elif self.args.command:
            cmdBuffer = b''
            while True:
                try:
                    clientSocket.send(b'BHP: #> ')
                    while '\n' not in cmdBuffer.decode():
                        cmdBuffer += clientSocket.recv(64)
                    response = execute(cmdBuffer.decode())
                    if response:
                        clientSocket.send(response.encode())
                    cmdBuffer = b''
                except Exception as e:
                    print(f'server killed {e}')
                    self.socket.close()
                    sys.exit()
```



Demonstration

Closing Remarks

Netcat on its own is a powerful tool. In the wrong hands it can also be dangerous. Using this netcat script in python, the functionality of netcat is still available, along with more customizability and less risk

Resources

The following resources were used in compiling the information used for this presentation.