A

Project Report On

SOCIAL ENGINEERING ATTACK BY PROFILE HIJACKING AND USING HUMANS AS BOTNETS

Submitted By

Mayank Jain	**B8058578**
Rahul Patil	**B8058598**
Anjali Shira	**B8058622**

Under the guidance of

Prof. S.C. Dharmadhikari

In partial fulfilment of

Bachelor of Engineering

[B. E. Information Engineering]

[May 2012]

AT

Department of Information Technology

Pune Institute of Computer Technology

Dhankawadi, Pune 411043

Affiliated to

University of Pune

Pune Institute of Computer Technology

Department of Information Technology
Dhankawadi, Pune 411043

CERTIFICATE

This is certify that the Dissertation entitled **"Social Engineering Attack By Profile Hijacking and Using Humans as botnets",** submitted by **Mr. Mayank Jain** is a record of bonafide work carried out by him, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Information Technology) at Pune Institute of Computer Technology, Pune under the University of Pune. This work is done during year 2011-2012, under our guidance.

_____

(Prof. S.C. Dharmadhikari)

Project Guide

_____ _____-

Prof. Emmanual M. Dr. P. T. Kulkarni

HOD, IT Department **Principal PICT**

Examination:

Examiner _____

Date:

Acknowledgements

I am profoundly grateful to **Prof. S.C. Dharmadhikari** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

I would like to express deepest appreciation towards **Dr. P. T. Kulkarni**, Principal PICT, Pune, **Prof. Emmanual M.** HOD Information Technology Department and **Prof. Manish R. Khodaskar** (Project Coordinator) whose invaluable guidance supported me in completing this project.

I am particularly grateful to **Dr. Navin Kabra** (Punetech) who allows me to work in the company.

At last I must express my sincere heartfelt gratitude to all the staff members of Information Technology Department who helped me directly or indirectly during this course of work.

Mayank Jain

Rahul Patil

Anjali Shira

CERTIFICATE

This is to certify that the project report entitled

Social Engineering Attack By Profile Hijacking and Using Humans as botnets

Submitted by

Mayank Jain B8058578
Rahul Patil B8058598
Anjali Shira B8058622

is a bonafide work carried out by them with the Sponsorship from ————————- under the supervision of Mr. ..... and has been completed successfully .

(Mr. .....)
(Designation)
External Guide

Place : Pune
Date:

ABSTRACT

Social Engineering attack is a major security threat that needs to be dealt. The only way to counter attack is to raise awareness among the users of what all is possible in Social engineering attack and how it can be done. We want to build a tool that will use Social Networking sites like facebook as a platform to harnest the users data available and use that to impersonate them via profile hijacking. Then we intend to use man in the middle attack and make them reveal some information which is not available as a public data. To test this we will use a modified version of the turing test.

Keywords : Security, Social Networks, Facebook, Man in the middle attack

Contents

List of Figures

Chapter 1

Introduction

1.1 Need

1.1.1 Social Engineering

Social Engineering is the act of manipulating a person to take an action that may or may not be in the targets best interest. This may include obtaining information, gaining access, or getting the target to take certain action[5]

1.1.2 Current Scneario

Businesses spend a significant portion of their annual information technology budgets on high-tech computer security. But the firewalls, vaults, bunkers, locks and biometrics those dollars buy can be pierced by attackers targeting untrained, uninformed or unmonitored users. Humans are the weakest link in any security system, according to KL-based organizers of the Hackers Halted Asia Pacific 2009 conference.[6]

The chief minister of Malacca, Datuk Seri Haji Mohd Ali Bin Mohd Rustam, said there is no perfect system in the world. Even if you have the best security devices and softwareyour organization still relies on humanswho are the weakest link in any security system. Public education and awareness is essential.

Figure 1.1 is a Data Security Breach Statistics of 2008 revealing that Malicious Insider and Careless/Untrained Insider is a bigger threat than an outside cracker.

Also, in the age of Social Networking Sites like Facebook, Twitter, Linkedin, Google+ information of companies internal hierarchy structure, employees, personal information is readily available online for the information gathering phase to breach companies security perimeter by various social engineering ways like phishing links,

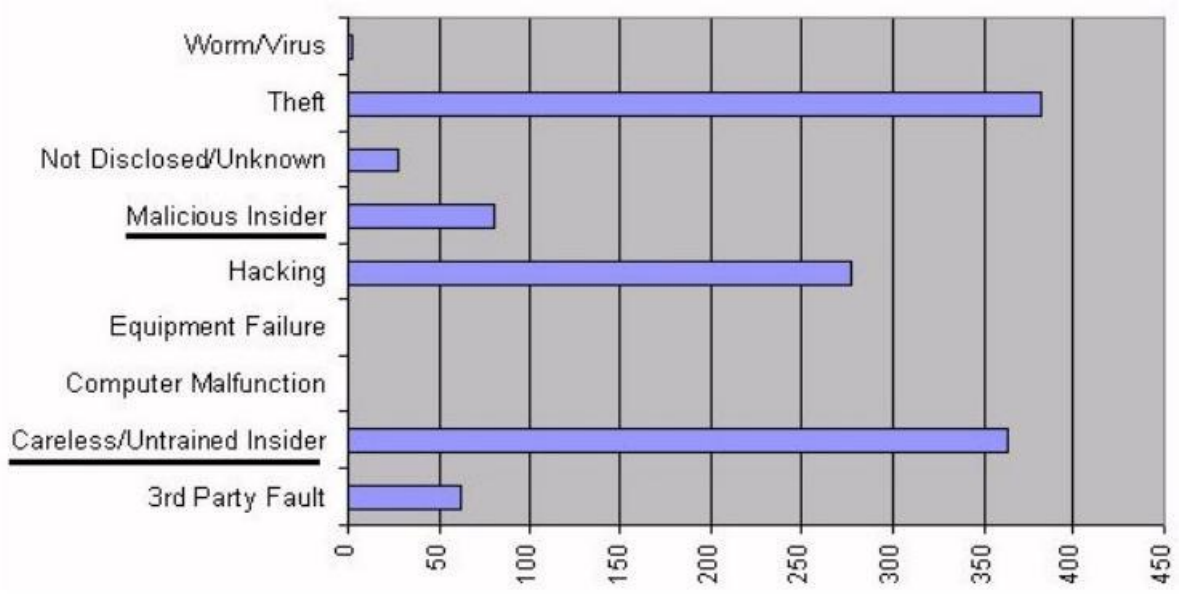

Figure 1.1: Figure 1.1: Data Security Breach Statistics of 2008

Trojans, Backdoors, Password guessing, breaking security questions etc.

Is it possible to break into the circle of friends network on social networking sites and make them reveal certain information about them which can be used further to infiltrate the security perimeter of a company?

Our project is designed to answer this Question and provide a Proof of Concept that Yes, it can be done and without even raising any suspicion on the target for a long time.

1.2 Basic Concept

In this project we will use two main concepts:

1.2.1 Profile Hijacking

In Profile Hijacking, we will impersonate the target(s) profile on Social Networking Sites like Facebook and use their identity to infiltrate the network of other targets friend to fit in the group. Profile Hijacking is important, to use them to reveal certain information without their knowledge.[2]

1.2.2 Using Humans as Botnets

We studied many AI based Chat Bots that are available on the web and we found that Using Chat Bots to chat with humans for Social Engineering Attack is not feasible

as Humans detect that the other person is not a human but a program and hence the attack fails even before it is launched.[4]

Since developing a fully convincing AI based Chat Bot is not possible considering the Scope, we will use another human to do the talking with our target while we sit in between, watch and modify the conversation towards the conversation which would reveal certain information which we are interested in.[1]

1.3 Application

Some of the best tools for fighting social engineering attacks are security awareness training and social engineering testing. The effectiveness of these controls will vary based on the quality of their implementation, including follow-up and retraining.

Social engineering testing, by its very nature, can be difficult to conduct without third-party assistance. One option is to engage an information security organization to conduct testing. The testing can uncover areas in which an organization is most vulnerable so that risk can be assessed and mitigation strategies can be formulated and implemented.

Rolling social engineering testing into a larger security penetration engagement can reduce the cost of the social engineering component, says Jim Patterson, director of consulting for Rapid7.[6]

Main Application of this Project is to develop a tool that will aid in doing Social Engineering Testing on Companies Employees, also it will provide as a live demonstration to employees under training on how social engineering can be done and how by being cautious one can prevent serious damage not only to the company but also to their private life.

Chapter 2

Literature Survey

2.1 Towards Automating Social Engineering Using Social Networking Sites

2.1.1 Summary

In this paper, we saw the use of artificial intelligence to create a chat bot to do the automatic social engineering attack via social networking sites. Experiments were done on a small group of people on facebook.

2.1.2 Advantages

- Automatic tool, minimal input required from the user.
- If perfected, could be the best way to automate social engineering attack.

2.1.3 Disadvantages

- Fails to convince that the chat bot is a real human.
- Has no input of real world information.
- Chat algorithm is hard coded using regular expressions.
- No real world testing because of ethical issues.
- Building a real world chat bot is out of scope.

2.2 All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks

2.2.1 Summary

Presents a novel concept of profile hijacking and cloning, which laid the foundation for the proposed idea.

2.2.2 Advantages

- Works with all social networking sites which has no authentication mechanism of who you say is who you are in real world.
- Very Easy to do, as most of the data required is already available.

2.2.3 Disadvantages

- Does not give us information which is not available directly.
- Impersonating or Faking a human identity is a punishable act.
- Real world testing has ethical issues.

2.3 Honeybot, Your Man in the Middle for Automated Social Engineering

2.3.1 Summary

This paper presents the basic idea we can do man in the middle attack on two users. The testing was done on public IRCs but was suggested that it can be done by using profile hijacking on a social networking sites like Facebook.

