Message Exchange Format Using Automatic Protocol Reverse Engineering

Yamini Pathania
Post Graduate Student
Dept. of Computer Science Engineering
Chandigarh University
pathaniayamini25@gmail.com

Gurpreet Kaur
Assistant Professor
Dept. of Computer Science Engineering
Chandigarh University
gurpreetcse.cgc@gmail.com

ABSTRACT

In recent year, Protocol reverse engineering has been a manual process which is considered tedious, time-consuming and error-prone. To tackle this limitation, many solutions have been proposed recently to allow automatic protocol reverse engineering. In this paper, we present the results of message format using automatic protocol reverse engineering for Facebook protocol. It is based upon packet capturing and behavior analysis by passive traffic monitoring approach. The message formats are presented in the form of constraint HTTP message specification, and its exchange sequences are represented in sequence diagrams. As the result; due to protocol reverse engineering process it will help us to know in which format the message is delivered when request is arrives.

Keywords

Reverse engineering, Automatic protocol reverse engineering, Graph API, Facebook SDK.

1. INTRODUCTION

Protocol plays an important role in network and information security. Protocol is a set of rules that describe the message format and its exchange sequence. In other words, “it governs the communications between computers on a network”. Different types of network protocols and standards are required to ensure that your computer can communicate with another computer located on the next desk or half-way around the world. One of the most common and known protocols is HTTP (Hyper Text Transfer protocol), that is a used to transmit data over the World Wide Web that is internet.

In recent year, protocol reverse engineering has been a manual process which is considered tedious, time-consuming and error-prone. To tackle this limitation, many solutions have been proposed recently to allow automatic protocol reverse engineering. Protocol reverse engineering consists of two subordinate procedures, message format reverse engineering and message exchange sequence reverse engineering. In this paper, we work on message format reverse engineering along with the source code trace analysis.

We are working on message exchange format using automatic protocol reverse engineering for Facebook Protocol in this paper. It will help us to know in which format the message is delivered when request is arrives. There are number of social network services like Orkut, Twitter, Facebook, Me2day etc. We can manage the relationships online without physical constraints by using these social network services. For our analysis, we choose Facebook because it has its own patented protocol. Its protocol was engineered reversely from raw traffic data to protocol specification through traffic monitoring method and source code trace method. The number of Facebook users over the whole world is also large that use mobile Facebook application and it is the most famous SNS services. Thus, the data traffic between mobile application and Facebook servers will also increases. For this purpose we use the automatic protocol reverse engineering, we work on message format reverse engineering along with the source code trace analysis. For sending intended Facebook messages and producing source code trace during sending and receiving; Facebook mobile client application has been developed using Facebook SDK and Facebook Graph API. We will briefly discuss the working of our work further in the next section.

This paper is organized as follows: In section1, we discuss the Introduction part. Section 2, presented the Background which is organized in two main areas: Reverse Engineering, Protocol Reverse Engineering. Section3 discuss what we are trying to achieve, Our Approach. Section4 describe the Experimental results. And in last Section 5, we discuss the Future work and Conclusion.

2. BACKGROUND

In this section, we briefly introduce reverse engineering and protocol reverse engineering with its applications, limitations, and tools used for automatic protocol reverse engineering.

2.1 Reverse Engineering

Reverse engineering may define as “the process of discovering the technological; principles of a device, system through analysis of its structure, function and operations.” In other words, it is the process of recreating a design by analyzing a final product. . It is the process of extracting knowledge or design information from anything man- made. Reverse Engineering is common in both hardware and software. It generally involves disassembling something and analyzing its components and running in detail, just to re-create it.

