EvolveSec: Python and Security 101

Part 1: Network Security and Python
Lab 1 Basic Sniffer

1.1 Creating a raw socket

The script below creates a simple raw socket sniffer by receiving the incoming packets and printing them to the console. Copy the contents of this script into a new file called *basic-sniffer.py* (This script uses a soft wrap to visually fit on the page).

```
import socket

try:
    raw_socket = socket.socket(socket.PF_PACKET, socket.SOCK_RAW,
                 socket.htons(0x0800))
except socket.error, e:
    print 'Error occurred while creating socket. Error code: ' + str(e[0]) + ' ,
           Error message : ' + e[1]
    sys.exit();

while True:
    packet = raw_socket.recvfrom(2048)
    print packet
```

Let's break down the above code.

First, we import the *socket* python module. Next, in order to account for error handling, we initiate a *try* block.

In following line, the socket created. The parameters breakdown as follows:

- *raw_socket* is the variable to which the socket is assigned.
- *PF_PACKET* indicates that the packet interface. *PF_PACKET* is Linux specific; *AF_INET* should be used for Windows.
- *SOCK_RAW* specifies the use of a raw socket.
- *htons(0x0800*) is the protocol. The value 0x0800 specifies we are interested in the IP protocol.

Last half of the code is for the *except* block. If an error is encountered, a *try* block code execution is stopped and transferred down to the *except* block. The *except* above prints the corresponding portions of the error message to the console.

In the final block, we use a *while* loop that keeps the script running (indicated by the *True* value followed by no false statements). This is referred to as an infinite loop. Within the loop, the *recvfrom* method is used to handle incoming packets. After the packet is received, it will be printed to the console.

Lab 2 Scapy

2.1 Interactive Scapy

To use Scapy via the interactive command prompt, type *scapy* into the terminal:

```
# scapy
```

Sniffing packets with Scapy can be performed as follows:

```
>>> packet = sniff(iface="eth0", count = 2)
```

This command will listen for the next two packet on interface eth0. If the process does not finish, you can generate traffic by executing a ping to *evolveacademy.io* in a new terminal window.

```
# ping evolveacademy.io -c 2
```

To view the captured packets, you can recursively print them to the Scapy console with:

```
>>> [i for i in packet]
```

You can exit the Scapy interactive prompt with *exit()*.

2.2 Programming with Scapy

Create a new Python file called *scapy-sniffer.py* and copy the contents below into the file.

```
from scapy.all import *

# packet callback
def packet_callback(packet):
    print packet.show()
# start sniffing
sniff(prn=packet_callback, filter="icmp", count=1)
```

Part 2 Web App Testing and Python
Lab 3 Mechanize & XSS

3.1 XSS

In this section, we are testing if the user input prints to the response without any validation. Make sure the following two files are in the same folder.

First, we must create a list of XSS attack vectors. Copy the XSS attack vectors below into a new text file called *xss-attack-vectors.txt*.

```
<SCRIPT>alert('XSS')</SCRIPT>
<SCRIPT>alert("XSS");<SCRIPT>
<IMG SRC=javascript:alert('XSS')>
```

Below is our attack script. Copy the contents of this script into a new file called *mech-xss.py*

```
import mechanize

url = "http://www.webscantest.com/crosstraining/aboutyou.php"
browser = mechanize.Browser()
attackSequence = 1

with open("xss-attack-vectors.txt") as f:
    for line in f:
        browser.open(url)
        browser.select_form(nr=0)
        browser["fname"] = line  # grab the first name field
        request = browser.submit()
        response = request.read()
        print "\nTesting Vector Number " + str(attackSequence)
        print "Vector: " + str(line).strip("\n")
        if response.find(line) > 0:
            print "Result: ** Possible XSS **"
        else:
            print "Result: Unsuccessful"
        attackSequence += 1
```

Let's break down the above code.

First, we import the *mechanize* Python module. Next, we assign a few parameters: 1) the target site to the URL parameter, 2) the mechanize browser function to browser, 3) the attackSequence iterator.

Next, we open the XSS attack vectors file, then read each line of the file into a *for* loop. Within the loop, we make an HTTP GET request to target site, select the target field from the form on the page, set the field value to our attack vector, then submit an HTTP POST request.

After submitting the POST, we read the response. If the corresponding XSS attack vector from *attackSequence* is present in the response without any escaping/encoding or validation, the field is potentially vulnerable to XSS.

Evolve Security Academy

Website: EvolveAcademy.io

Phone: (312) 957-5682

Twitter: Twitter.com/TheEvolveSec

Github: Github.com/EvolveSecurity

Slack: Slack.EvolveSecurity.io

For security services, check out EvolveSecurity.io