2.3.2 Advantages

- Results of success is much higher than by using artificial intelligence via a chat bot.
- Easy to build compared to building artificial intelligence.

2.3.3 Disadvantages

- Required gender conversion in IRCs chat when impersonating a different gender.
- Testing was done on IRCs which is not targetted information gathering.

2.4 Eight Friends Are Enough Social Graph Approximation via Public Listings

2.4.1 Summary

Presents a novel idea of Publically available data of users on social networking sites. Also presents the idea of Dominating Sets. It was also mentioned that the facebook friends public data policy keeps changing, before it was 10 friends than it became 8 and now again its back to 10.

2.4.2 Advantages

- Avoids detection by Facebook.
- Presents a novel idea of using Dominating Sets.

2.4.3 Disadvantages

- Is not succesful in the current scenario where users are security concious.
- Has very low sucess rate as per our experiments.

Chapter 3

Proposed Work

3.1 Problem Statement

To develop a tool which can be used for automated social engineering attack on users of social networking sites using profile hijacking and man in the middle attack. Also test the tool via modified version of Turing test at basic conversation level

3.2 Passive Information Gathering Module

3.2.1 Social Networks

A social networking service is an online service, platform, or site that focuses on building and reflecting of social networks or social relations among people, who, for example, share interests and/or activities and people with similar or somewhat similar interests, backgrounds and/or activities make their own communities.

A social network service consists of a representation of each user (often a profile), his/her social links, and a variety of additional services.

Here are some statistics about Facebook (Alexa estimates, as of 20/04/2012):-

- daily page views : 8.4 billion
- daily visitors : 650 million
- views per visitor : 12.9
- site rank : 2nd
- traffic fraction : 0.45% 1 in 220 of all web traffic

Worldwide	Unique Visitors	Percentage
Facebook.com	792,999	55.1%
Twitter.com	167,903	11.7%
LinkedIn.com	94,823	6.6%
Google Plus	66,756	4.6%
MySpace	61,037	4.2%
Others	255,539	17.8%
Total	1,438,877	100%

According to ComScore, Facebook has the largest number of Social Network User shares (up to end of November 2011)[7]

This shows that there is a lot of user generated information that is available to tap into. Facebook has its own API but we cannot use that to extract data about anyone. So we thought of writing a module which can do Web Scraping.

3.2.2 Web Scraping

Web scraping (also called web harvesting or web data extraction) is a computer software technique of extracting information from websites. Usually, such software programs simulate human exploration of the World Wide Web by either implementing low-level Hypertext Transfer Protocol (HTTP), or embedding a fully-fledged web browser, such as Internet Explorer or Mozilla Firefox.[8]

3.2.3 Features

This module will have the ability to scrape contents of facebook profile users page. It'll basically focus on the friends list of a user. The data generated from this module will help in focusing on who are the victims friends, what are the statistics of them etc.

Figure 3.1 is a screenshot of a user and his friends list

This module should be able to extract

- Full Name
- Gender : Male/Female/NA
- username : Username/NA

Figure 3.1: Friends List on Facebook

- Number of Friends : Total/NA
- Unique UserID

And in addition it should be able to extract each user's friends detail in a BFS Manner. And it can continue this as long as we want it to.

3.3 Facebook Chat Attack Module

3.3.1 Extensible Messaging and Presence Protocol

Extensible Messaging and Presence Protocol (XMPP) is an open-standard communications protocol for message-oriented middleware based on XML (Extensible Markup Language). The protocol was originally named Jabber, and was developed by the Jabber open-source community in 1999 for near-real-time, extensible instant messaging (IM), presence information, and contact list maintenance. Designed to be extensible, the protocol today also finds application in VoIP and file transfer signaling.[9]

Facebook uses XMPP (Extensible Messaging and Presence Protocol) for chatting.

3.3.2 Attack as Example

Following is an example.
Lets say we have two marks (targets) Alice (Human) - aliceHuman and Bob (Human) - bobHuman

We make two profiles both run by bots. The highlight is bothbots hijack the profilesof marks namely alice and bob.

So aliceBots profile is similar to aliceHumans profile and bobBots profile is simlar to bobBots profile. aliceBotsends a friend request to bobHuman. bobBotsends a friend request to aliceHuman. So now, aliceHuman is a friend of bobBot and bobHuman is a friend of aliceBot. Also, bobBot and aliceBot can exchange information outside of facebook to each other.

So now lets say, bobHuman and aliceHuman are online. Our bots start the conversation. Whatever is being passed to one of the Bot by one human, it is passed on to the other Bot and in turn passed on to the other human, i.e. two humans are having conversation through two bots but they think that they are talking to a human since the conversation sounds like a human (which it is).

After some amount of bonding between them our bots start modification by using injecting questions inside the conversations

AliceHuman - bobBot : Hey how was the movie yesterday?
botBot - aliceBot : Hey how was the movie yesterday?and hey btw whats your fav color?
aliceBot - BobHuman : Hey how was the movie yesterday? and hey btw whats your fav color?

BobHuman - aliceBot : movie was great, and its blue btw, whats yours?
aliceBot - bobBot : movie was great,you know my fav color is blue, what is yours?
bobBot - AliceHuman : movie was great, you know my fav color is blue, what is yours?

AliceHuman - bobBot : mine is pink :)
botBot - aliceBot : mine is pink :)
aliceBot - BobHuman : mine is pink :)

Notice how the conversation has been altered to make it unclear that no one asked each other about favorite color and yet both of them told us their favorite color. We got hold of two marks information without raising suspicion.

