Project 2

Problem Statement :-

Part 1: Exploring Network Traces

Security analysts and attackers both frequently study network traffic to search for vulnerabilities and to characterize network behavior. In this section, examine a network packet trace that we recorded on a sample network for this assignment. You will search for specific behaviors and relevant details using the Wireshark network analyzer

Part 2: Network Anomaly Detection

In Part 1, you manually explored a network trace. Now, you will programmatically analyze a pcap file to detect suspicious behavior. Specifically, you will be attempting to identify port scanning.

Abhishek Verma

PART - 1

==========

1. Multiple devices are connected to the local network. What are their MAC and IP addresses? Who manufactured these devices?

-> 00:1e:8c:ea:1a:b4, 10.0.2.191, Asustekc

00:1f:c6:8f:29:17, 10.0.2.231, Asustekc

a4:2b:8c:f6:eb:81, 74.125.225.212, Netgear

Cisco 00:1f:6d:e8:18:00 67.194.192.1 # Networking Device providing routing capabilities

Cisco_Li 00:12:17:31:10:7c 192.168.1.1 # Networking Device providing routing capabilities

==========

2. What type of network does this appear to be (e.g., a large corporation, an ISP back-bone, etc.)? Point to evidence from the trace that supports this.

-> This PCAP file has all the Source IP belonging to a private IP range. So, it implies, this capture is taken on an enterprise router. So, this network belongs to a large corporation.

==========

3. One of the clients connects to an FTP server during the trace.

a. What is the DNS hostname of the server it connects to?

-> dl.xs4all.nl

b. Is the connection using Active or Passive FTP?

-> Active, on port 10,0,2,2,199,51

c. Based on the packet capture, what is one major vulnerability of the FTP protocol?

-> The data transfer via FTP is in clear text. Thus, any sensitive information such as usernames, passwords can be easily read network packet capture techniques such as packet sniffing.

Based on the packet capture, one login attempt by a user involved the use of username as "laticia.langhans" and the password as "goblue3859".

d. Name at least two network protocols that can be used in place of FTP to provide secure file transfer.

-> SFTP, FTPS, SCP

==========

4. The trace shows that at least one of the clients makes HTTPS connections to sites other than Facebook. Pick one of these connections and answer the following:

a. What is the domain name of the site the client is connecting to?

-> www.youtube.com

b. Is there any way HTTPS server can protect against the leak of information in (a)?

-> Yes, by using a more secure cipher suite, can use TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, or even by using GCM over CBC.

c. During the TLS handshake, the client provides a list of supported cipher suites. List the first three cipher suites and name the crypto algorithms used in each.

-> TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

256-bit AES encryption with SHA-1 message authentication and ECDH key exchange signed with an RSA certificate

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

256-bit AES encryption with SHA-1 message authentication and ECDH key exchange signed with an ECDSA certificate

TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA

256-bit CAMELLIA encryption with SHA-1 message authentication and DHE key exchange signed with an RSA certificate

d. Are any of these cipher suites worrisome from a security or privacy perspective? Why?

-> TLS_ECDHE_RSA_WITH_RC4_128_SHA => IETF has prohibited RC4 for its use in TLS

e. What cipher suite does the server choose for the connection? ->TLS_ECDHE_RSA_WITH_RC4_128_SHA

==========

5. One of the clients makes a number of requests to Facebook.

a. Even though logins are processed over HTTPS, what is insecure about the way the browser is authenticated to Facebook?

These requests to facebook are not made directly to facebook but by a third party website to facebook possibly to validate the membership of the user as can be seen in the referrer section

There are two things which appear insecure about the way the browser has authenticated to Facebook:

GET /plugins/likebox.php?api_key=116663708370869&channel=http%3A%2F%2Fstatic.ak.facebook.com%2Fconnect%2Fxd_arbiter.php%3Fversion%3D11%23cb%3Df1226c4bb%26origin%3Dhttp%253A%252F%252Fwww.aljazeera.com%252Ff388f9dcec%26domain%3Dwww.aljazeera.com%26relation%3Dparent.parent&colorscheme=light&header=false&height=62&href=http%3A%2F%2Fwww.facebook.com%2Faljazeera&locale=en_GB&sdk=joey&show_faces=false&stream=false&width=298 HTTP/1.1
Host: www.facebook.com

Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_1) AppleWebKit/537.4 (KHTML, like Gecko) Chrome/22.0.1229.79 Safari/537.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://www.aljazeera.com/news/americas/2012/10/20121082737435876.html
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

HTTP/1.1 200 OK
Cache-Control: private, no-cache, no-store, must-revalidate
Expires: Sat, 01 Jan 2000 00:00:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
X-FB-Debug: tsSyOv6EuMf2w1svAFgoTR1FyIQ3f+0yUtcDm2GEWUs=
Date: Mon, 08 Oct 2012 04:41:09 GMT
Transfer-Encoding: chunked
Connection: keep-alive

1. There is a missing “**Upgrade-Insecure-Requests**” Header which does not attempt to upgrade any additional (iframe,etc) requests to https
2. The response header sets “**X-XSS-Protection**” to 0 which makes it vulnerable to Cross Site Scripting attack.

