Protocol Reverse Engineering Challenge

This is a 2 part challenge. I was given a client for a stock trading application. I had to pretend to be a server and try to reverse engineer the custom protocol. Then I also had to find way of exploiting the protocol.

References

Art of Software Security Assessment http://www.amazon.ca/The-Software-Security-Assessment-Vulnerabilities/dp/0321444426

Aditya Aghi

1.0 Protocol Communication Flow

By taking turns acting like a server and a client I was able to reverse engineer the messages exchanged between them. Below is an overview of the communication flow:

1) Client sends username and password to server.

'\x00\x1f\x00\x03\x00\x0bjkriebtrich\x00\x0cgo!bears1984'

The orange highlight represents the length of the username and password that follow in green.

2) Server responds with one of the 3 values:

Server Response	Meaning
\x00\x06\xff\x03\x03\xf2	Login Successful
'\x00\x19\xff\xffauthentication failed'	Authentication Failed
'\x00\x10\xff\xffno such user'	User does not exist

3) Client responds with a request for a list of capabilities(orange) followed by system time.

'\x00\x08\x00\x02Sa\xc2\x9f'

4) Server responds with List of capabilities(commands) available to the client:

'\x00X\xff\x02\x00\x05\x00\x08holdings\x00\x06\x00\x04sell\x00\x01\x00\x05hello\x00\x07\x00\x03buy\x00\x02\x00\x0ccapabilities\x00\x03\x00\x05login\x00\x14\x00\x08accounts\x00\x04\x00\x07balance'

Command Number	Command Action
1	hello
2	capabilities
3	login
4	balance
5	holdings
6	sell
7	buy

5) The Client responds with a command for the server to execute. The format of the buy, sell , list holdings, and account balance commands is as follows:

a) Buy Command

'\x00\x0f\x00\x07\x03\xf2\x00\x03APL\x00\x00\x002'

The pink highlight represents the command number(07 – buy), blue highlight the account number, yellow highlight the length of symbol, Orange highlight the symbol, and the green highlight the quantity to buy.

b) **Sell Command**

'\x00\x0f\x00\x06\x03\xf2\x00\x03APL\x00\x00\x002'

Exactly similar format to buy command except command number is now 06.

c) **List Holdings Command**

'\x00\x06\x00\x05\x03\xf2'

Pink highlight is command number (05- list holdings) and blue highlight the account number.

d) **Account Balance**

'\x00\x06\x00\x04\x03\xf2'

Pink highlight is command number (04- account balance) and blue highlight the account number.

2.0 Security Concerns

I am using a severity rating scale of High, Medium and Low.

2.1 Sever does not check authentication state - Severity Level High

I was able to skip steps 1-4 entirely in the communication flow discussed in section 1.0. I tried executing several commands right after opening a socket connection to the server. The **buy, sell, list holdings and account balance** commands all worked **without authentication**. I was even able to change the account number parameter to modify other accounts on the system.

The server does not check the authentication state of a connection before executing a command. The developers might have made an assumption that since the protocol is bespoke no one else would write a client for it. Hence they have decided to trust all clients to always authenticate and close the connection if authentication fails. An attacker who reverse engineers the protocol can easily write a malicious client that executes buy, sell and list holdings commands without authentication to his benefit.

2.2 Password exchanged in plain text – Severity Level High

Another problem is that the password is exchanged in plain text. This can enable someone on the network like an employee to easily sniff packets for passwords.

2.3 Username Enumeration – Severity Level Medium

The “user does not exist” authentication response also enables one to enumerate valid usernames which can then be used for other attacks like password guessing.

2.4 Dangerous testing function – Severity Level High

I also found that command 01- hello resets the stock holdings to a random set and the balance to $5000. I do not see a use for such a command in a real life stock trading application. My guess is that the command was created to reset the environment after every unit test. This command can easily be

used to corrupt data held in the trading application. It should have been removed before releasing to production.

2.5 Buying twice resets account - Severity Level High

Another bug I found is that if you try to buy the same stock twice the second order fails. In addition your account holdings are reset to a random set and your account balance to $5000. This behaviour is reproduce-able using the web QA interface hence I don't think it is a side-effect of using a malicious client. It could be that buying the same stock sends the system in an unstable state (due to a bug) and the system has an auto reset feature inbuilt.

Appendix

Send2.py is the program I used as a test client.

Listen.py is the program I used as a test server.

Both are included in the email attachments.