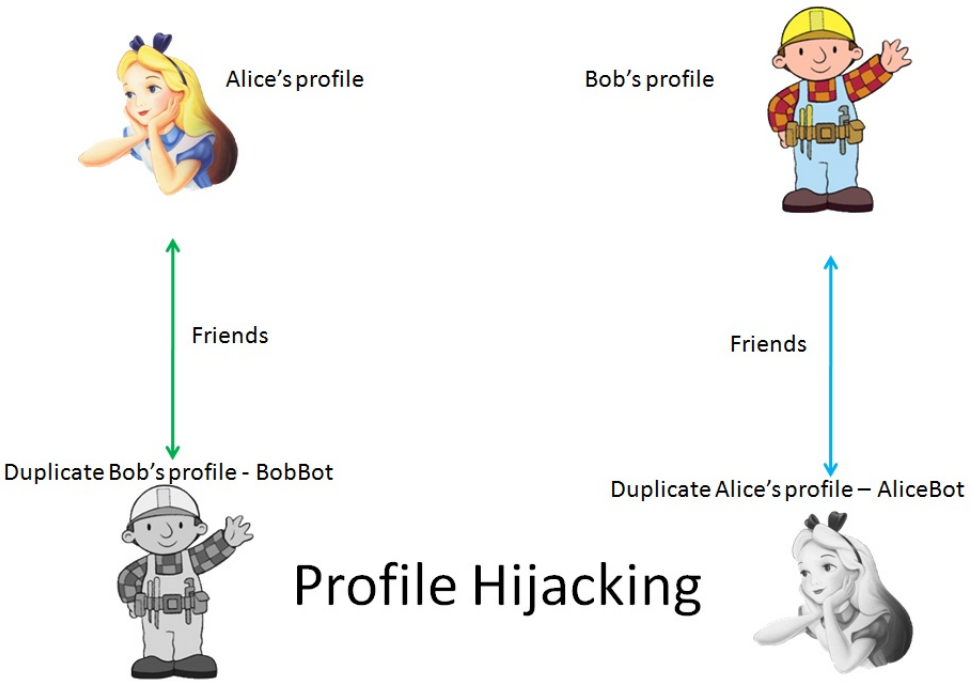

Figure 3.2: Example of Profile Hijacking

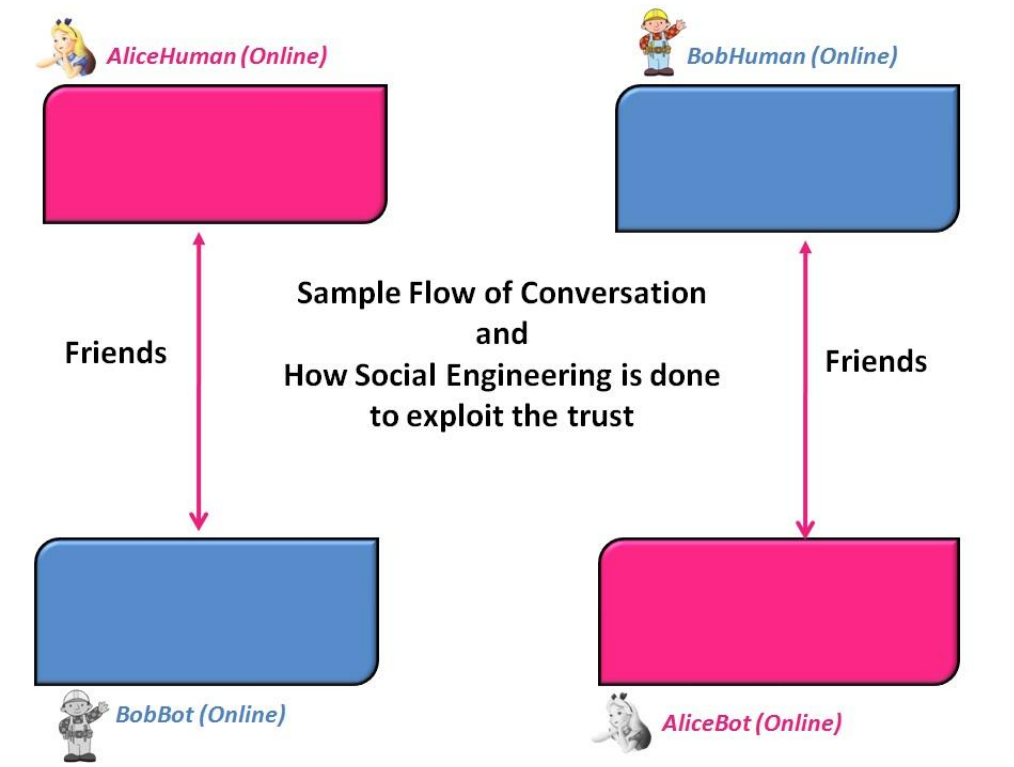

Figure 3.3: Example of Conversation Modification - 1

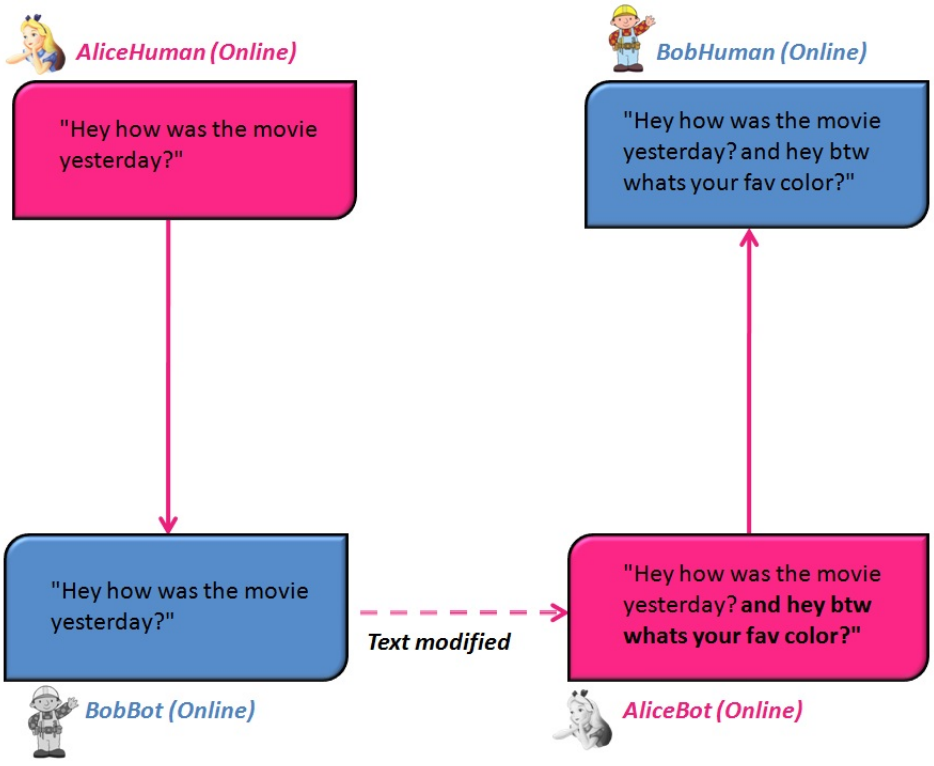

Figure 3.4: Example of Conversation Modification - 2

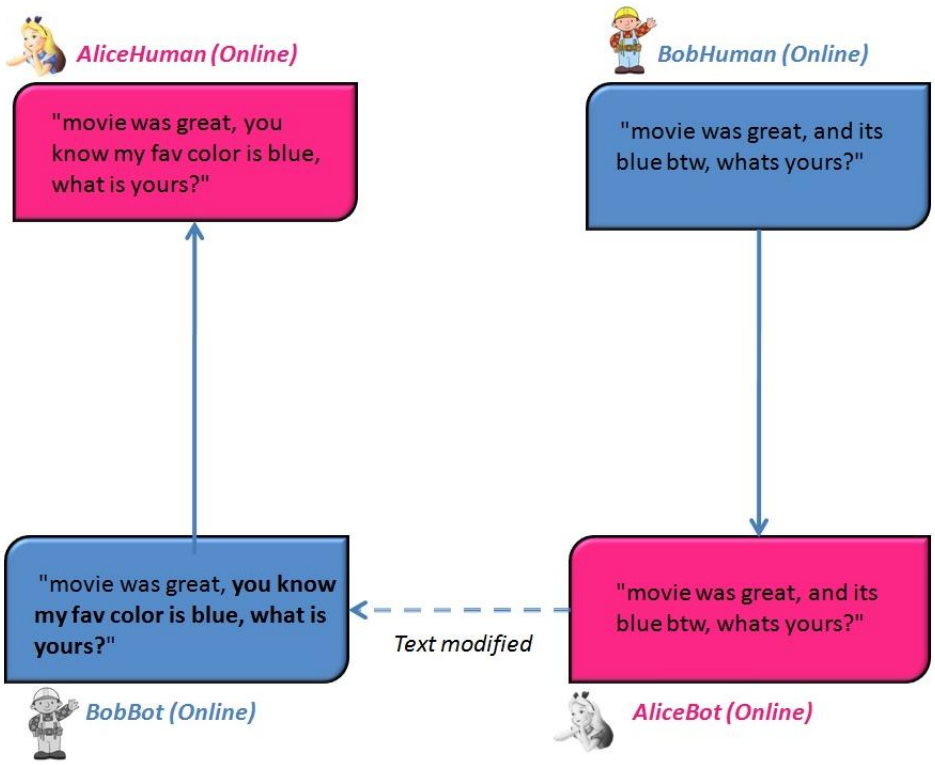

Figure 3.5: Example of Conversation Modification - 3

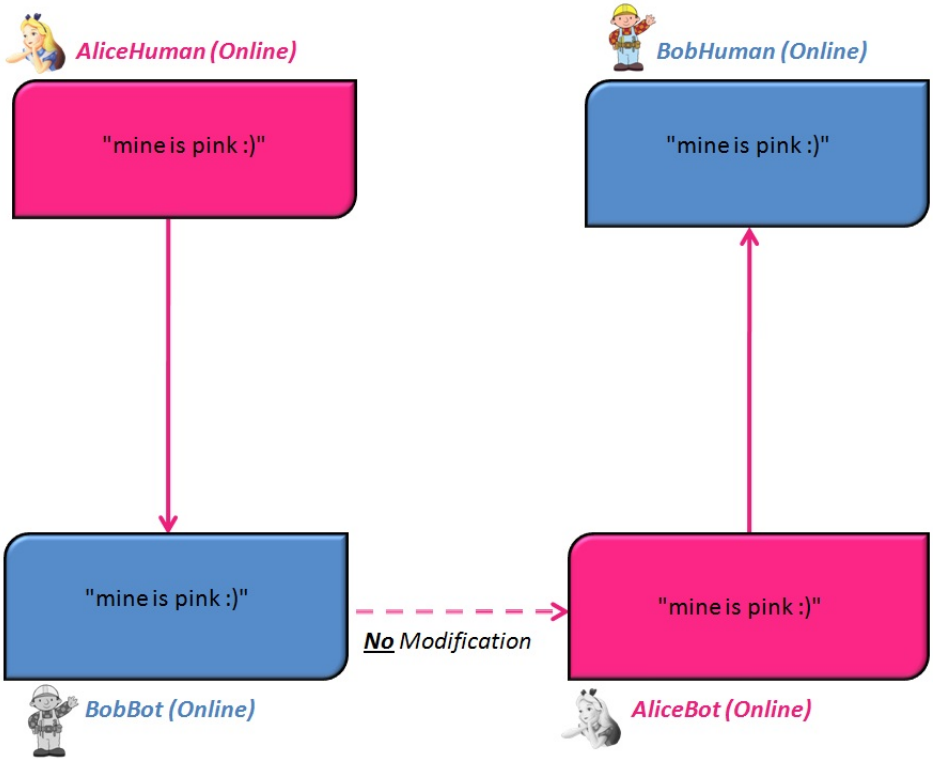

Figure 3.6: Example of Conversation Modification - 4

3.3.3 Constraints

We will develop a POC (Proof of Concept) only. To modify the conversation we will use simple regex based parsing to match and substitute it with matching strings. There will be no Artificial Inteligence or Machine Learning involved as that is out of scope at this level. Though it can be done and will make the modification more scalable and justified.

Chapter 4

Research Methodology

4.1 Passive Information Gathering Module

4.1.1 HTTP

To implement Web Scraping module for Information Gathering we need to understand the HTTP protocol. HTTP (The Hypertext Transfer Protocol) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Request methods

HTTP defines nine methods (sometimes referred to as ”verbs”) indicating the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

Following are the main request methods we will be dealing with :-

POST : Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

GET : Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.) The W3C has published guidance principles on this distinction, saying, ”Web application design should be informed by the above principles, but also by the relevant limitations.”

4.1.2 urllib2

Since we will be using Python 2.x for our project, we will be using the standard library for doing the HTTP requests to be made. The urllib2 module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world basic and digest authentication, redirections, cookies and more.[10]

urllib2.urlopen(url[, data][, timeout])

- Open the URL url, which can be either a string or a Request object.

4.1.3 Regular Expressions

A Regular Expression is the term used to describe a codified method of searching invented, or defined, by the American mathematician Stephen Kleene.[11]

4.2 Facebook Chat Attack

4.2.1 XMPPPY

xmpppy is a Python library that is targeted to provide easy scripting with Jabber. Similar projects are Twisted Words and jabber.py.

This library was not designed from scratch. It inherits some code from jabberpy and have very similar API in many places. Though it is separate project since it have almost completely different architecture and primarily aims to work with jabberd2 - the new Open Source Jabber Server.
We tried to use this library but we had issues[13][14]. Even though we were finally able to resolve the issue[15]

We didn't use this library due to insufficient documentation. The project was stuck for a while and we could not move it forward.