The user is requesting the webpage through some third party web page and accepting any text/html xml ,implying a complete trust on the server’s response to his/her request .

b. How would this let an attacker impersonate the user on Facebook?

Since the entire communication is in HTTP , any simple MITM can follow the stream and take on the user’s identity by capturing the unique identification of the user such as session id, cookie.

The server could be holding a malicious script which will be executed on the user’s side because of their Accept ,which can reveal more data about the user.

c. How can users protect themselves against this type of attack?

Using some sort of VPN to achieve some level of Encryption.Disabling running of scripts(JavaScript) on their browser.

d. What did the user do while on the Facebook site?

The user was redirected to facebook ,

1. Aljazeera-Read some news on Americas
2. Blogspot-Read some article about judge ruling in wifi case

==========

Part - 2 - Problem Statement

Your task is to develop a Python program that analyses a pcap file in order to detect possible **SYN** scans. To do this, you should use a library called **dpkt**. You can find helpful tutorials on its documentation website.

This site also tells you how to get this python library and how to install. Your program (called **detector.py**) must take as a command-line argument the path of a pcap file to be analyzed.

A skeleton file for **detector.py** is included in the assignment files. The output should be the set of IP addresses (one per line)that sent more than three times as many **SYN** packets as the number of **SYN-ACK** packets they received.

Your program should **silently ignore** packets that are malformed or using **another protocol** besides TCP. We will test your program using a variety of input pcap files to make sure your program works with arbitrary network traces.

Part – 2 - Solution

```
from sys import argv
import dpkt
import socket


def detect anomaly(packet capture):
    """
    Process a dpkt packet capture to determine if any syn scan is
detected. For every IP address address that are
    detected as suspicious. We define "suspicious" as having sent more
than three times as many SYN packets as the
    number of SYN+ACK packets received.
    :param packet capture: dpkt packet capture object for processing
    """
    ip_syn = {}
    ip_syn_ack = {}
    count = 0
    for timestamp, buf in packet capture:
        count = count + 1
        print("Packet count :"+str(count))
        try:
        eth = dpkt.ethernet.Ethernet(buf)
        if not isinstance(eth.data, dpkt.ip.IP):
            continue
        ip = eth.data
        if ip.p==dpkt.ip.IP_PROTO_TCP:
            src = socket.inet ntoa(ip.src)
            dst = socket.inet ntoa(ip.dst)
            tcp = ip.data
            syn_count = 0
            syn_ack_count = 0
            if ((tcp.flags & dpkt.tcp.TH_SYN) and (tcp.flags &
dpkt.tcp.TH ACK)):
                if src in ip_syn_ack.keys():
                    syn ack count  = ip syn ack[src]
                syn_ack_count = syn_ack_count + 1
                ip syn ack[src] = syn ack count
            elif (tcp.flags & dpkt.tcp.TH SYN):
                if src in ip_syn.keys():
                    syn count  = ip syn[src]
                syn_count = syn_count + 1
                ip syn[src] = syn count
    except:
        pass
    print("IP Addresses performing scan are as follows:")
    for skey in ip syn.keys():
        syncount = 0
        synackcount = 0
        syncount = ip syn[skey]
        if skey in  ip syn ack.keys():
```

```
        synackcount = ip_syn_ack[skey]
    syn_check = synackcount * 3
    if syncount > syn_check:
        print(skey)

# parse the command line argument and open the file specified
if __name__ == '__main__':
    if len(argv) != 2:
        print('usage: python detector.py capture.pcap')
        exit(-1)

    with open(argv[1], 'rb') as f:
        pcap_obj = dpkt.pcap.Reader(f)
        detect_anomaly(pcap_obj)
```

Output:

IP Addresses performing scan are as follows:

128.3.23.168
128.3.23.251
119.29.113.198
128.3.23.85
128.3.161.252
128.3.23.81
128.3.23.120
128.3.164.229
198.190.239.129
128.3.23.5
128.3.23.244
128.3.183.133
128.3.23.6
128.3.23.227
128.3.23.3
128.55.174.226
128.3.23.67
128.3.23.15
128.3.23.32
128.3.23.98
128.3.23.50
128.3.23.74
128.3.23.158
128.3.23.117
119.29.113.197
198.128.26.206
128.3.23.216
128.3.23.217
128.3.23.210

128.3.23.231
128.3.23.232
128.3.23.239
128.3.23.2
128.3.23.47
128.3.161.22
128.3.164.135
128.3.23.49
58.93.173.93
128.3.23.42
128.3.23.21
128.3.23.22
128.3.71.100