MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Scripting for Cybersecurity

Interacting with Network Services

Dylan Smyth

- Telnet
- SSH

Telnet

- Telnet is protocol used to allow communication between hosts over a network
- Most commonly it would be used in the context of providing a way to manage a remote host over the network
- We've seen this in previous lectures and labs, where we accessed a Telnet server and executed commands on the server host

Telnet

```
root@ubuntu-VirtualBox:/home/ubuntu/mininet# telnet 10.0.0.4
Trying 10.0.0.4...
Connected to 10.0.0.4.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
ubuntu-VirtualBox login: ubuntu
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-53-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

22 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Nov 15 16:55:16 GMT 2020 on pts/10
ubuntu@ubuntu-VirtualBox:~$
```

Telnet

- We can automate the process of connecting to a Telnet server and running commands using Python

- For this, we'll use telnetlib

```
#!/usr/bin/python3

from telnetlib import Telnet

con = Telnet("10.0.0.4", 23)
print(con.read_some())
```

Telnet

- This example creates a Telnet connection and prints some text received from the connection, which happens to be the login prompt from the Telnet server

```
#!/usr/bin/python3

from telnetlib import Telnet

con = Telnet("10.0.0.4", 23)
print(con.read_some())
```

```
ubuntu@ubuntu-VirtualBox:~/my_scripts$ ./telnet_example.py
b'Ubuntu 20.04.1 LTS\r\nubuntu-VirtualBox login:'
```

Telnet

- This library has several functions we can use to read data from the connection

`Telnet.`**`read_until`**(*expected*, *timeout=None*)
Read until a given byte string, *expected*, is encountered or until *timeout* seconds have passed.

When no match is found, return whatever is available instead, possibly empty bytes. Raise `EOFError` if the connection is closed and no cooked data is available.

`Telnet.`**`read_all`**()
Read all data until EOF as bytes; block until connection closed.

`Telnet.`**`read_some`**()
Read at least one byte of cooked data unless EOF is hit. Return `b''` if EOF is hit. Block if no data is immediately available.

`Telnet.`**`read_very_eager`**()
Read everything that can be without blocking in I/O (eager).

Raise `EOFError` if connection closed and no cooked data available. Return `b''` if no cooked data available otherwise. Do not block unless in the midst of an IAC sequence.

`Telnet.`**`read_eager`**()
Read readily available data.

Raise `EOFError` if connection closed and no cooked data available. Return `b''` if no cooked data available otherwise. Do not block unless in the midst of an IAC sequence.

`Telnet.`**`read_lazy`**()
Process and return data already in the queues (lazy).

Telnet

- The documentation can be found here: https://docs.python.org/3/library/telnetlib.html
- The previous example used read_some() to get some data sent by the server
- The read_until() function can be used to read data until we find a string that we're looking for
- We can use this to detect when the Telnet login prompt has been sent by the server

Telnet

- In this example we
  - Connect to the Telnet server
  - Define a variable *text* and provide with an ascii encoded string
  - Use the read_until to read until the text string is seen
  - Print out the text that we received

```
#!/usr/bin/python3

from telnetlib import Telnet

con = Telnet("10.0.0.4", 23)
text = ("login:").encode("ascii")
recv = con.read_until(text)
print(recv)
```

Telnet

- This line is used to *encode* the string “login:” to ASCII

```
text = ("login:").encode("ascii")
```

- ASCII is one of many character encoding standards

- Encoding defines the mapping between binary data and the character that data represents

Telnet

- The ASCII table, or character mappings as defined in the ASCII encoding standard, can be seen by running the command “**man ascii**”

- Part of the output is shown below

Oct	Dec	Hex	Char
100	64	40	@
101	65	41	A
102	66	42	B
103	67	43	C
104	68	44	D
105	69	45	E
106	70	46	F
107	71	47	G

Telnet

- Python3 uses Unicode by default
- Unicode is an encoding standard that includes the ASCII mappings but also includes much more
- It contains mappings for characters outside of the English alphabet, common symbols, numbers, etc.