4.2.2 SleekXMPP

To resolve the issue of XMPP library we Found SleekXMPP.

SleekXMPP is an MIT licensed XMPP library for Python 2.6/3.1+

SleekXMPPs design goals and philosphy are:

Low number of dependencies : Installing and using SleekXMPP should be as simple as possible, without having to deal with long dependency chains.
As part of reducing the number of dependencies, some third party modules are included with SleekXMPP in the thirdparty directory. Imports from this module first try to import an existing installed version before loading the packaged version, when possible.

Every XEP as a plugin : Following Pythons batteries included approach, the goal is to provide support for all currently active XEPs (final and draft). Since adding XEP support is done through easy to create plugins, the hope is to also provide a solid base for implementing and creating experimental XEPs.

Rewarding to work with : As much as possible, SleekXMPP should allow things to just work using sensible defaults and appropriate abstractions. XML can be ugly to work with, but it doesnt have to be that way.

Chapter 5

Project Design

5.1 Use Case diagrams

5.1.1 Use Case - System Component

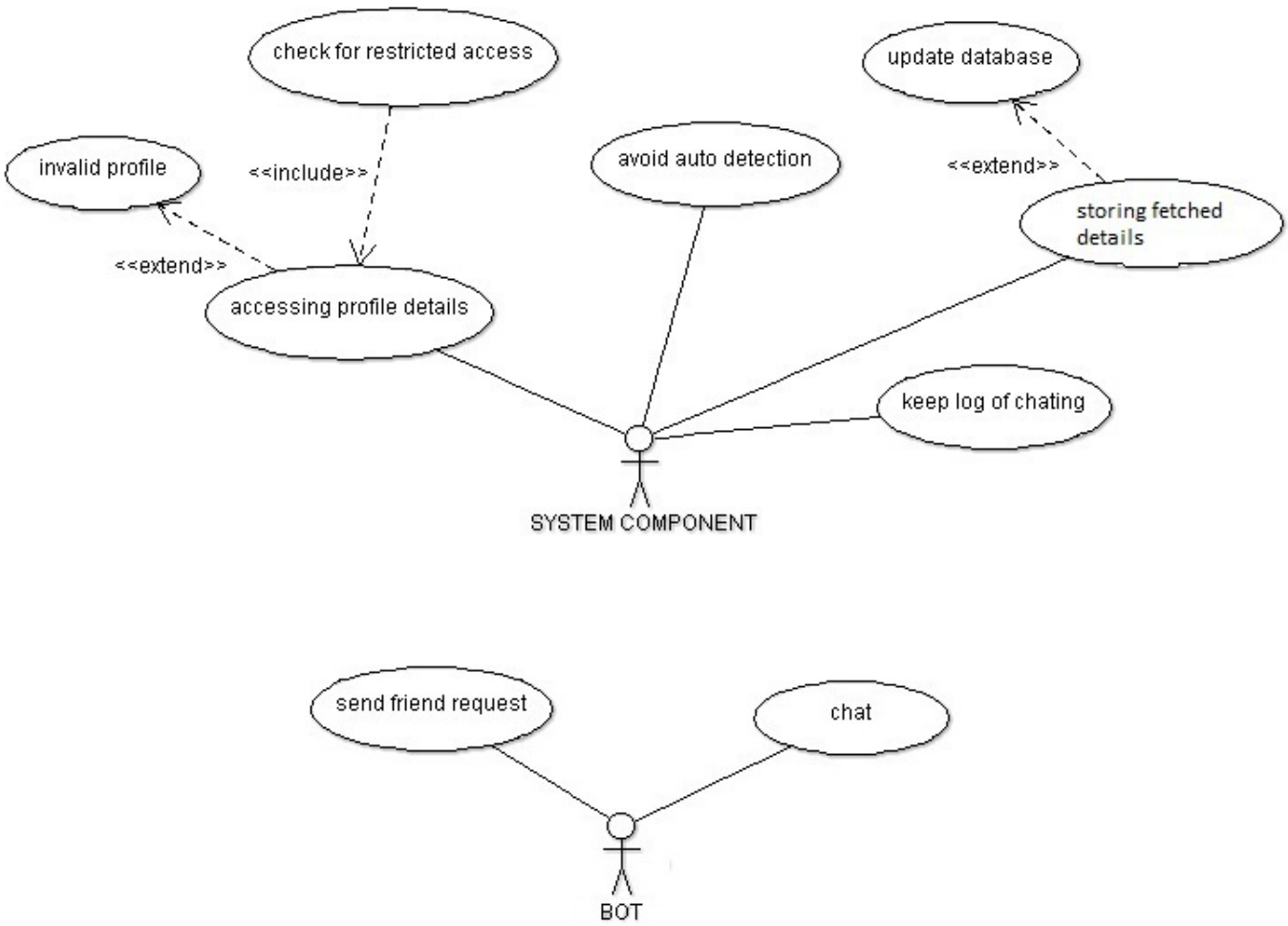

Figure 5.1: Use Case Diagram - System Component

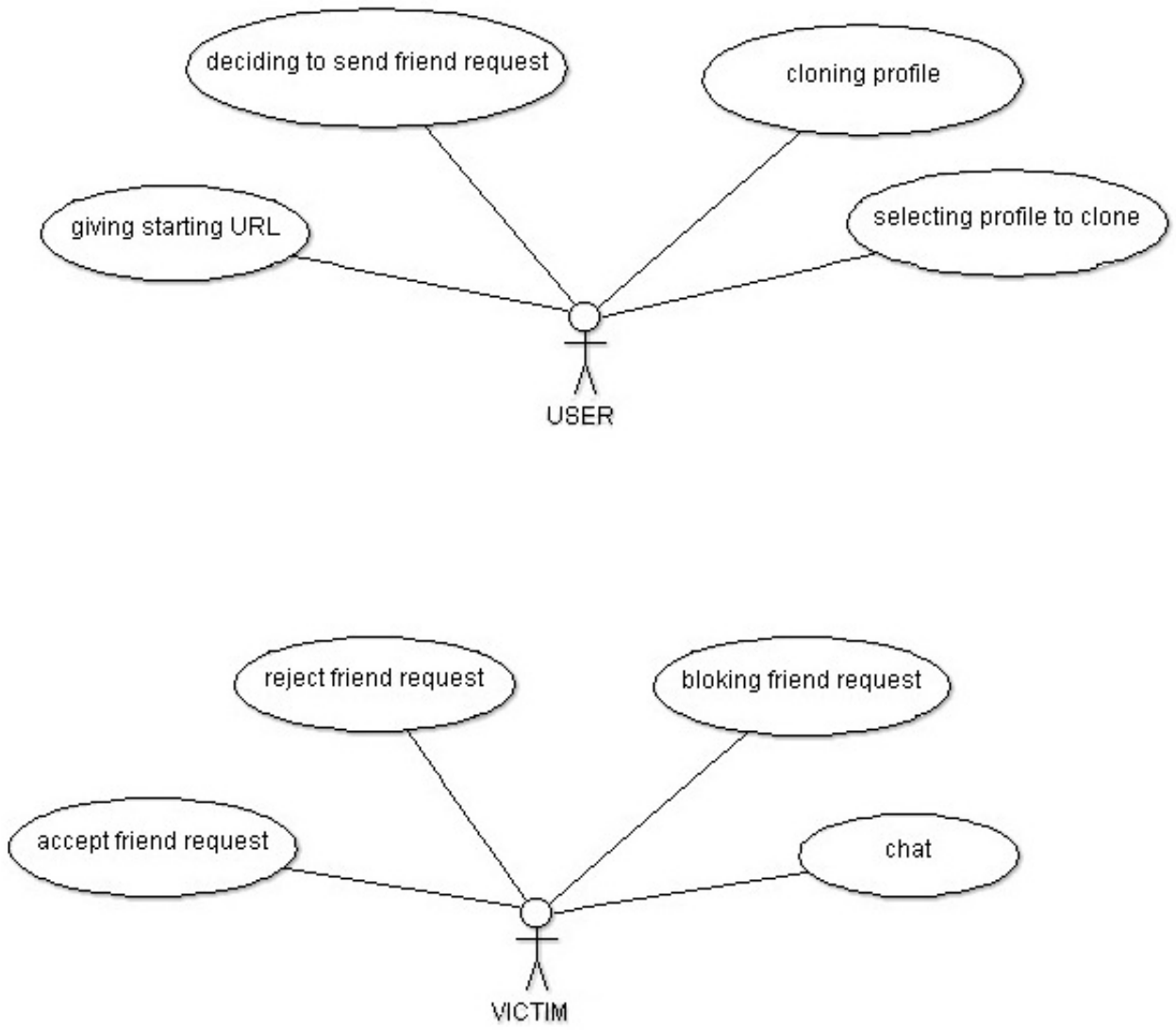

Figure 5.2: Use Case Diagram - Victim and User

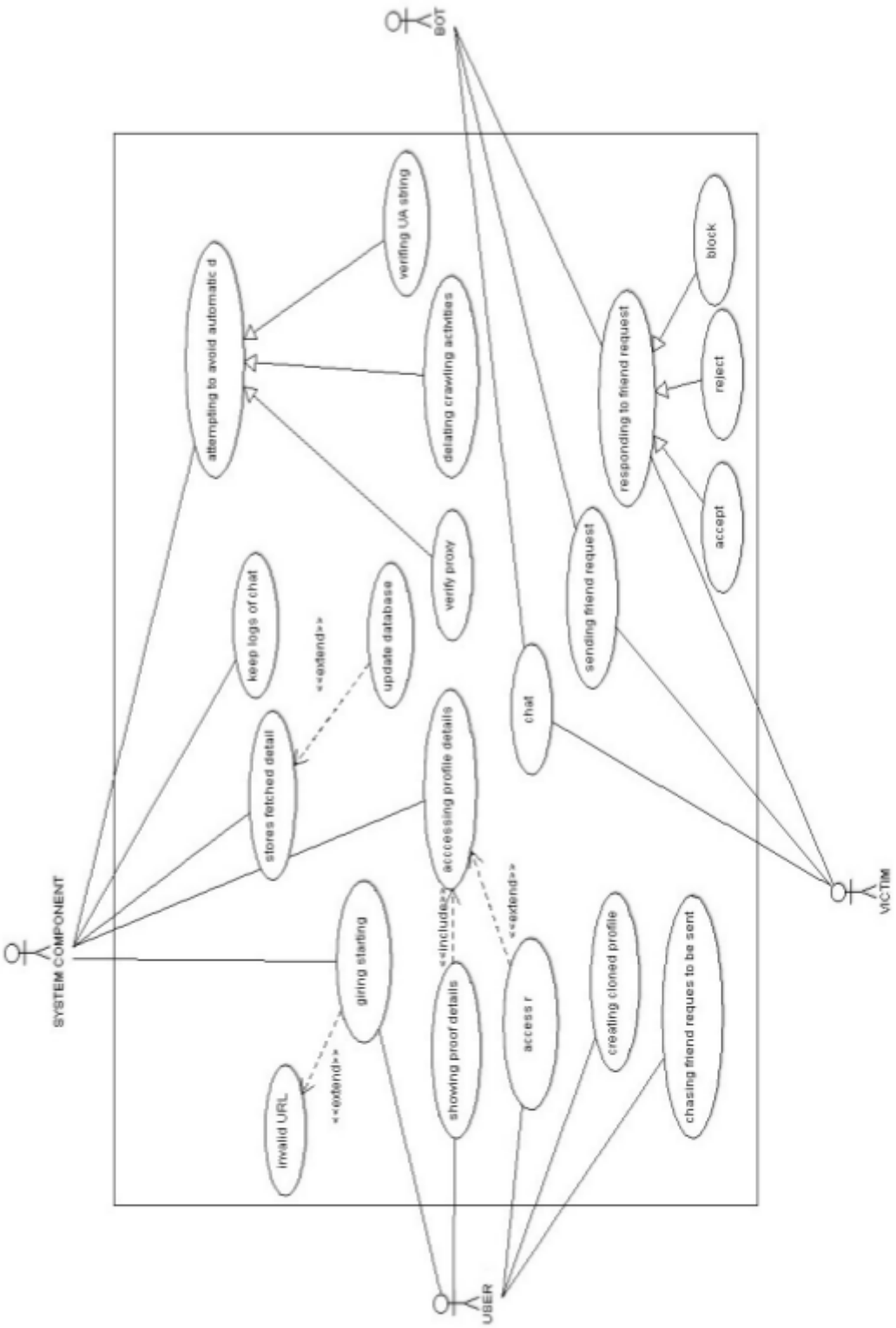

Figure 5.3: Use Case Diagram - Main

5.2 ER Diagram

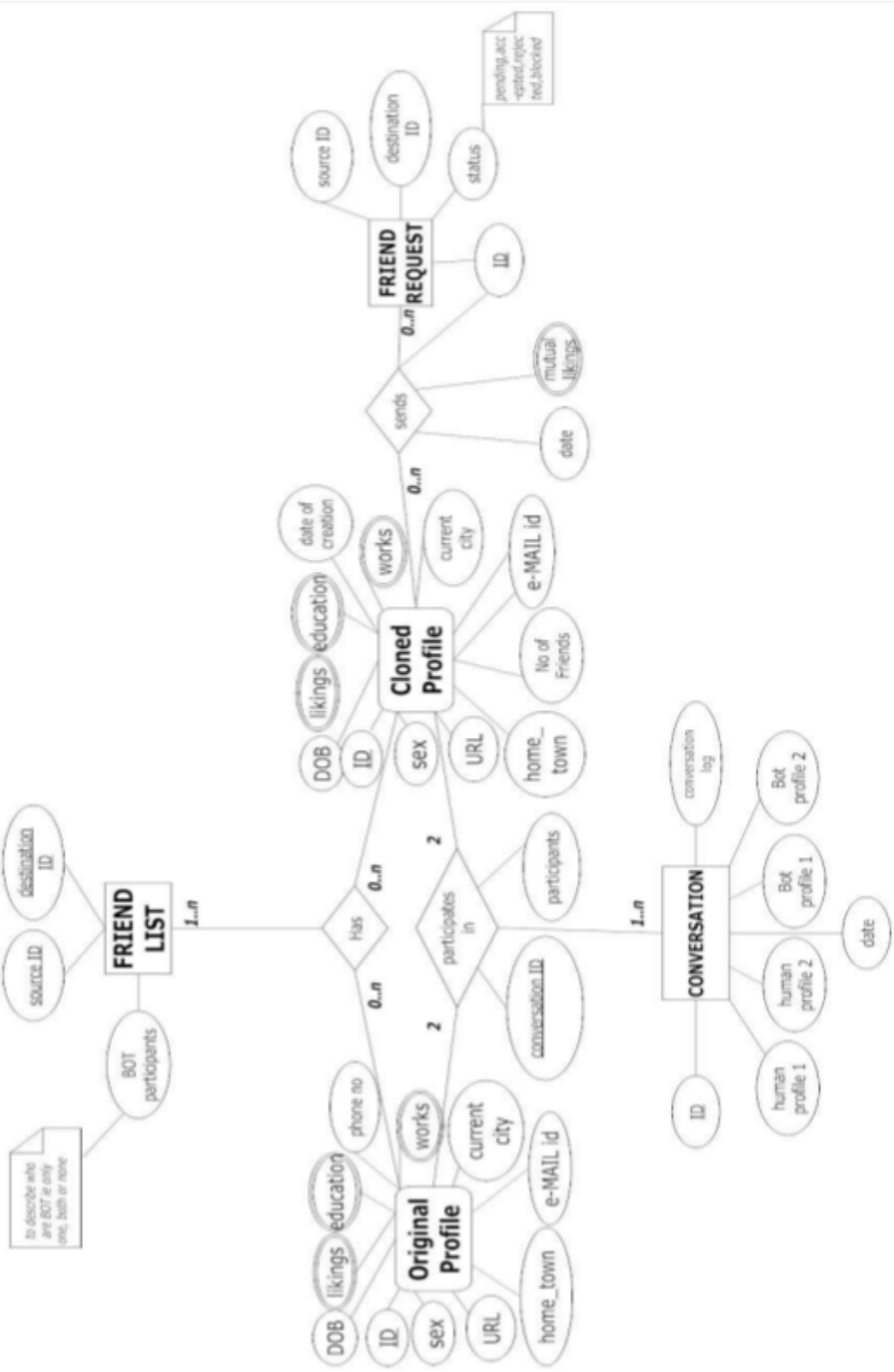

Figure 5.4: ER Diagram

5.3 DFD (level 0/1/2)

5.3.1 DFD Level 0

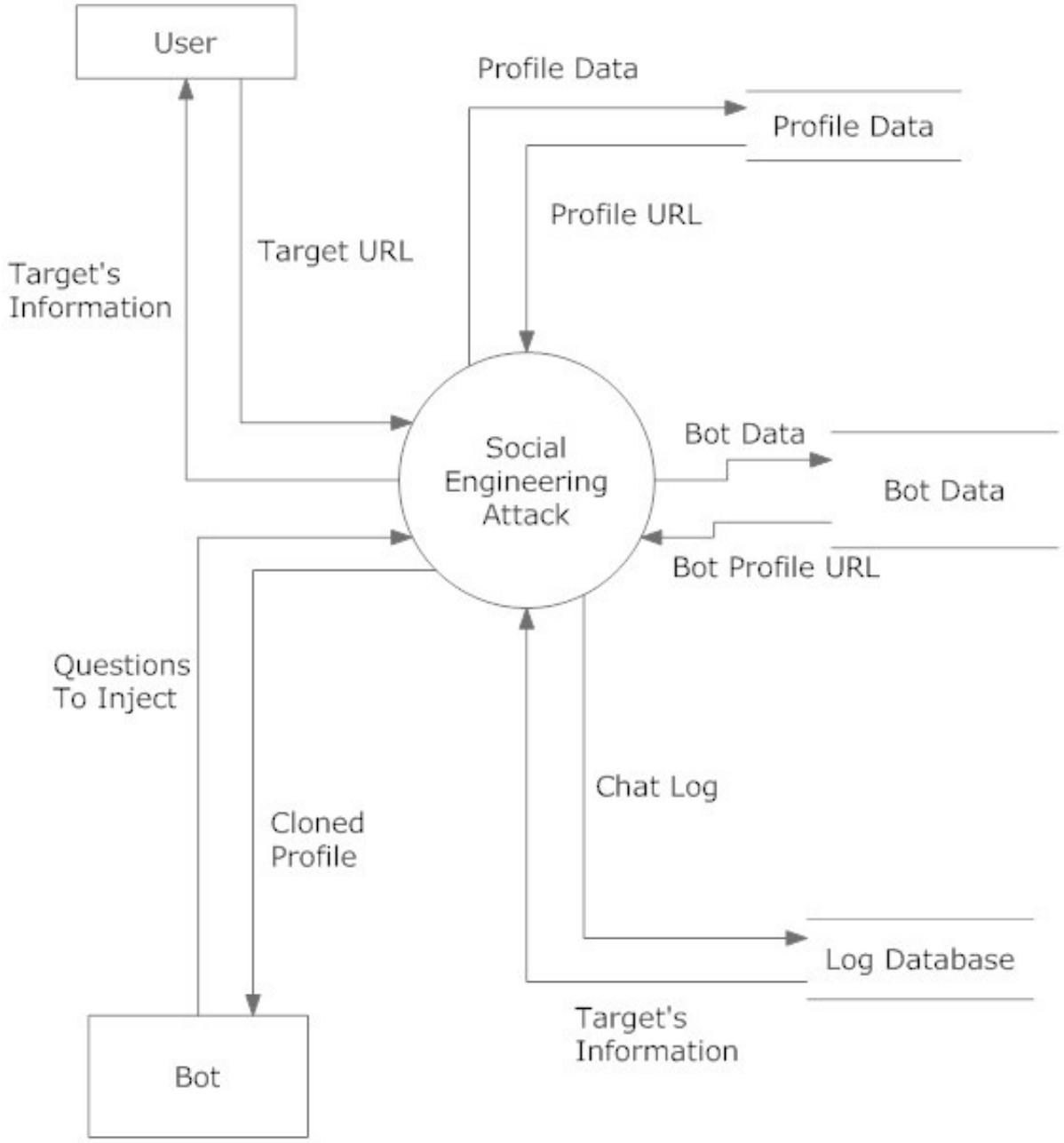

Figure 5.5: DFD - Level 0

5.3.2 DFD Level 1

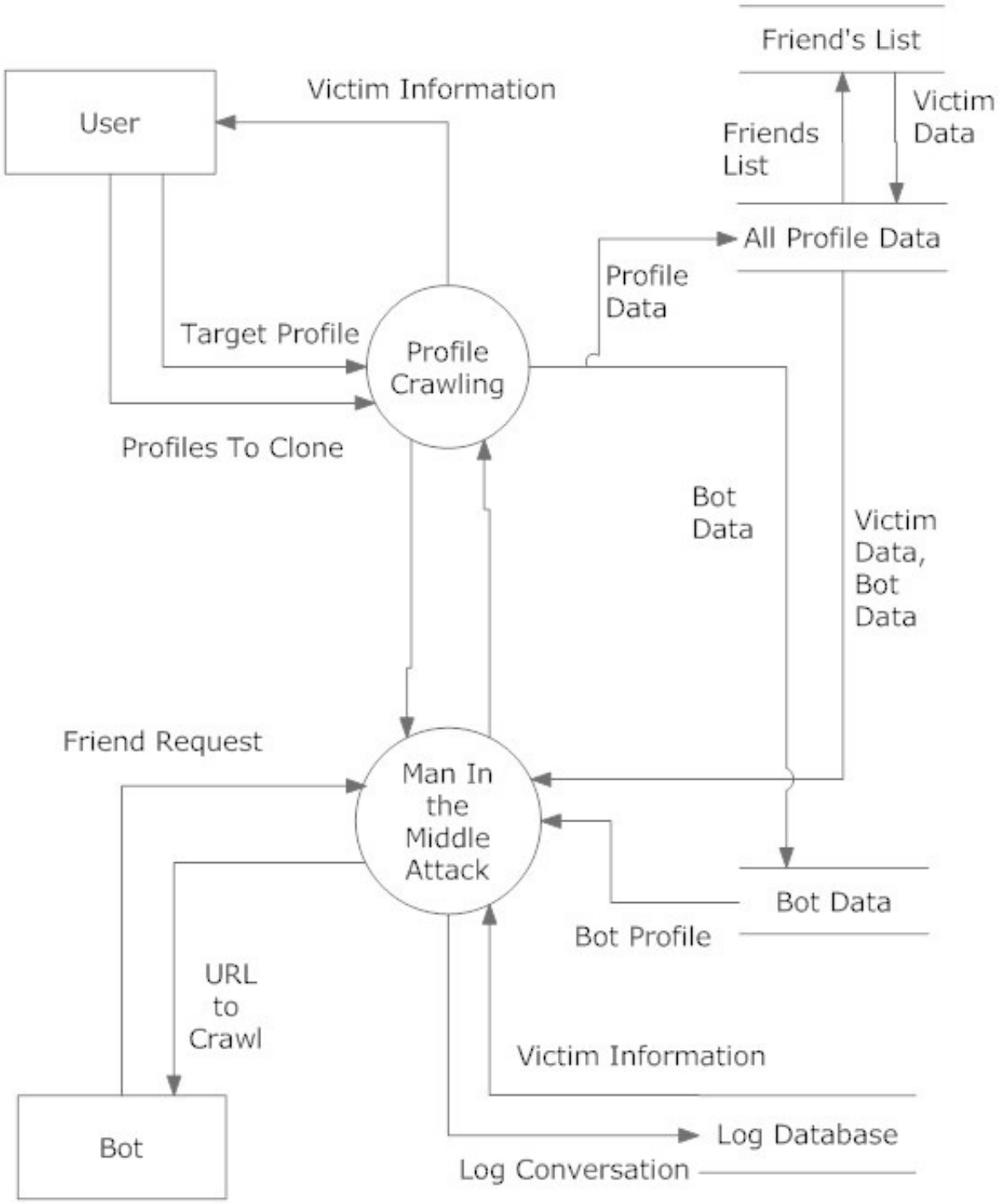

Figure 5.6: DFD - Level 1

5.3.3 DFD Level 2

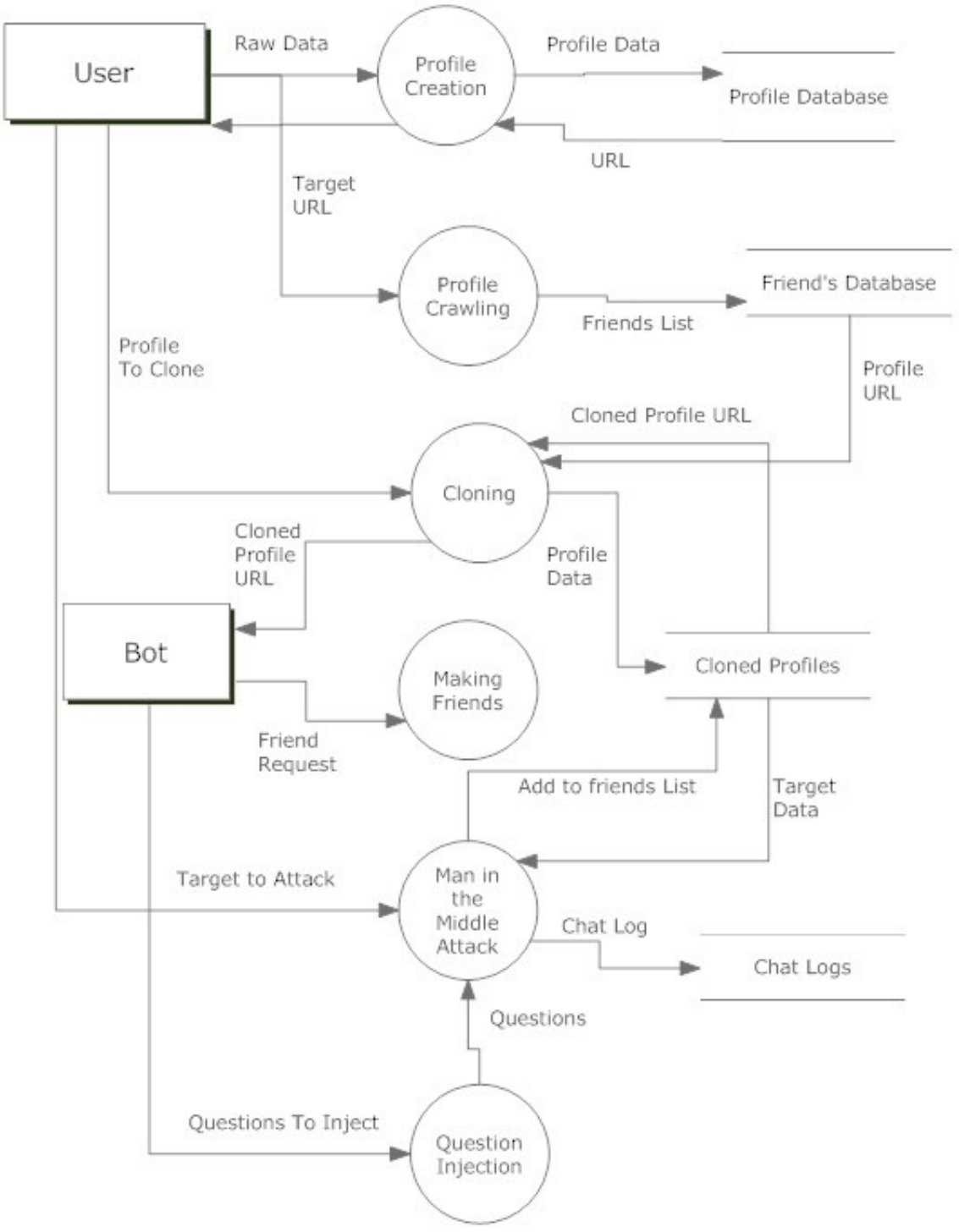

Figure 5.7: DFD - Level 2

5.4 Activity Diagram

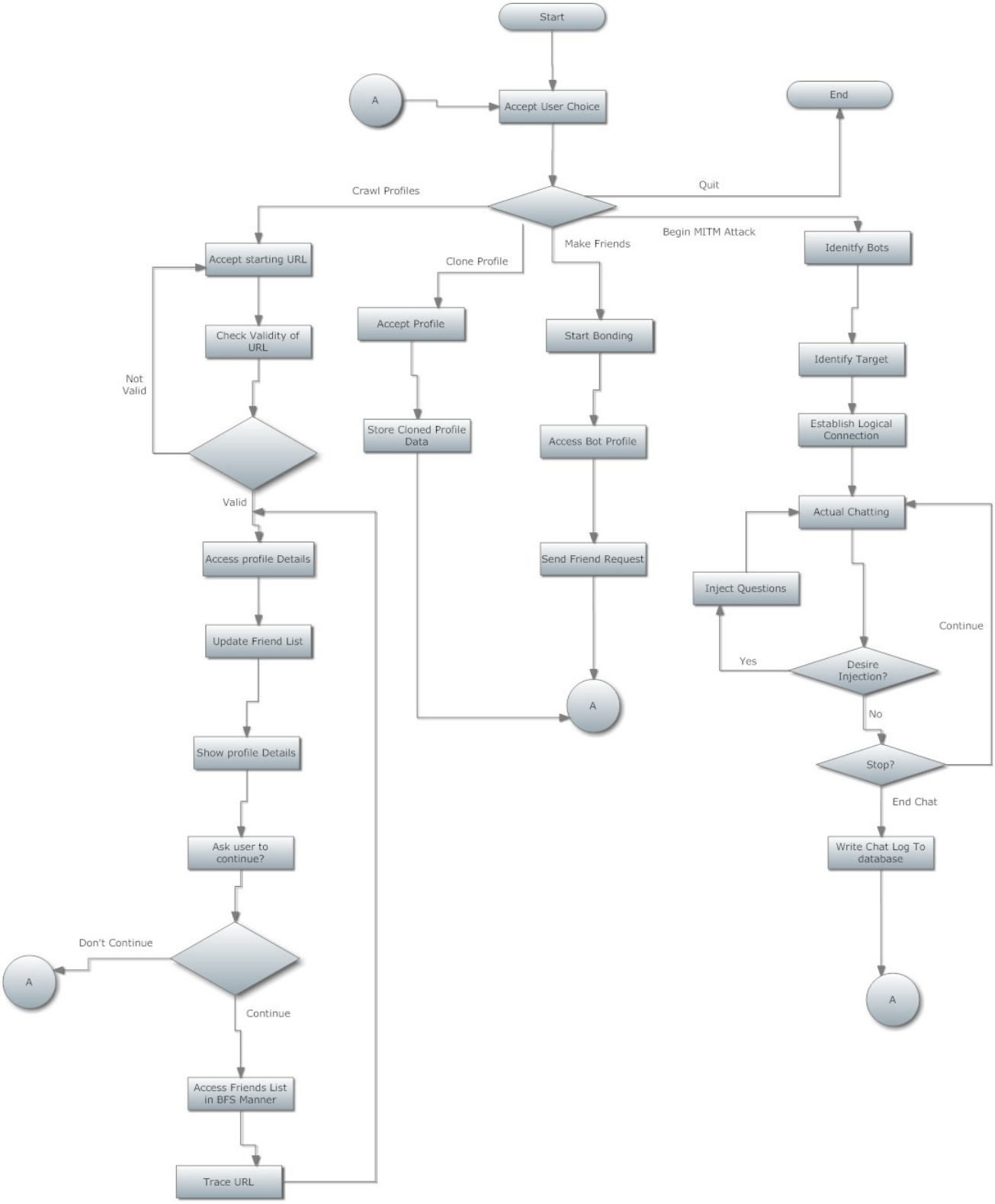

Figure 5.8: Activity Diagram