Reverse engineering has its origins in the analysis of hardware for commercial or military advantage. The purpose is to decrease design decisions from end products with little or no extra knowledge about the procedures involved in the original production. The same techniques are secondly being researched for application to bequest software systems, not for industrial, but rather to change incorrect, incomplete, or otherwise unavailable documentation. This is used for many purposes:

As a learning tool as a way to make new;
Easy to use products that are cheaper than what’s currently on the market and for making software interoperate more productively or to bridge data between different operating systems or databases;
Used in maintenance;

Used to make a new device or program that does the same thing without copying anything from the original.
Other purpose of reverse engineering include security auditing, removal of copy protection (“cracking”), circumvention of access restriction often present in consumer electronics, customization of embedded systems (such as engine management systems), in-house repairs of retrofits, enabling of additional features on low-cost “crippled” hardware(such as some graphics card chip-sets), or even mere satisfaction of curiosity.

2.2 Protocol Reverse Engineering

Protocol reverse engineering is the technology of extracting the specific documented or undocumented protocol. Protocol reverse engineering consists of two subordinate procedures, message format reverse engineering and message exchange sequence reverse engineering. Protocol reverse engineering is necessary for many security applications. Zhi Wang et.al. [1**]**, proposes ReFormat, a system that enables existing automatic protocol reverse engineering tools to handle encrypted messages. ReFormat achieves high accuracy in locating the decrypted message buffers and extracting the related message structure by evaluated it with a variety of protocol messages from real-world (known or unknown) protocols.

Wang Ying et.al. [3], presented a new method for protocol reverse engineering, which combines both the dynamic and static binary analysis. The result shows that the combination of dynamic binary analysis method and static binary analysis method is effective to infer the message format, but also speculate the state machine model through relevant field attributes conveniently. Yong Wang et.al.[5], proposes a novel framework for automatic unknown protocol formats reverse engineering. The protocol formats reverse engineering technique can be used to detect such attacks; however, earlier works are focused on the application layer protocol analysis, and also problematic work under the scenarios that the captured data is only in binary format due to the lack of semantics.

Network based firewalls require the knowledge of protocol specifications to understand the content of network communication sessions. Accordingly, the problem of protocol reverse engineering can be partitioned into two sub problems: message format and state-machine reverse – engineering.

The **message formats** have traditionally been reverse-engineered through a prolix manual process which is considered tedious, time-consuming and error-prone. This process involved analysis of how protocol implementations process messages. Now-a-days, recent research proposed a number of automatic solutions. These automatic approaches either group observed messages into clusters using various clustering analyses, or emulate the protocol implementation tracing the message processing.

The **protocol state-machines** can be dividing into offline learning and online learning. In offline learning, which passively observes communication and attempts to build the most general state-machine accepting all observed sequences of messages, and in online learning, which allows interactive generation of probing sequences of messages and listening to responses to those probing sequences.

2.2.1 Applications of Protocol Reverse Engineering

Polyglot: we provide the first binary analysis technique for automatic protocol reverse engineering. Polyglot uses a different intuition every time, the way that an implementation of the protocol proceeds the received application data discloses a wealth of information about the style of accepted message.

Dispatcher: Dispatcher is our latest automatic protocol reverse engineering tool and superseeds Polyglot. Dispatcher is able to reverse engineer encrypted protocols by identifying the buffers holding the unencrypted received message after it has been decrypted by the application, and the buffers holding the unencrypted message about to be sent before it is encrypted by the application. Then, it applies the automatic protocol reverse engineering techniques on those buffers.
Replayer: The ability to accurately replay application protocol dialogs is useful in many security-oriented

applications, such as replaying an exploit for forensic analysis or demonstrating an exploit to a third party.

2.2.2 Limitations of Protocol Reverse Engineering

Trace Dependency: The format generated by any tool that operates only on the trace is limited by the diversity of traffic seen in the trace. It is impossible for such a tool to infer those message formats or identify those fields as variable fields, if certain message formats never occur in the trace, or if certain variable fields never take more than one value in the trace,

Pre-Defined Semantics: It is not possible to find all the possible semantics of all fields just from a trace, the best one can hope for is to have an extensible framework where new semantic modules can be added as desired.

2.2.3 Tool Used

Disassembler:-A disassembler attempts to dissect a binary executable into human readable assembly language. The disassemble software reads the raw byte stream output from the processor and parses it into groups of instructions most commonly used disassemble is IDA Pro IDA.