Telnet

- Telnetlib comes from Python2
- The library expects strings to be encoded in ASCII rather than Unicode, thus the reason for the following line:

```
text = ("login:").encode("ascii")
```

- The .encode() function expects an encoding format to be provided as an argument
- The function returns the Byte representation of the string encoded in the encoding standard we provided

Telnet

- Any text we write to the Telnet connection needs to be encoded using ASCII and any text we read will be encoded using ASCII
- We can write a quick helper function to help with this
- Now instead of “string”.encode(“ascii”) we can write enc(“string”)

```
def enc(s):
    return s.encode("ascii")
```

Telnet

- To log into the Telnet server we will need to
  - Wait for the login prompt
  - Provide the username
  - Wait for the password prompt
  - Provide the password
- After the above we should be connected

Telnet

```
#!/usr/bin/python3

from telnetlib import Telnet

def enc(s):
  return s.encode("ascii")

con = Telnet("10.0.0.4", 23) # Connect
con.read_until(enc("login:")) # Wait for login prompt
con.write(enc("ubuntu\n")) # Provide username
con.read_until(enc("Password:")) # Wait for password prompt
con.write(enc("ubuntu\n")) # Provide password

con.write(enc("ifconfig\n")) # Run a command
con.write(enc("exit\n")) # Exit session

print(con.read_all()) # Print session data
```

Telnet

Demo!

Telnet

- Telnet is old
  - It was first proposed in RFC-15 from 1969
  - https://tools.ietf.org/html/rfc15
- Data sent over Telnet is not encrypted
- The Telnet client can be used to communicate with things that aren't a Telnet server…

Telnet

- 10.0.0.4 is running a Python web server

```
ubuntu@ubuntu-VirtualBox:~/my_scripts$ telnet 10.0.0.4 8000
Trying 10.0.0.4...
Connected to 10.0.0.4.
Escape character is '^]'.
test
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
        <title>Error response</title>
    </head>
    <body>
        <h1>Error response</h1>
        <p>Error code: 400</p>
        <p>Message: Bad request syntax ('test').</p>
        <p>Error code explanation: HTTPStatus.BAD_REQUEST - Bad request syntax o
r unsupported method.</p>
    </body>
</html>
Connection closed by foreign host.
```

SSH

- Secure Shell (SSH) is a protocol used to interact with network devices
- Similar to Telnet but is encrypted
- SSH listens on port 22
- Linux has an SSH client by default. Server can be installed

```
sudo apt-get install openssh-server
```

SSH

- A client can connect to an SSH server like so:
  - ssh user@server

```
ubuntu@ubuntu-VirtualBox:~/my_scripts$ ssh ubuntu@10.0.0.4
ubuntu@10.0.0.4's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-53-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

22 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Nov 22 17:14:35 2020 from 10.0.0.4
ubuntu@ubuntu-VirtualBox:~$
```

SSH

- We can automate the process of connecting to a server and running commands using Python

- There are several libraries available for this. We'll use paramiko

```
sudo apt-get install python3-paramiko
```

SSH

```
#!/usr/bin/python3

from paramiko import SSHClient, AutoAddPolicy

client = SSHClient()
client.set_missing_host_key_policy(AutoAddPolicy())
client.connect("10.0.0.4", username="ubuntu", password="ubuntu")

client.close()
```

- In this example we connect to an SSH server running on 10.0.0.4

- We connect and exit straight away

SSH

- We can run commands on the server after connecting:

```
#!/usr/bin/python3

from paramiko import SSHClient, AutoAddPolicy

client = SSHClient()
client.set_missing_host_key_policy(AutoAddPolicy())
client.connect("10.0.0.4", username="ubuntu", password="ubuntu")

stdin, stdout, stderr = client.exec_command("ifconfig")

command_output = stdout.read()

print(command_output)

client.close()
```

SSH

- stdin -> If we run a command that expects further input (e.g. the Python interactive shell) we can interact with it using stdin
- stdout -> The output of the command we just ran
- stderr - > If the command fails and produces an error we can see that here

SSH

Demo!

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Thank you