5.5 Hardware and software requirements

5.5.1 Hardware

- Standard Desktop Computer
- 2 GB Ram
- 80GB Hard Disk
- Internet Connection

5.5.2 Software

Operating System : Any Linux based distro

Language : Python 2.x

Regular Expression : Python's re Library

HTTP Requests : Python's urllib, urllib2 Library

XMPP : Python's SleekXMPP Library

Chapter 6

Implementation

6.1 Passive Information Gathering Module

6.1.1 Database Connections

Following snippet initializes the database connections :-

```
import MySQLdb

def initdbconfig():

    """
    Inputs  : Nothing
    Does    : Sets up the database
    Returns : Table_Profile_Name, Table_Friends_Name and Database Connection
    """

    #Configuration Values
    tbProfile = "User_profile"
    tbFriends = "User_friends"

    #Update User Password if its on different machine.
    conn = MySQLdb.connect(host='localhost', port=3306, user='root', passwd='
        password',charset='utf8')

    return (tbProfile, tbFriends, conn)
```

project/code/initdbconfig.py

This code creates the two Tables :-

```
import MySQLdb

def initializedb(conn, tbProfile, tbFriends):

    """
    Inputs  : Database Connection, Table_Profile_Name, Table_Friends_Name
    Does    : Creates Database, Tables, Sets UTF8 enabled.
    Returns : Nothing
    """

```

```
    c = conn.cursor()

    #Supress warning if table already exists
    c.execute ('SET sql_notes = 0')

    c.execute("create database if not exists fbdata")

    #Switch to fbdata database.
    c.execute("use fbdata")

    #check following to understand how to create AUTO_INCREMENT on a non_primary
        key coloumn
    #http://stackoverflow.com/questions/922308/is-it-possible-to-have-an-auto-
        increment-column-that-permits-duplicates


    c.execute('create table if not exists ' + tbProfile +  '(seqid BIGINT
        UNSIGNED NOT NULL AUTO_INCREMENT, pid BIGINT UNSIGNED primary key, name