Debugger:-The most commonly used debugger is ollydbg. It is often used by crackers to crack software made by other developers.

WireShark: - WireShark (formerly known as Ethereal) is a free, open source network protocol analyzer. It can be useful for capturing data packets on the network wire for later analysis to understand how proprietary multimedia networking protocols operate.

Internet Junk buster: - It is a useful tool for filtering web requests that a web browser makes to sites that are known to do little more than serve advertisements. However, it is also a highly configurable, general HTTP proxy server and can be leveraged as sort of a poor man's network protocol analyzer.

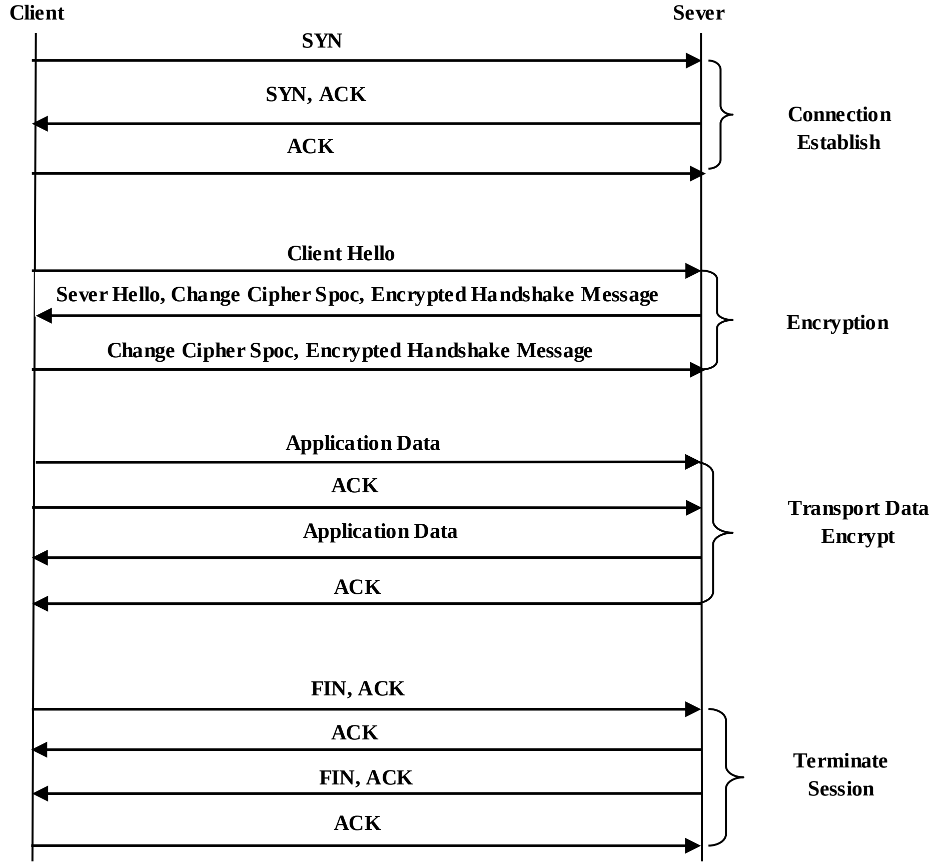


Fig 1: Writing letters in Wi-Fi environment

3. WORK DONE

For our analysis of mobile Facebook data processing and service procedures, we used Facebook android application by creating the experimental environment under network connection of Wi-Fi. Facebook mobile client application has been developed using Facebook SDK and Facebook Graph API which is used for sending intended Facebook messages and producing source code trace during sending and receiving;. We used mobile device (model: GT-S5302, android version: 2.3.6, OS: Gingerbread) and capture packet in such environment by Shark-For-Root application; there is necessity to make certain kind of kernel modification. When we capture and trace data through Wi-Fi by using laptop, we used Wireshark tool. This tool is used to analyze and capturing packet in real time; by analyzing captured data we can get the information about packet size, communication order between Facebook server and client when executing particular service.

