Lab cheat sheet

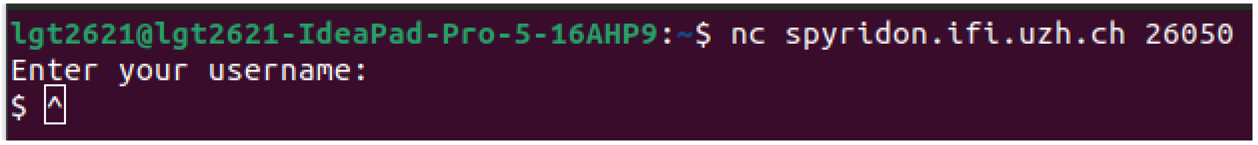
Ncat

- Tool to send and receive network traffic
- Installation
  - Linux: apt-get install netcat
  - Windows: Download and install nmap

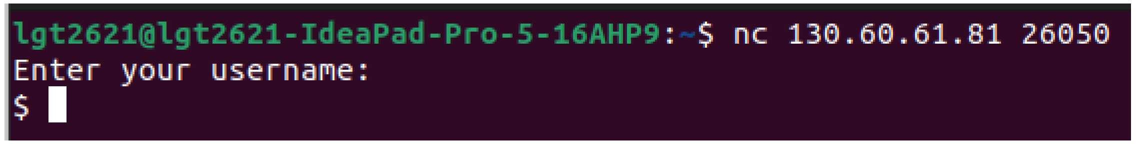
Ncat

- Linux: nc HOST PORT
  - HOST – IP address or Fully Qualified Domain Name (FQDN) of the device
  - PORT – The port number of the device

- Windows: ncat HOST PORT



```
lgt2621@lgt2621-IdeaPad-Pro-5-16AHP9:~$ nc spyridon.ifi.uzh.ch 26050
Enter your username:
$ ^
```



```
lgt2621@lgt2621-IdeaPad-Pro-5-16AHP9:~$ nc 130.60.61.81 26050
Enter your username:
$
```

pwntools

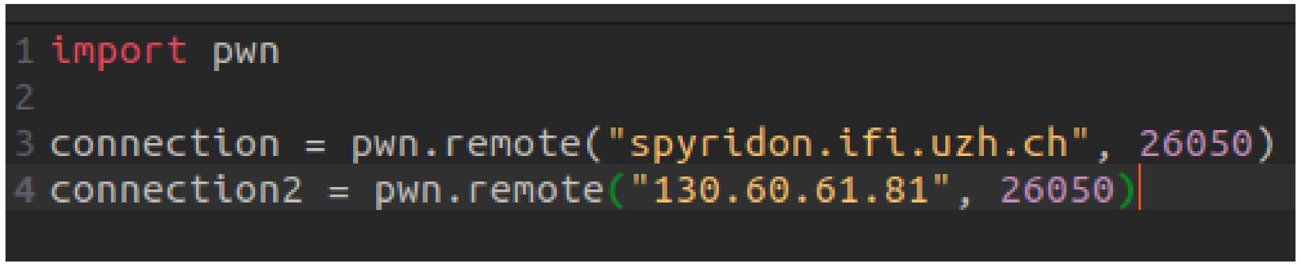
- CTF framework and exploit development library
- Installation
    - pip install --upgrade pwntools

      or
    - python3 -m pip install --upgrade pwntools

PWNTOOLS

pwntools – Opening a connection

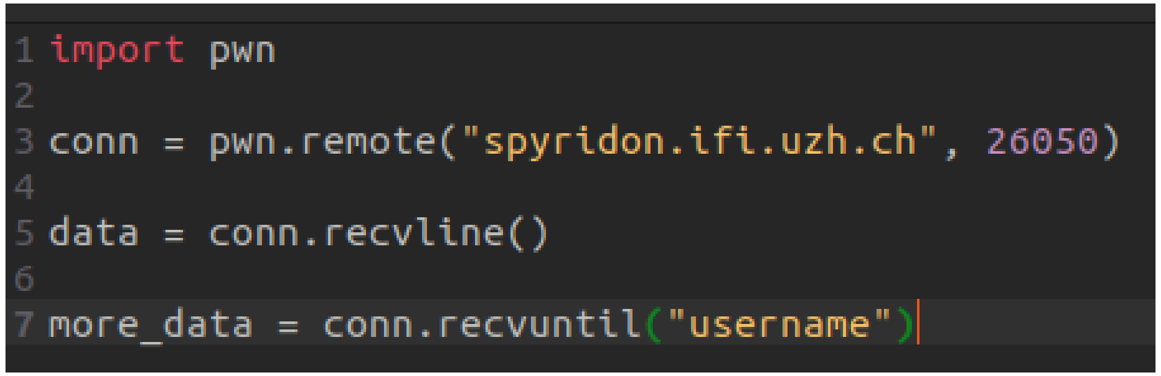
- Import the pwn package
  - Add "import pwn" to the start of your file
- Use pwn.remote() to connect to the device
- conn = pwn.remote(host, port)
  - host – The FQDN or IP of the remote device (string)
  - port – The port of the remote device (int)
  - conn – The open connection used to interact with the device



```python
import pwn

connection = pwn.remote("spyridon.ifi.uzh.ch", 26050)
connection2 = pwn.remote("130.60.61.81", 26050)
```