        varchar(50) character set utf8 NOT NULL, usrname varchar(30), frnds
        integer, stamp timestamp, scraped integer, isdone integer,Level integer,
         Sex varchar(8) NOT NULL, KEY (seqid))')

    c.execute('CREATE TABLE if not exists  '+tbFriends+' (pid1 BIGINT, pid2
        BIGINT, INDEX fpid1 (pid1), FOREIGN KEY (pid1) REFERENCES '+tbProfile+'(
        pid), INDEX fpid2 (pid2), FOREIGN KEY (pid2) REFERENCES '+tbProfile+'(
        pid))')

    #Allows you to view unicode Character
    c.execute("set names utf8")

    #Enable warnings Again
    c.execute ('SET sql_notes = 1')


    conn.commit()
    c.close()
```

project/code/initializedb.py

6.1.2 Facebook Login

To implement scraping, we needed to login via the script. Following is the snippet

```
import re, urllib, urllib2, cookielib, getpass, time
from Decorators import openlink

def LoginFB():

    """
    Inputs  : Nothing.
    Does    : Logins to facebook given account.
    Returns : Browser Object, Access Token.
    """
    user = raw_input("Your Facebook Login ID: ")
```

```
12      passw = getpass.getpass()
13
14      # Initialize the needed modules
15      CHandler = urllib2.HTTPCookieProcessor(cookielib.CookieJar())
16      browser = urllib2.build_opener(CHandler)
17      browser.addheaders = [('User-agent', 'Mozilla/5.0 (X11; Linux i686 on x86_64
            ; rv:7.0.1) Gecko/20100101 Firefox/7.0.1')]
18      urllib2.install_opener(browser)
19
20      # Initialize the cookies and get the post_form_data
21      print 'Initializing..'
22
23      #uses decorator to keep trying ten times
24      (res) = openlink(browser, 'http://m.facebook.com/index.php')
25
26      mxt = re.search('name="post_form_id" value="(\w+)"', res.read())
27      pfi = mxt.group(1)
28      print 'Using PFI: %s' % pfi
29      res.close()
30
31      # Initialize the POST data
32      data = urllib.urlencode({
33          'lsd'                   : '',
34          'post_form_id'          : pfi,