By using mobile Facebook android application, we posted status on our wall through Wi-Fi. Then we captured packets and those packets find domain name Graph.facebook.com via DNS server.

From the above Fig 1, there are four steps to message transfer process. In first step, to establish a session between client and server of Facebook; the message used TCP 3way Handshake. It establishes a secure communication over established TCP session using TLS handshake via TLSv, in second step. Third, it exchanging encrypted data to communicates between server and client. In last step, by sending FIN, ACK. to terminate sessions which gives information about transferred packet sequence. But, detailed information was not available due to data encryption through TLS. Thus, we produced an application manually using Facebook Graph API and Facebook SDK.

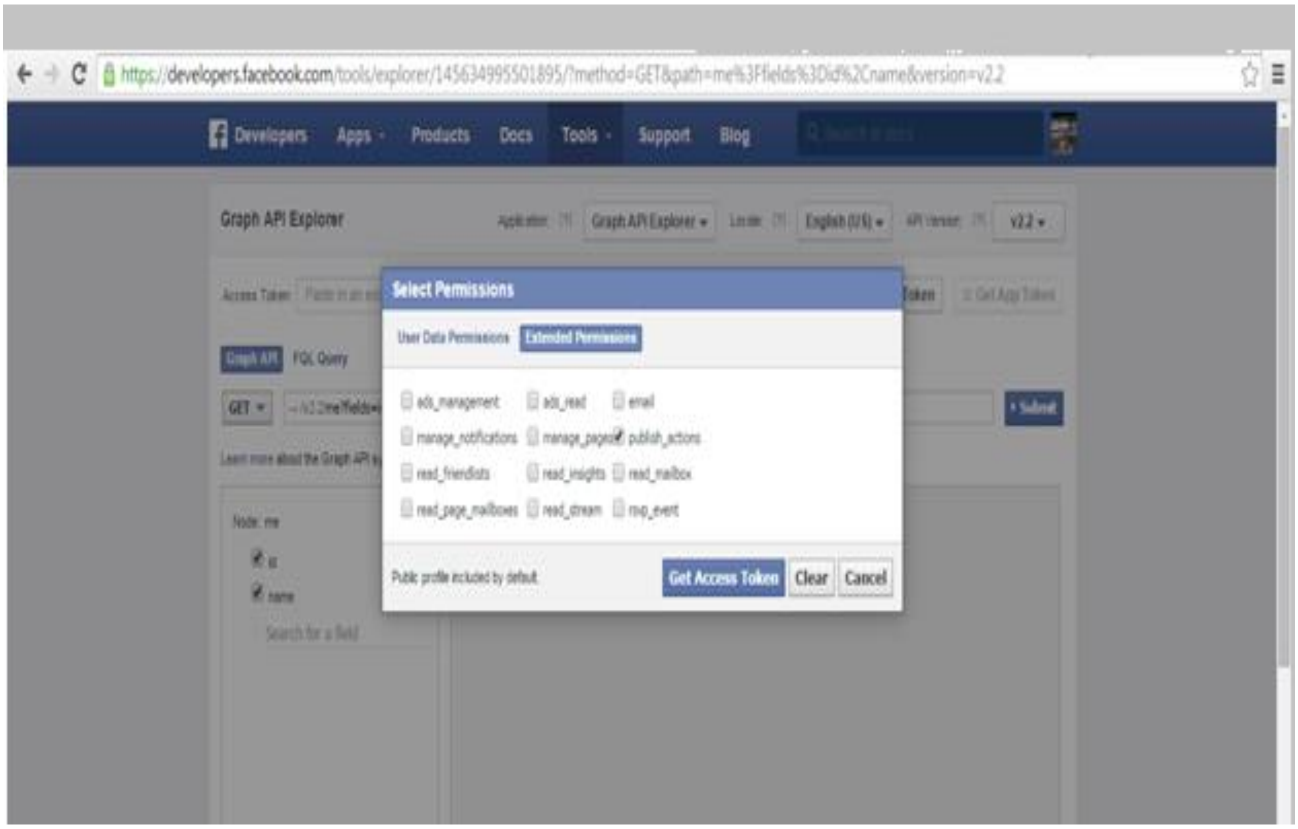
3.1 Facebook SDK and Graph API

Facebook SDK (Software Development Kit) contains predefines libraries functions which we directly call in the program. It provides the API libraries and developer tools necessary to build, test and debug apps for Android. SDK allow developers to creating applications for a certain software package, video game console, software framework, operating system, hardware platform, computer system, or similar development platform [9].

We made our mobile Facebook Application which includes different function such as login, posting status on wall, logout by using Facebook SDK. We receives unique APP_ID from Facebook Developer Page shown in figure 2, it help us to operate our application normally [8]. Packet transmitted from mobile client is encrypted and we analyzed transmitted data when posting letters. Information of each method is stored variable of BUNDLE type at android.os. At the same time, HTTP header option is made and determined through Util.java of Facebook SDK. Facebook uses HTTP, when we posting status on Facebook wall. In this, client request message to sever and server response message back to