**Lab 8 – compromise investigation:**

For readability here, I’ve inserted superscript numbers wherever there is a relevant screencap, and I have sorted the screencaps below to correspond to the superscripts

First thing I did was pull it up in networkminer to get a summary of activity, users, and interactions. Right away what catches my eye is the metasploitable1 tag on 192.168.3.5. Just looking at the session summaries this (192.168.3.5) is the server.   
just scrolling over the session history, over 2k+ sessions, and virtually all of them coming from 203.0.113.10. Right off the bat I know this is a scan of the network.

Peeking at the parameters tab it is clear our suspect successfully logged into the ftp service on 192.168.3.5, meaning we have a breach of the system, and probably loss of data/information leakage.

The parameters tab highlighted some parts of the exchange, with failed ftp connections on 192.168.3.5 and 192.168.3.13 apparently trying to open a CLI2. followed by an http GET over port 80… but then there’s some weird parameters being passed over port 803, a connection:close, what appears to be php version info, apache version info. This sequence ended in an ‘error closing link’ message getting sent out port 6667, commonly used for IRC.

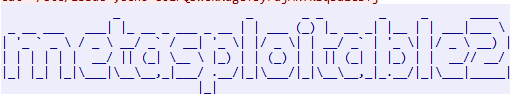
At the end of this exchange I can see there was a successful login attempt to the ftp service4 on 192.168.3.5 by 203.0.113.10 Using credentials [ user: 0M:) ] and [ Pass: azz ]at 21:38:38, and then a timeout message from the server 5 minutes later. This was followed up by an unsuccessful ftp login attempt on 192.168.3.13 with different credentials [ user: 1dxF:) ] and [ Pass: 0ibjZ ], and then dhcp traffic for a few hours before the end of pcap.

I opened the pcap on wireshark now, used the filter ip.src == 203.0.113.10. My suspicions about the client running a scan were confirmed, as the results were a whole lot of of ICMP traffic followed by a flood of tcp traffic, presumably testing all available ports on anything that responded.

What I’m looking for is what went on in that http session, the IRC connection, and what files (if any) were pulled over FTP.

Before I add any more filters I just scroll all the way through. And I start noticing a whole lot of tcp traffic between 203.0.113.10 and our server 192.168.3.5 over some odd ports 60155 and 6200…  
port 6200 is used for lm-x licensing software,  
port 60155 is used for Xsan and Xsan file system access… by the name, that doesn’t sound like something an outsider should be touching.

I follow the tcp stream and play around with it a bit. A couple curious findings come up.  
first, when I click through the packets one by one, looking at the packet content bytes, there’s what appears to be a request for ID, a root root ID submission, and then a couple packets later a ‘whoami’5… I’m not 100% on where this falls in the sequence of events, these ports are foreign to me, and they weren’t mentioned in what I saw in the summary, but I can find that later. What I do know is: this is where they got in.

Second, The tcp stream displays essentially a record of everything he did on the CLI that he managed to pull up.6a it is apparent that he got metasploitable execution on the machine, and it looks like he got a hold of the encrypted passwords6b 6c, and he must have ran them through something like john the ripper to get the credentials he used for the ftp login. The session ends with “cat /etc/fstab” I’m not sure what this directory holds, and then a ping to google, and then a reset packet from 203.0.113.10.

Then I looked at the ftp exchange. And surprisingly after the login, there was no further data exchanged.7

Then I looked up the IRC connection, which turned out to be a dead end.8

The packet numbers of these packets though are significantly lower than the shell execution above. This happened around 5k packets, and the successful shell was opened around 9k packets~~. So that places it on our timeline between the IRC connection and the FTP login~~.

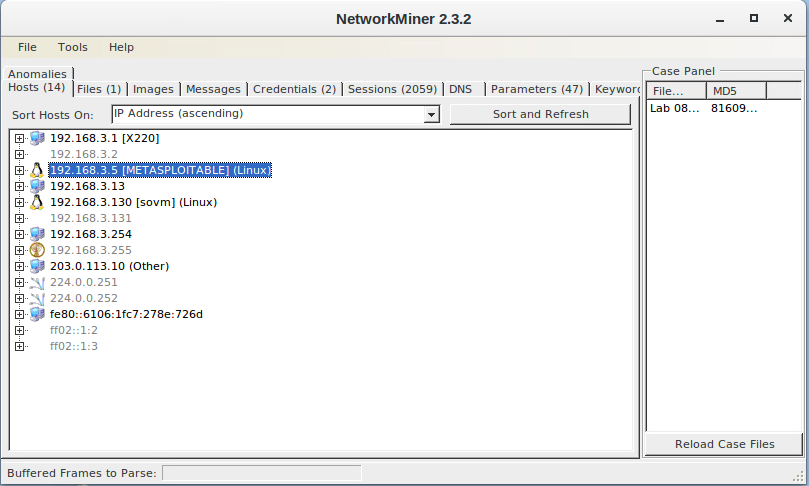
I did apply this filter: tcp.port ==80

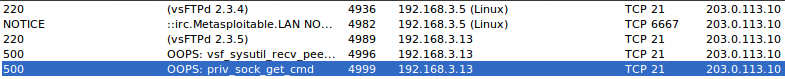
And scrolled through watching the bytecode for something intelligible, and opened followed the tcp stream once I found something.  
This filter/stream caught my attention  
tcp.stream eq 20239

