PROTOCOLS - THE PROBLEM WITH CLEARTEXT

Networks are marvelous things are they not, they allow us to transfer data to co-workers or friends, to download files from servers, to work on remote machines and with the internet we can even do this on a global scale. We use various applications to accomplish different tasks, applications are things like email, telnet, ftp, web browsing, etc. And each of these applications has it own "language", or protocol, which it uses to transfer the data used.

This is also where the problem arises. Certain protocols are "cleartext" protocols, this means that if an attacker could listen in on the traffic traveling between you and the machine you were connecting to with one of these applications, they would be able to see the data that is being transferred. Another way to understand the term "cleartext" is as "human-readable". There are multiple common protocols which fall into this category such as pop3, imap, telnet, ftp, snmp, smtp, http, etc. I want to show why this type of protocol is a bad thing, and I want to use two commonly used protocols - telnet and ftp.

In order to capture the packet data used in my examples, I have used the *tcpdump* utility, but there are many, many other utilities for both linux and windows. Some examples are Ethereal, Ngrep, Ettercap, etc (please see the links page for links to these utilities).

Checking Out FTP
FTP stands for File Transfer Protocol and as can be guessed, is used for the uploading or downloading of files. It is a very handy application, but it is cleartext. First we will look at what a user would see logging onto a ftp server;

```
220 syplh FTP server (Version wu-2.6.1-18) ready.
User (10.0.0.11:(none)): demo
331 Password required for demo.
Password:
230 User demo logged in.
ftp>
```

Doesn't look like there is a problem, it did not show the password, so why the fuss? Well, lets take a look at the actual data passed between the client and the server in the above exchange (the relevant sections of the packet data have been emphasized) ;

Packet No.	Packet Contents	Explanation
1	10.0.0.11.ftp > 10.0.0.120.1235 E..[..@.@.ZI.....x..... P...n...**220.syplh.FTP.server.(Version.wu-2.6.1-18)**	This is the server showing the ftp banner text
2	10.0.0.120.1235 > 10.0.0.11.ftp E..(.G@...D....x.............5P.D=.....	
3	10.0.0.120.1235 > 10.0.0.11.ftp E..3.I@...C....x.............5P.D=.:..**USER.demo..**	Here the user sends the username
4	10.0.0.11.ftp > 10.0.0.120.1235 E..(..@.@.Z{.....x.....5....P...H4..	
5	10.0.0.11.ftp > 10.0.0.120.1235 E..I..@.@.ZY.....x.....5....P.....331.**Password.required.for.demo**...	Here the server asks for the user's password
6	10.0.0.120.1235 > 10.0.0.11.ftp E..(.K@...D....x.............VP.D.....	

7	10.0.0.120.1235 > 10.0.0.11.ftp E..5.M@...C....x.............VP.D.....**PASS.secret**..	And here we can see the users password!
8	10.0.0.11.ftp > 10.0.0.120.1235 E..B..@.@.Z_.......x.......V....P...`...**230.User.demo.logged.in**...	Here we see that the password is valid!

So we can see how easy it is for an attacker to listen in on your ftp usage and find your username and password, in fact it is even possible for an attacker who captures all your ftp traffic to rebuild a copy of any files you uploaded or downloaded.

Checking Out Telnet
Telnet allows a user to remotely access a server and work as if they were sitting in front of the server itself. This is also a very handy utility in that multiple users can work on a server a one time, and also so that someone can work on the server from a remote location. It is, unfortunately, also a cleartext protocol. Let's again look at what a user would see logging into a server via telnet;

```
Red Hat Linux release 8.0 (Psyche)
Kernel 2.4.7-10 on an i686
login: demo
Password:
Last login: Wed Sep 29 14:25:04 from 10.0.0.120
[demo@syplh demo]$
```

Again, we see that the password is not displayed, maybe it is fine this time around? Let's look at the data transferred between the client and the server in the above exchange (the telnet data takes a bit more work to read so the relevant sections of the packet data have been emphasized) ;

Packet No.	Packet Contents	Explanation
1	10.0.0.11.telnet > 10.0.0.120.1230 E..h..@.@.i........x.....L....8.P...U2..**Red.Hat.Linux.release.8.0.(Psyche)..Kernel**	This is the server showing the telnet banner
2	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
3	10.0.0.11.telnet > 10.0.0.120.1230 E../..@.@.iH.......x.....L....8.P.......**login:**.	This is the server asking for the username
4	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
5	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L..P.C.....**d**.....	telnet transfers the username 1 character at a time. Here the client is sending the 1st character of the username.
6	10.0.0.11.telnet > 10.0.0.120.1230 E..)..@.@.iM.......x.....L....8.P.......**d**	The server echoes the 1st character
7	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
8	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L..P.C.....**e**.....	The client sends the 2nd username character
9	10.0.0.11.telnet > 10.0.0.120.1230 E..)..@.@.iL.......x.....L....8.P.......**e**	The server echoes the 2nd character