client. Request line, General Headers, Request Headers, Entity Headers, Empty Line, and Message Body are consisting in Request Message of HTTP protocol. Request line contains 3 purposes that client use, (<method> <request URI> <HTTP version>). When server responses to client request; general Headers are used to control the processing of the message, and to provide additional information to the recipient. Entity Headers describe entity via message.

In March 2010, Graph API was launched which is a core of Facebook Platform, with the intention of replacing the older REST API. Graph API is a Facebook developer tool, which is used to generate access token. Access token permits users to use different features of Facebook. It is the primary way for apps which provide developer to read and write data to the Facebook social graph and uniformly representing objects in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend relationships, shared content, and photo tags).

It links objects together using their relationships via https://graph.facebook.com/<ID>/<CONNECTION_TYPE>?access_token=<ACCESS_TOKEN>.CONNECTION_TYPE means relation between objects such as Friends, LIKE, PERMISSION [9]. Requesting data, posting data, and deleting data are three essential methods of using the Graph API. Each object has its unique object ID. To obtain information of Facebook ID; send request query via https://graph.facebook.com /<FACEBOOK ID and for more object information; Access_Token is necessary because permission is needed for using each the content in Facebook.



https://developers.facebook.com/tools/explorer/145634995501895/?method=GET&path=me%3Ffields%3Did%2Cname&version=v2.2

Graph API Explorer

Application: Graph API Explorer · Locale: English (US) · API Version: v2.2

Select Permissions

User Data Permissions · Extended Permissions

□ ads_management □ ads_read □ email
□ manage_notifications □ manage_pages ☑ publish_actions
□ read_friendlists □ read_insights □ read_mailbox
□ read_page_mailboxes □ read_stream □ rsvp_event