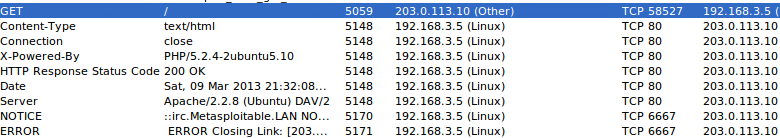
And it turns out all those parameters I thought were strange were part of this tcp exchange, components to one specific packet even (5148).10

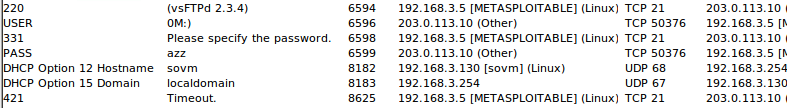
Before I called it quits I figured to check what lead to the open shell on 6200-60155 so I pulled up the stream, checked the packet number, and scrolled around a bit in that range. There was a lot of netbios queries, and portmap requests. In particular I noticed an exchange on 117, which is apparently used for UUCP, a suite of computer programs and protocols allowing remote execution of commands and transfer of files.

I came across the ftp login packets along the way, and I noticed that the order was backwards. I had assumed they got password for the ftp server from the shell execution, but they were already on the machine before they opened the ftp session.  
  
the more I look at it the more odd it becomes.

1

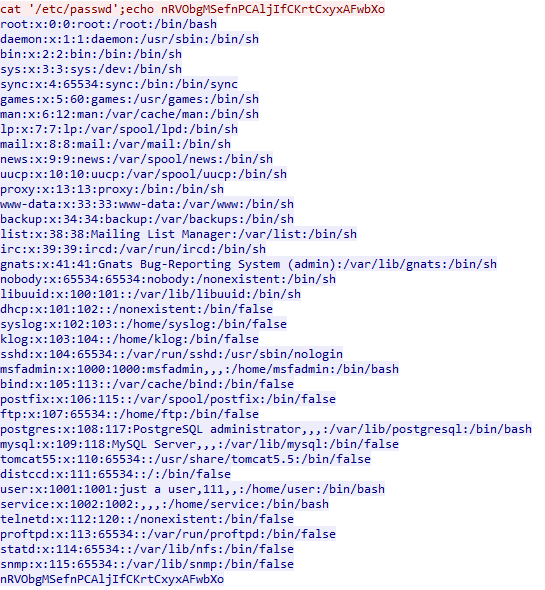
2

3

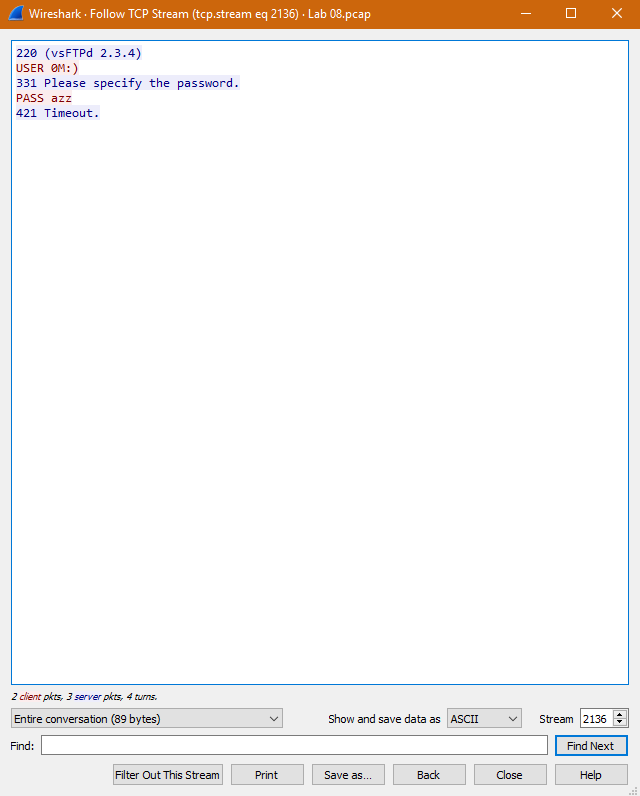
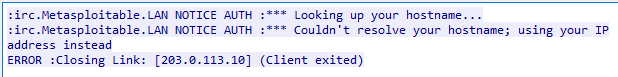
4

<https://gyazo.com/6ccc09ee858a81ade821a74f102b387f> --------------------------------- 5

<https://gyazo.com/e5c68dd8e1a09d4b0f2b5cd5ae61b04a> -------------------------------- 6a

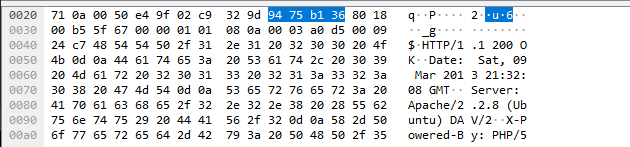
6b

6c

7  


8

9

10