35          'charset_test'          : urllib.unquote_plus('%E2%82%AC%2C%C2%B4%2C%E2%82%
                AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84'),
36          'email'                 : user,
37          'pass'                  : passw,
38          'login'                 : 'Log+In'
39      })
40
41      # Login to Facebook
42      print 'Logging in to account ' + user
43
44      #Trying to connect to FB and retrying if failes
45      connected = False
46      errval = 0
47      while not connected:
48          try:
49              errval +=1
50              res = browser.open('https://www.facebook.com/login.php?m=m&refsrc=
                    http%3A%2F%2Fm.facebook.com%2Findex.php&refid=8', data)
51              connected = True
52
53          except :
54              time.sleep(5)
55              print "Trying to Login Again to Facebook, count = " + str(errval)
56
57              if errval == 10:
58                  break;
59
60              pass
61
62
```

```
63      #If Log out value is not shown, something went wrong, check your
            localization of your account,
64      #maybe its in some other language.
65      rcode = res.code
66      if not re.search('Logout', res.read()):
67          print 'Login Failed'
68
69          # For Debugging (when failed login)
70          fh = open('debug.html', 'w')
71          fh.write(res.read())
72          fh.close
73
74          # Exit the execution :(
75          exit(2)
76      res.close()
77
78      # Get Access Token
79      #Keep retyring till you get it, using decorators here.
80      (res) = openlink(browser,'http://developers.facebook.com/docs/reference/api'
            )
81
82      #Extract the access token value
83      conft = res.read()
84      mat = re.search('access_token=(.*?)"', conft)
85      acct = mat.group(1)
86      print 'Using access token: %s' % acct
87
88      return (browser, acct)
```