Public profile included by default

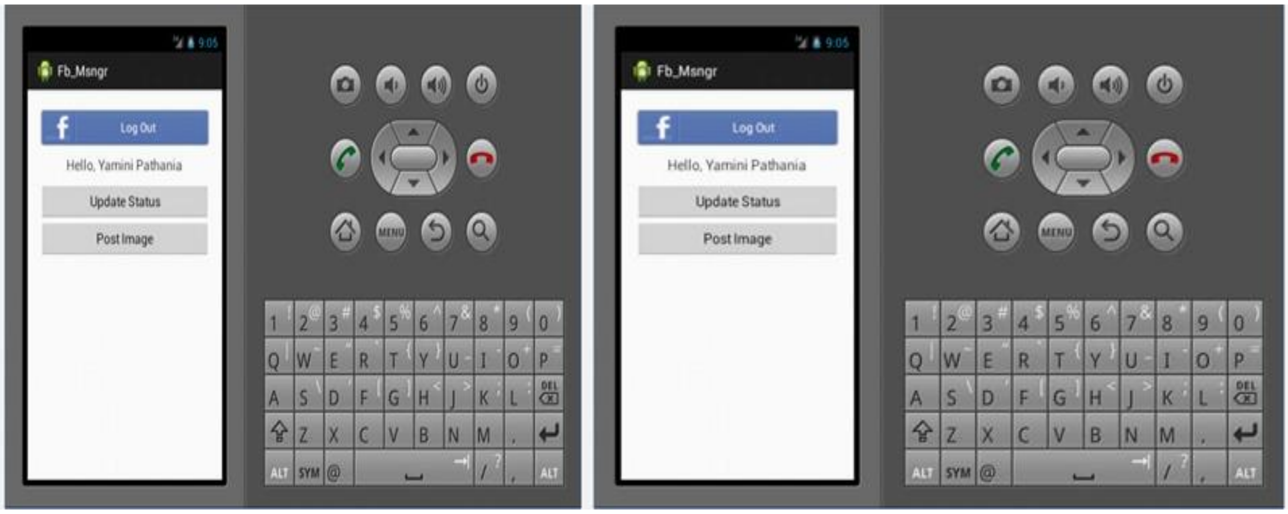
Node: me
☑ id
☑ name

Fig 2: Get the permission through Graph API

In mobile Facebook application, Access_Token provide authority to be issued upon login. In case of mobile Facebook application, 'permit' needs to be obtained to import data of each requested contents. USER_DATA_PERMMISION enables access to user's own information. For example, if we update status on Facebook wall, then "Publish_actions" access token must be required, which provide App ID shown in Fig 2. It gives us permission to post our status form mobile Facebook android application.

4. EXPERIMENTAL RESULTS

We have an experiment on Test Facebook application to know the format of message and its exchange request. So, firstly we login on the Facebook account to provide the application ID through graph API to the user, from above figure 2; when we login into the developed Facebook application shown in Fig 6.1(a).



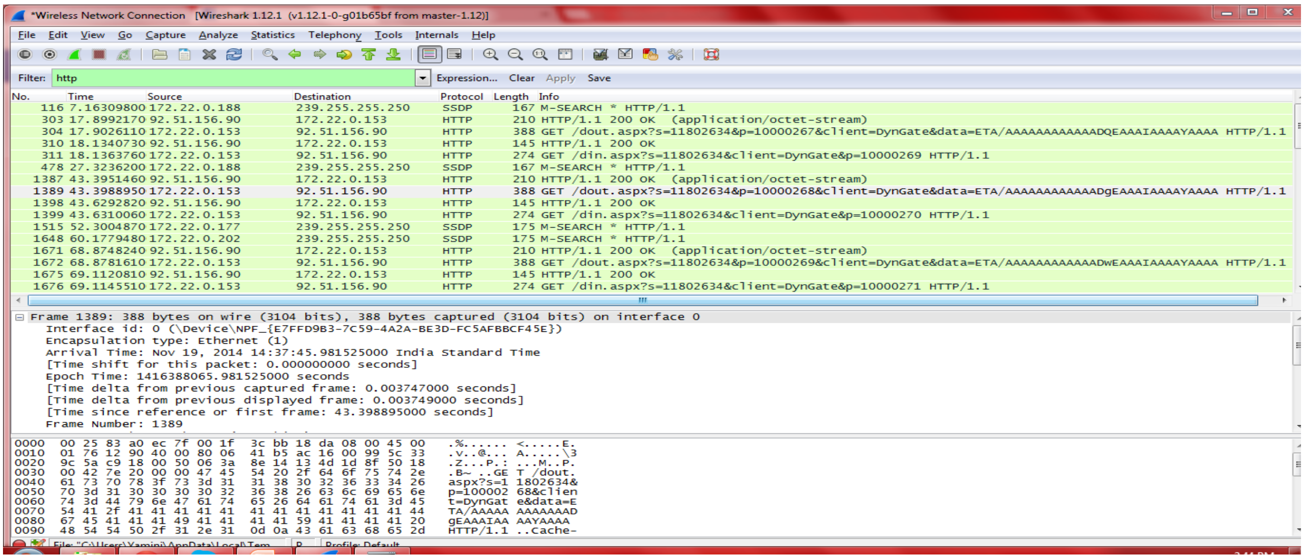
(a) (b)

