Victor Tavares – 907 125 5302
Problem 1

Trace 1

1 - Filter used: http && ip.src == 192.168.0.100 && http.host contains "www"

- www.amazon.com
- www.baidu.com,
- www.madisonproperty.com

2 - ip.src == 192.168.0.100 && http.host contains "?" && http.request.method == "GET"

URL: http://www.amazon.com/s/ref=nb_sb_noss_1 , Packet (1879):
Query:

- url=search-alias%3Dstripbooks
- **field-keywords=adventures in Stochastic Processes (Search Query)**

URL: http://www.bing.com/search, Packet 4813
Query:

- **q=chicago metro (Search Query)**
- go=%E6%8F%90%E4%BA%A4
- qs=n
- form=QBRE
- pq=chicago metro
- sc=8-13
- sp=-1
- sk=
- cvid=3dafaeb94bb04c35a9efa4621371beb1

URL: ": http://www.bing.com/search", Packet 4078
Query:

- **q = madison map (Search Query)**
- go=
- qs=n
- form=QBLH
- pq=madison map
- sc=8-5
- sp=-1
- sk=
- cvid=21ab45dc55634583bed819be11051864

Trace 2

1. Username: shiningmoon (packet 8)
   Password: Public (packet 12)

2. FTP is a service that use two ports, a data port and a command port (also know as control port). In an active FTP the client connects from a random port to a FTP server's command port sending the command port N+1. The server will send back a ACK packet and then initiate a connection on its local data port to the data port the client specified earlier. The main problem in this approach is that from the firewall perspective this appears to be an outside system initiating a connection to a internal client, a kind of procedure that is usually blocked.

   A passive FTP solve this issue, in this approach the client initiates both connection to the server. First, the client contact the server on the command port and issue a PASV command. The server will open a random port , and send back N to the client, telling him which port it is listening to. After that, the client initiates the data connection from its data port the data port specified by the server.

3. None of the FTP connections are active. We don't see any port command on the FTP packets, and also the filter ftp.active.port doesn't return anything.

4. All the FTP connections are passive, we can see PASV commands, a command used in passive connections, on packets 30, 75, 110, 146, 180, 217 and 243.

5. Using the filter ftp.request.command == "RETR":
   a. Dragon.zip
   b. ARP.java
   c. L2Switch,java
   d. Phase1.html

Trace 3

1. Source IP address: 192.168.0.100
2. Destination IP address: 74.125.225.46
3. 192.168.0.1
   10.131.180.1
   96.34.20.20
   96.34.17.95
   96.34.16.112
   96.34.16.77
   96.34.2.4
   96.34.0.7
   96.34.0.9
   96.34.3.9
   96.34.152.30
   209.85.254.120
   209.85.250.28

4. 192.168.0.1 -Internet Assigned Numbers Authority - IANNA(Private address)
   10.131.180.1 -Internet Assigned Numbers Authority - IANNA(Private address)
   96.32.0.0 - 96.42.255.255 - Charter Communications
   209.85.128.0 - 209.85.255.255 - Google Inc.

Trace 4

1. Username: cs155@dummymail.com, password: whitehat (packet 6930, 7381)
2. 5 messages (packet 9006)
3. Packet 9728
   Date: "Fri, 23 Apr 2010 08:20:52 -0700"
   From: "joe <cs155@dummymail.com>"
   To: "cs155@dummymail.com"
   Subject: "foobar"

   Packet 11006
   Date: "Fri, 23 Apr 2010 08:23:25 -0700"
   From: "joe <cs155@dummymail.com>"
   To: "cs155@dummymail.com"
   Subject: "can you see this subject?"

   Packet 11961
   Date: "Fri, 23 Apr 2010 10:25:00 -0700"
   From: hariny <harinym@stanford.edu>
   To: "cs155@dummymail.com"
   Subject: "test message"

   Packet 13302
   Date: "Fri, 23 Apr 2010 08:22:28 -0700"
   From: hariny <harinym@stanford.edu>
   To: "cs155@dummymail.com"
   Subject: "geology rocks!"