pwntools – Receiving data

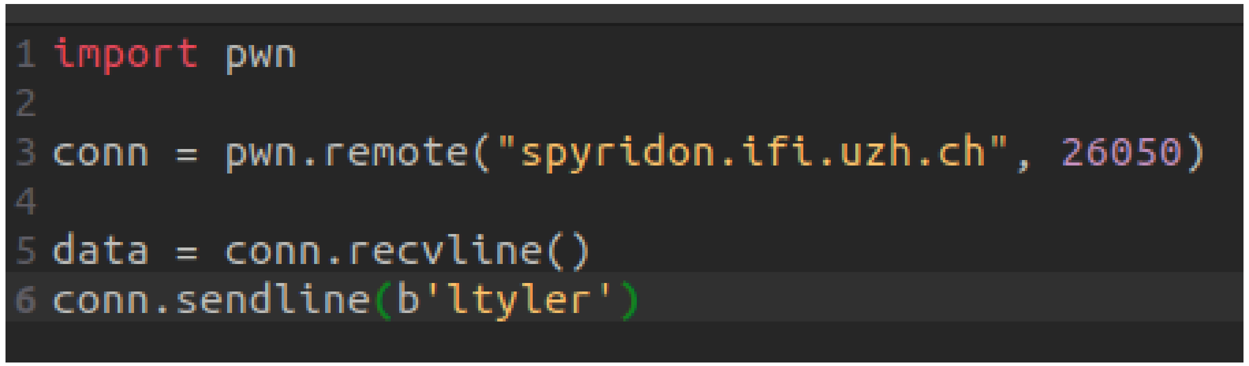
- conn.recvline()
  - Reads in the next line of data from the network
- conn.recvuntil(pattern)
  - pattern - a string to look for within the parsed data
  - Reads data from the network until the supplied pattern is reached
- Data is returned as bytes (bytestring)



```python
import pwn

conn = pwn.remote("spyridon.ifi.uzh.ch", 26050)

data = conn.recvline()

more_data = conn.recvuntil("username")
```

pwntools – Sending data

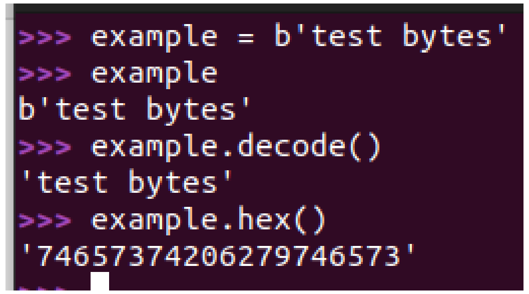
- conn.sendline(data)
  - Sends the supplied data to the server
  - data is in bytes



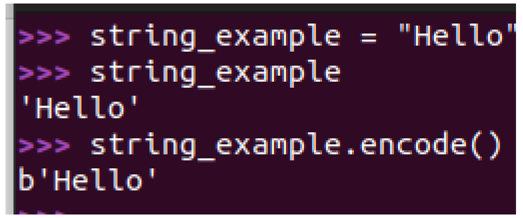
```python
import pwn

conn = pwn.remote("spyridon.ifi.uzh.ch", 26050)

data = conn.recvline()
conn.sendline(b'ltyler')
```

Useful Python Type Conversions

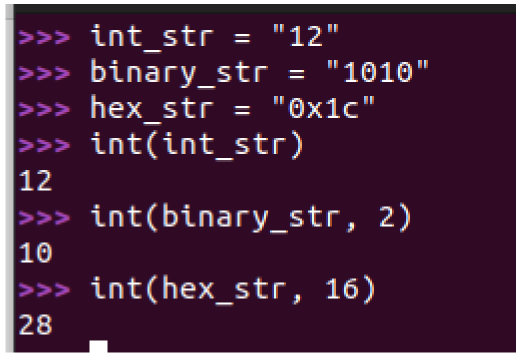
- Bytes/Bytestring
  - b'data'/b"data" - create a bytestring
  - .decode() - decode bytes into a python string
  - .hex() - convert bytes to hexadecimal string
- String
  - 'data'/"data" - create a string
  - .encode() - encode string as bytes/bytestring
  - int(str, base) - convert the provided str to an integer
    - base – the base of the number system ex. 16 for hexadecimal
    - str - must be a valid number in the selected base



```
>>> example = b'test bytes'
>>> example
b'test bytes'
>>> example.decode()
'test bytes'
>>> example.hex()
'746573742062797465 73'
```



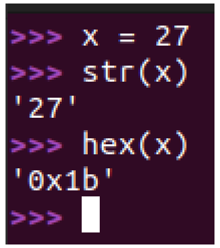
```
>>> string_example = "Hello"
>>> string_example
'Hello'
>>> string_example.encode()
b'Hello'
```