Fig 3.1 (a) Login Facebook Account and (b) upload Photo on Facebook wall

Above Fig 3.1(b) shows that, we upload the status or photo on the Facebook wall then we capture the packets through Wireshark tool. Wireshark is the network

.

capture analyzer, which capture the live packets from the network


*Wireless Network Connection [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: http

```
No.     Time       Source         Destination      Protocol Length Info
    116 7.16309800 172.22.0.188   239.255.255.250  SSDP      167 M-SEARCH * HTTP/1.1
    303 17.8992170 92.51.156.90   172.22.0.153     HTTP      210 HTTP/1.1 200 OK  (application/octet-stream)
    304 17.9026110 172.22.0.153   92.51.156.90     HTTP      388 GET /dout.aspx?s=11802634&p=10000267&client=DynGate&data=ETA/AAAAAAAAAAAADQEAAAIAAAAYAAAA HTTP/1.1
    310 18.1340730 92.51.156.90   172.22.0.153     HTTP      145 HTTP/1.1 200 OK
    311 18.1363760 172.22.0.153   92.51.156.90     HTTP      274 GET /din.aspx?s=11802634&client=DynGate&p=10000269 HTTP/1.1
    478 27.3236200 172.22.0.188   239.255.255.250  SSDP      167 M-SEARCH * HTTP/1.1
   1387 43.3951460 92.51.156.90   172.22.0.153     HTTP      210 HTTP/1.1 200 OK  (application/octet-stream)
   1389 43.3988950 172.22.0.153   92.51.156.90     HTTP      388 GET /dout.aspx?s=11802634&p=10000268&client=DynGate&data=ETA/AAAAAAAAAAAADgEAAAIAAAAYAAAA HTTP/1.1
   1398 43.6292820 92.51.156.90   172.22.0.153     HTTP      145 HTTP/1.1 200 OK
   1399 43.6310060 172.22.0.153   92.51.156.90     HTTP      274 GET /din.aspx?s=11802634&client=DynGate&p=10000270 HTTP/1.1
   1515 52.3004870 172.22.0.177   239.255.255.250  SSDP      175 M-SEARCH * HTTP/1.1
   1648 60.1779480 172.22.0.202   239.255.255.250  SSDP      175 M-SEARCH * HTTP/1.1
   1671 68.8748240 92.51.156.90   172.22.0.153     HTTP      210 HTTP/1.1 200 OK  (application/octet-stream)
   1672 68.8781610 172.22.0.153   92.51.156.90     HTTP      388 GET /dout.aspx?s=11802634&p=10000269&client=DynGate&data=ETA/AAAAAAAAAAAADwEAAAIAAAAYAAAA HTTP/1.1
   1675 69.1120810 92.51.156.90   172.22.0.153     HTTP      145 HTTP/1.1 200 OK
   1676 69.1145510 172.22.0.153   92.51.156.90     HTTP      274 GET /din.aspx?s=11802634&client=DynGate&p=10000271 HTTP/1.1
```

```
⊟ Frame 1389: 388 bytes on wire (3104 bits), 388 bytes captured (3104 bits) on interface 0
    Interface id: 0 (\Device\NPF_{E7FFD9B3-7C59-4A2A-BE3D-FC5AFBBCF45E})
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 19, 2014 14:37:45.981525000 India Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1416388065.981525000 seconds
    [Time delta from previous captured frame: 0.003747000 seconds]
    [Time delta from previous displayed frame: 0.003749000 seconds]
    [Time since reference or first frame: 43.398895000 seconds]
    Frame Number: 1389
```

```
0000  00 25 83 a0 ec 7f 00 1f  3c bb 18 da 08 00 45 00   .%...... <.....E.
0010  01 76 12 90 40 00 80 06  41 b5 ac 16 00 99 5c 33   .v..@... A.....\3
0020  9c 5a c9 18 00 50 06 3a  8e 14 13 4d 1d 8f 50 18   .Z...P.: ...M..P.
0030  00 42 7e 20 00 00 47 45  54 20 2f 64 6f 75 74 2e   .B~ ..GE T /dout.
0040  61 73 70 78 3f 73 3d 31  31 38 30 32 36 33 34 26   aspx?s=1 1802634&
0050  70 3d 31 30 30 30 30 32  36 38 26 63 6c 69 65 6e   p=100002 68&clien
0060  74 3d 44 79 6e 47 61 74  65 26 64 61 74 61 3d 45   t=DynGat e&data=E
0070  54 41 2f 41 41 41 41 41  41 41 41 41 41 41 41 44   TA/AAAAA AAAAAAAD
0080  67 45 41 41 41 49 41 41  41 41 59 41 41 41 41 20   gEAAAIAA AAYAAAA
0090  48 54 54 50 2f 31 2e 31  0d 0a 43 61 63 68 65 2d   HTTP/1.1 ..Cache-
```