   Packet 13594
   Date: "Fri, 23 Apr 2010 08:21:51 -0700"
   From: hariny <harinym@stanford.edu>
   To: "cs155@dummymail.com"
   Subject: "wassup"

Problem 2

a. The IP structure specified by <netinet/ip.h> is not compatible with IPv6 addresses. The attacker could use it to deceive the IDS or crash it.
   Solution: The network engineer could implement a new kind of IP structure that supports IPv6, and after check if its IPv4 or Ipv6, assigns the given value to the right structure.

b. Just sniffing and saving the TCP SYN packets will not solve the problem of TCP SYN flood attacks, without a filtering mechanism or other kind of countermeasure, the program would be just saving a bunch of IP's for no reason.
Solution: There's a couple of countermeasures for TCP SYN floods. At the End-Host, we could increase the TCP backlog or reduce the SYN-Received timer. Likewise, in a network level we could use SYN cache and SYN cookies.

c. There's almost no verification of the input values. The attacker could spoof what is suppose to be the ip header protocol to 0x06 and the tcp header->th flags to TH_SYN, passing through the condition on line 78, and overwrites payload on line 80 with any value the attacker wants to.
Solution: The program should check if the input values corresponding to the expected values, specially the ones that are used to control the program, such as ip header protocol and length, or the TCP header.

d. There is also no verification from the value of ipheader->iplen, so the attacker could use a integer overflow technique to make the reportBuf the size he wants to.
Solution: the program should check if the iplen value is valid.

e. If the programs uses offline packet logging, the attacker could fill the storage capacity of the ID's disk and crash or prevent it of saving new data.
Solution: If the program start using pcap open live() the problem will be solved, because this function uses a pre-specified buffer size value. Also deleting older log data when the log file is bigger than a pre-established size(1 GB for example) would be a good solution.

Problem 3

I'm using the script language python version 2.7. I'm also using the library dpkt that is attached.
Python File: ids.py.

Problem 4

Question 1

In order to perform a successfully DNS cache poising, the attacker should know the small business' UDP port and the IP address from its nameserver, he also should assume that the small business name server allows recursive queries from the internet. Using this information he could forge a DNS response by guessing the Query ID value from the Small business name server . A query ID is a unique identifier created in the query packet that allows the server making the request, to associate the request with the question, this value is kept intact by the server sending the reply and each query has a different query ID.
A DNS cache poising can be mounted in 2 different ways, at the first and most common one, the attacker will fool everybody intended to visit a website A(www.smallbussiness.com), to visit his own malicious website. This kinds of attack can be done in 6 steps:

1. The attacker send a DNS query to the victim nameserver(small business nameserver) with the hostname he wishes to hijack (www.smallbusiness.com).
2. Knowing that the victim nameserver will be asking ns1.smallbusiness.com(as directed from the Root/GTLD Servers) for the hostname IP address, the attacker will flood the small business nameserver with forged DNS reply packets. All purport to be from ns1.smallbusiness.com, but the answer will have the IP address of the attacker malicious website.
3. The Root/GTLD provides referral to ns1.smallbussiness.com.
4. The small business nameserver asks ns1.smallbussiness.com for the IP address of www.smallbusiness.com with a query ID of value N.
5. The real nameserver ns1.smallbussiness.com provides a legitimate response, but if the attacker have successfully matched the query ID(N) on the step 2, this reply will arrive too late and is ignored.
6. With the malicious website IP address in cache, it provides the poisoned answer to the attacker's client and to future DNS clients, because the its stored on the cache.

In a second and more harmful attack, the attacker will hijack the authority records, now instead of just one host, he could take over all the second level domain. This attack can be done as following:

1. The attacker requests a randon name within the target domain (www.121121smallbusiness.com). Something unlikely to be in cache.
2. As before, the attacker will send forget packets to the victim nameserver(small business nameserver), but instead of IP address records, it delegates to another name server via authority records.
3. The authority data may contain the "right" www.smallbusiness.com nameserver hostname (ns1.smallbusiness.com), but IP address will be from the attackers name server. Having a Query ID match as in the first example, makes the victim believes that the attacker name's server are the "real" ones for www.smallbusiness.com.
4. Now the attackers owns the entire second level domain and the rest of the steps doesn't really matter, all the rest will be redirect to the attacker's server.

As explained before, a usefully DNS cache poising, will make an user who have the intention of visit the website www.good.com, to visit the malicious website www.bad.com. This could be used by the attacker to download some malicious software in the client computer, or even grab the client user information phishing the "original" website www.good.com.

Question 2

The attacker will know if the website "www.buysomeparts.com" was visited or not in the last 15 minutes, by doing a DNS query. If the replies came from the cache, the name server will return a answer with the "AA - authoritative answer" set to 0, otherwise, he will return an answer with the AA set to 1. When the client hits the cache it means that the website was visited at least once on the least 15 minutes(TTL from the cache).

The TTL limits the accuracy of the attack. If the site was visited multiple times by multiple employees within the 15 minutes, only one visit would be recorded. If TTL is set to a value lower than 15 minutes, we could improve the accuracy, but we the cache efficiency would be affected.