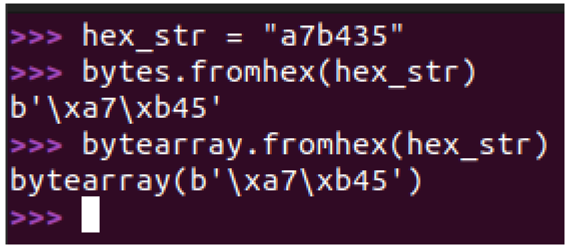
```
>>> int_str = "12"
>>> binary_str = "1010"
>>> hex_str = "0x1c"
>>> int(int_str)
12
>>> int(binary_str, 2)
10
>>> int(hex_str, 16)
28
```

Useful Python Type Conversions

- Int
  - str(int) - converts an integer to a string
  - hex(int) - converts an integer to a hex string
- Hex (special case of strings)
  - bytes.fromhex(hex_string) - convert hex string to bytes
  - bytesarray.fromhex(hex_string) -convert hex string to bytesarray



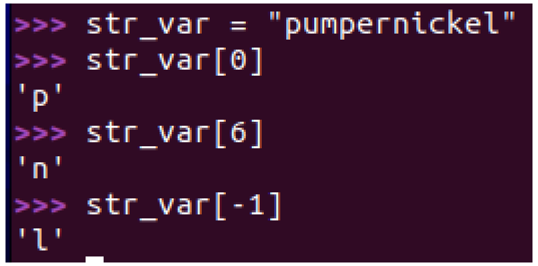
```
>>> x = 27
>>> str(x)
'27'
>>> hex(x)
'0x1b'
>>> 
```



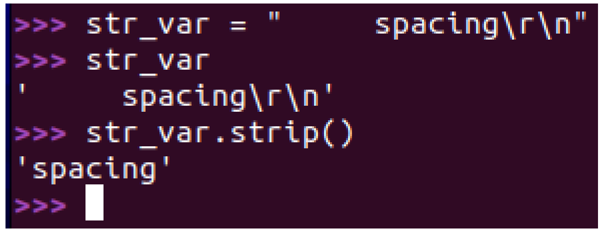
```
>>> hex_str = "a7b435"
>>> bytes.fromhex(hex_str)
b'\xa7\xb45'
>>> bytearray.fromhex(hex_str)
bytearray(b'\xa7\xb45')
>>> 
```

Useful Python String Manipulations

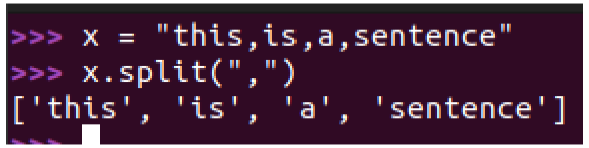
- Strings are indexable
  - Can access individual characters by specifying the index
  - str_var[i] - access the character at index i in str_var
  - Indexing starts at 0
  - Supports negative indexing
- str_var.strip() - remove leading and trailing whitespace
  - Removes spaces and hidden characters like \n
- str_var.split(pattern) - split a string
  - Pattern – the string delimiter to split at
  - Returns a list of substrings



```
>>> str_var = "pumpernickel"
>>> str_var[0]
'p'
>>> str_var[6]
'n'
>>> str_var[-1]
'l'
```



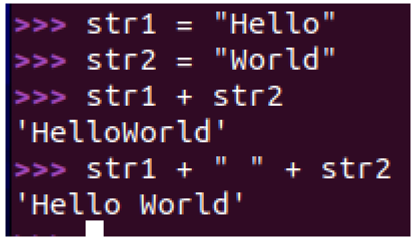
```
>>> str_var = "      spacing\r\n"
>>> str_var
'
    spacing\r\n'
>>> str_var.strip()
'spacing'
>>>
```



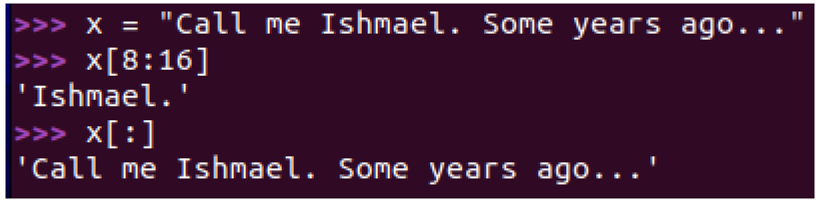
```
>>> x = "this,is,a,sentence"
>>> x.split(",")
['this', 'is', 'a', 'sentence']
```

Useful Python String Manipulations

- Concatenation
  - Strings can be combined with the "+" operator
  - "foo" + "bar" = "foobar"
- Slicing
  - Substrings can be selected using [start:stop]
  - start is inclusive, stop is exclusive
  - "sentence"[3:6] = "ten"
  - Use [:] to make a copy of the string



```
>>> str1 = "Hello"
>>> str2 = "World"
>>> str1 + str2
'HelloWorld'
>>> str1 + " " + str2
'Hello World'
```



```
>>> x = "Call me Ishmael. Some years ago..."
>>> x[8:16]
'Ishmael.'
>>> x[:]
'Call me Ishmael. Some years ago...'
```