10	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
11	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L..P.C.....**m**.....	The client sends the 3rd username character
12	10.0.0.11.telnet > 10.0.0.120.1230 E..)..@.@.iK.......x.....L....8.P.......**m**	The server echoes the 3rd character
13	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
14	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L..P.C.....**o**.....	The client sends the 4th username character
15	10.0.0.11.telnet > 10.0.0.120.1230 E..)..@.@.iJ.......x.....L....8.P.......**o**	The server echoes the 4th character. And there we see the username is "demo"
16	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
17	10.0.0.120.1230 > 10.0.0.11.telnet E..*..@...F....x.........8..L..P.C..........	
18	10.0.0.11.telnet > 10.0.0.120.1230 E..*..@.@.iH.......x.....L....8.P.........	
19	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L..P.C..........	
20	10.0.0.11.telnet > 10.0.0.120.1230 E..2..@.@.i?.......x.....L....8.P...;...**Password:**.	Here the server is asking for the password
21	10.0.0.120.1230 > 10.0.0.11.telnet E..(..@...F....x.........8..L.#P.C..........	
22	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L.#P.C.....**s**.....	Here we start seeing it! Here is the 1st character of the password sent by the client to the server!
23	10.0.0.11.telnet > 10.0.0.120.1230 E..(..@.@.iH.......x.....L.#..8.P...#...	The server does not echo the character back, but that doesn't help keep the password safe.
24	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L.#P.C.....**e**.....	The client sends the 2nd password character
25	10.0.0.11.telnet > 10.0.0.120.1230 E..(..@.@.iG.......x.....L.#..8.P...#...	
26	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L.#P.C.....**c**.....	The client sends the 3rd password character
27	10.0.0.11.telnet > 10.0.0.120.1230 E..(..@.@.iF.......x.....L.#..8.P...#...	
28	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L.#P.C.....**r**.....	The client sends the 4th password character
29	10.0.0.11.telnet > 10.0.0.120.1230 E..(..@.@.iE.......x.....L.#..8.P...#...	
30	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L.#P.C.....**e**.....	The client sends the 5th password character
31	10.0.0.11.telnet > 10.0.0.120.1230 E..(..@.@.iD.......x.....L.#..8.P...#...	

32	10.0.0.120.1230 > 10.0.0.11.telnet E..)..@...F....x.........8..L.#P.C.....**t**.....	The client sends the 6th password character. Now we can see the password of "secret"
33	10.0.0.11.telnet > 10.0.0.120.1230 E..(..@.@.iC.......x.....L.#..8.P...#...	

We can see that it also easy for an attacker to listen in on your telnet traffic and find your username and password. This would allow the attacker to access the telnet server with all the privileges and access that the legitimate user has.

What Can Be Done?
The good news is that there are ways to prevent this type of attack. One of the easiest is to use a protocol called SSH. It stands for Secure SHell, and is a drop-in replacement for both telnet and ftp. By drop-in, I mean that you can replace telnet and ftp with ssh, but the actual usage and commands used by your users will not change.

SSH helps prevent eavesdropping attacks by creating an encrypted tunnel between the client and the server at time of connection, and all data travels through this tunnel and is thus encrypted, and is therefore not "human-readable" or "cleartext". Let's start by seeing how a client log's on using ssh;

```
login as: demo
Sent username "demo"
demo@10.0.0.11's password:
Last login: Wed Sep 29 14:30:08 2004 from 10.0.0.120
[demo@syplh demo]$
```

Looks a lot like telnet, the username was echoed back to the client, and the password was not displayed. Lets test, lets take a look at the data transferred for the above exchange (relevant sections of the packet data have been emphasized) ;

Packet No.	Packet Contents	Explanation
1	10.0.0.11.ssh > 10.0.0.120.1240 E..?].@.@.........x.....+.:..s^P....)..**SSH-1.99-OpenSSH_2.9p2.**	Here is the server giving it's data.
2	10.0.0.120.1240 > 10.0.0.11.ssh E..D..@...AZ...x.........s^.+.QP.DY....**SSH-1.5-PuTTY-Release-0.53b.**	Here is the client responding. From here onwards it should be encrypted.
3	10.0.0.11.ssh > 10.0.0.120.1240 E..(].@.@.........x.....+.Q..szP.......	Encrypted data.
4	10.0.0.11.ssh > 10.0.0.120.1240 E..<].@.@.........x.....+.Q..szP..... .\...dJ.v......#...I+m.VZ.....&	Encrypted data.
5	10.0.0.120.1240 > 10.0.0.11.ssh E.....@...@....x.........sz.+.eP.CE........^.. ..\...dJ.v...LP>U7.F.Xi+b..7..[K7.	Encrypted data.
6	10.0.0.11.ssh > 10.0.0.120.1240 E..(].@.@.........x.....+.e..t.P....;..	Encrypted data.
7	10.0.0.11.ssh > 10.0.0.120.1240 E..4].@.@.........x.....+.e..t.P....2........./...	Encrypted data.

Looks a lot better, right. We could not see the username, password, in fact besides the inital exchange of data between the server and client, all cleartext had been encrypted. Much better.

Well, thats it for our look at cleartext protocols and their problems, I hope it was informative. Remember that SSH is a brillant protocol that can do a lot more besides encrypt cleartext, but for the purposes of replacing FTP and TELNET, it is a easy and secure solution. Don't be lazy, using a cleartext protocol only makes an atackers job easier, change to encrypted protocols now.