Fig 3.2 Capture the data through Wireshark

The given Fig. 3.2 show our experimental results, when packet is capture through Wirshark. By using Wireshark we know the whole detail about the HTTP protocol in which format it transfer the data in form client server computing model. It helps user to get the information about in which format the message is exchange over the Facebook client and Facebook server.

5. CONCLUSION AND FUTURE SCOPE

Our objectives were present the result of protocol reverse engineering which shows the Facebook message format and is exchange sequences by using passive monitoring approach. We have done our analysis by developing Facebook android application using Facebook SDK and Graph API. The results will provide the exact format of the Facebook message and its exchange request and we were able to know how the messages are exchanged when it arrived. Our proposed approach can also deduce useful additional features (i.e. comment, post photo, like, message etc.). But there is also a limitation of this approach that anyone can hack Facebook user's account by getting the information via https://graph.facebook.com /<FACEBOOK ID. Our future work will be to develop a technique which helps us to protect Facebook user's account from hackers.

REFERENCES

[1] Zhi Wang, Xuxian Jiang, Weidong Cui2, Xinyuan Wang, and Mike Grace. “"ReFormat: Automatic reverse engineering of encrypted messages." Computer Security–ESORICS 2009. Springer Berlin Heidelberg, 2009. 200-215.

[2] Li, Xiangdong, and Li Chen. "A survey on methods of automatic protocol reverse engineering." Computational Intelligence and Security (CIS), 2011 Seventh International Conference on. IEEE, 2011.

[3] WANG, Ying, GU Li-ze, LI Zhong-xian, YANG Yi-xian. "Protocol reverse engineering through dynamic and static binary analysis." The Journal of China Universities of Posts and Telecommunications 20 (2013): 75-79.

[4] Yong Wang, Nan Zhang, Yan-mei Wu, Bin-bin Su, Yong-jian Liao. “Protocol Formats Reverse Engineering based on Association Rules in Wireless Environment.” Security and Privacy in Computing and Communications, 2013 12th International Conference on Trust, IEEE, 2013.

[5] Pan, Fan, et al. "Efficient Protocol Reverse Method Based on Network Trace Analysis." JDCTA: International Journal of Digital Content Technology and its Applications 6.20, 2012.

[6] Cui, Weidong, Jayanthkumar Kannan, and Helen J. Wang. "Discoverer: Automatic Protocol Reverse Engineering from Network Traces." USENIX Security. 2007.

[7] Shevertalov, Maxim, and Spiros Mancoridis. "A reverse engineering tool for extracting protocols of networked applications, 14th Working Conference on Reverse Engineering. IEEE, 2007.

[8] Graph Api, ttp://developers.facebook.com/docs/reference/api/, 2012.

[9] F. Muller, F. Thiesing, “Social Networking APIs for companies,” Computational Aspects of Social Networks (CASoN), Oct. 2011.