project/code/LoginFB.py

6.1.3 Fetching Friends

Following Code fetches Friends list of a user.

```
1  import re, json
2
3  from Database import initdbconfig, updateProfile2db, storefriendsdb,
       setProfile2db, initializedb, checkdb
4  from LoginFB import LoginFB
5  from Decorators import openlink
6
7  def FetchFriends(friendscount, pid, browser, acct, conn, tbProfile, tbFriends,
       Level, orgcount, MainCount):
8
9      """
10     Inputs  : Friendscount incmremented by 10, profile ID, browser object,
           access token, Database Connection,
11                 Table_Profile_Name, Table_Friends_Name, Level, Original Friends
                       Count, Total People Fetched Count
12     Does    : Gets the new list of Friends of the Given pid.
13                 friendscount is used in for loop to generate the count incremented
                       by 10, as m.facebook domain
```

```
14              shows friends list 10 per page. If the actual friend count is lets
                    say 437 (orgcount), friendcount
15              value will be 447, which in range(0,447,10) will generate value
                    upto 440 hence we get all the 437 values
16              as an example.
17      Returns :
18      """
19
20      #TODO - Note this Function is quite long, break it for simplicity.
21
22      count = 0
23
24      for val in range(0,friendscount,10):
25
26          print "
                ------------------------------------------------------------------------------
                "
27
28          #Keep retyring till you get it, using decorators here.
29          url = 'http://m.facebook.com/friends/?id=%s&f=%s&refid=5&access_token=%s
                ' % (pid, val, acct)
30          (res) = openlink(browser, url)
31
32          htmlfile = res.readlines()
33
34          debugfile = open("debug.html", "w")
35
36          #To debug the current Friends list HTML Being fetched.
37          for line in htmlfile:
38              debugfile.write(line)
39
40          debugfile.close()
41
42
43          for line in htmlfile:
44
45              #Search for the Line which says Add Friend in the HTML Page just
                    fetched.
46              m = re.search("Add Friend", line)
47
48              if m:
49
50                  #Find all the Friends Name in this HTML page which is on the Add
                        Friend Line.
51                  nameit = re.compile(u'name="[^>_=]+"><span>([^>_=]+)</span>')
52                  nameList = nameit.findall(line)
53
54                  #Reverse the nameList as we will pop out one by one values
55                  nameList.reverse()
56
57                  #Find all the either Profile IDS OR Usernames from the Regex
                        below.
58                  matchit = re.compile(r'<div class="c"><a href=".profile.php\?id
                        =([\d]+)|<div class="c"><a href=".([^?]+)')
```

```
59                  idList = matchit.findall(line)
60
61                  for (profileID,username) in idList:
62
63                      print "
                            ----------------------------------------------------------------------
                            "
64
65                      #Store the 1st Name in name variable
66                      name = nameList.pop()
67
68
69                      if profileID:
70                          print "Value of profileID = " + profileID
71
72                          #Using the profile id just found, get his name, gender,
                                username.
73                          # VIA FB API and Access Token.
74                          frndpid = str(profileID)
75                          (frndpid, name, gender, username) = getUserid(browser,
                                frndpid, acct)
76
77                          print "Name = " + name + "\t\t and "  + "Userid = " +
                                str(frndpid)
78
79                      else:
80                          print "Value of username = " + username
81
82                          #Using the username get the profile ID, name, gender,
                                VIA FB API and Access Token.
83                          (frndpid, name, gender, username) = getUserid(browser,
                                username, acct)
84
85                          print "Name = " + name + "\t\t and "  + "Userid = " +
                                str(frndpid)
86
87                      #Increment the count of friends done for this users friends
                            list.
88                      count += 1
89
90                      #Get the friends number of friends using Regex.
91                      (actualcount) = getNumberOfFriends(browser, frndpid)
92
93                      #TODO : Note For some profiles, FB API, returns False.
                            Either leave it, or FIX it by
94                      #       Swicthing to REGEX Completly. Example Username =
                            vnittoor
95                      #If FB API Did not return a False (= 0) on the friends
                            profile ID, store it, else skip it.
96                      if frndpid != 0:
97                          storefriendsdb(conn, str(pid), str(frndpid), tbFriends)
98                          setProfile2db(conn, frndpid, name, actualcount,
                                tbProfile, Level, gender, username)
99
```

```
100                     else:
101                         print "PID = 0, Not inserting it in the database"
102
103                     print "----------Count = %s
                            ----------" % count
104                     MainCount += 1
105                     print "Total People Done = " + str(MainCount)
106
107
108                     #Reissue FB Access Token Every 500 Requests.
109                     if MainCount % 500 == 0:
110                         (browser, acct) = LoginFB()
111                         print ">" * 200
112                         print "-----ReLogging In to Facebook and getting New
                                Access Token-----"
113                         print "<" * 200
114
115
116                 print "Count of Friends = " + str(count)
117
118
119     print "Actual Friends = " + str(orgcount) + " And Scrapped Friends = " + str
        (count)
120     print "Done All Friends"
121
122     #If Friends Fetched Equals Actual Count, set isdone = 1, else -1
123     if orgcount == count:
124         updateProfile2db(conn, str(pid), tbProfile, count, 1)
125     else:
126         updateProfile2db(conn, str(pid), tbProfile, count, -1)
127
128     return (MainCount,browser, acct)
```

project/code/FetchFriends.py

And Following is a sample of data collected of a Facebook User.

Attributes	User1	UserN
Seq Id	**1**	**3**
Profile ID	**100002012773778**	**698854997**
Full Name	**Hemant Rathore**	**Ashish Chauhan**
Usrname	**NA**	**chauhan.ashish**
Frnds	**35**	**1356**
TimeStamp	**2011-12-26 01:42:02**	**2011-12-26 03:28:44**
Scrapped	**35**	**1280**
IsDone	**1**	**-1**
Level	**0**	**1**
Sex	**male**	**male**

6.1.4 Summary of Data

Target Profile ID : 100002012773778

Username : NA

Name : Hemant Rathore

Gender : Male

Profile Link : http://www.facebook.com/profile.php?id=100002012773778

Total Friends of Hemant Rathore (Level 1): 35

Total Friends of Friends of Hemant Rathore (Level 2): 7443

Total Friends of Friends of Friends of Hemant Rathore (Level 3): 17,73,288

Number of Males at Level 1: 30

Number of Females at Level 1: 3

Number of NA at Level 1: 1

Number of Males at Level 2: 4420

Number of Females at Level 2: 1241

Number of NA at Level 2: 97

Number of People who have hidden their Friends list at Level 1: 11/35

Number of People who have hidden their Friends list at Level 2: 1733/7443

Total Number of People who have hidden their Friends list: 1744/7478 Approx 25% Only.

Total Number of People who Do NOT have usernames set: 3128/7478 = 40% Approx

Total Number of People who Have Hidden their Gender: 98/7478 = 1% Approx

Run Duration: 473 Mins.

Friends Fetched/Min: (7443+35) / 473 = 15 Frnds/mins (rounded off)

Data Base Size: 0.94989872 MB

Number of Unique friends edges found: 6613

Number of Unique Profiles found: 5793

6.2 Facebook Chat Attack

Following is a snippet of Chat Replay and Modification Code

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import logging
import getpass
from optparse import OptionParser
import re

import sleekxmpp

if sys.version_info < (3, 0):
    reload(sys)
    sys.setdefaultencoding('utf8')
else:
    raw_input = input


class EchoBot(sleekxmpp.ClientXMPP):

    def __init__(self, jid, password):
        sleekxmpp.ClientXMPP.__init__(self, jid, password)

        # The session_start event will be triggered when
        # the bot establishes its connection with the server
        # and the XML streams are ready for use. We want to
        # listen for this event so that we we can initialize
        # our roster.
        self.add_event_handler("session_start", self.start)

        # The message event is triggered whenever a message
        # stanza is received.
        self.add_event_handler("message", self.message)

        self.flag = 0
```

```
36
37      def start(self, event):
38          """
39          Process the session_start event.
40
41          Typical actions for the session_start event are
42          requesting the roster and broadcasting an initial
43          presence stanza.
44
45          Arguments:
46              event -- An empty dictionary. The session_start
47                       event does not provide any additional
48                       data.
49          """
50          self.send_presence()
51          self.get_roster()
52
53      def message(self, msg):
54          """
55          Process incoming message stanzas. Be aware that this also
56          includes MUC messages and error messages.
57          """
58
59
60          #This is the modification attack code.
61          #Check if the incoming message is the initialization string
62          #if yes, set the flag and append the attack vector
63          if msg['body'].lower() == 'how was the movie yesterday?'.lower() and
                self.flag == 0:
64              msg['body'] = msg['body'] + ' and btw, whats your fav color?'
65              self.flag = 1
66
67          elif self.flag == 1:
68
69              #check if response had the word color.
70              colors = ['Pink', 'Red', 'Orange', 'Brown', 'Yellow', 'Gray', 'Green
                    ', 'Cyan', 'Blue', 'Violet']
71              for x in colors:
72                  if msg['body'].__contains__(x.lower()):
73                      #if yes, reset the flag and modify the outgoing message as
                            well
74                      msg['body'] = "you know my fav color is " + x + " whats
                            yours?"
75                      self.flag = 0
76
77
78          if msg['type'] in ('chat', 'normal'):
79
80              #if message from simran's id
81              if msg['from'] == '-100003188198954@chat.facebook.com':
82
83                  #alex bot) will send the message to simranbot id
84                  sendto = '-100003024364455@chat.facebook.com'
85
```

```
86              #if message is from simran bot id
87              elif msg['from'] == '-100003024364455@chat.facebook.com':
88
89                  #alex bot will send the message to simran's id
90                  sendto = '-100003188198954@chat.facebook.com'
91
92              self.send_message(mto=sendto, mbody=msg['body'], mtype='chat')
93
94  if __name__ == '__main__':
95      # Setup the command line arguments.
96      optp = OptionParser()
97
98      # Output verbosity options.
99      optp.add_option('-q', '--quiet', help='set logging to ERROR',
100                     action='store_const', dest='loglevel',
101                     const=logging.ERROR, default=logging.INFO)
102     optp.add_option('-d', '--debug', help='set logging to DEBUG',
103                     action='store_const', dest='loglevel',
104                     const=logging.DEBUG, default=logging.INFO)
105     optp.add_option('-v', '--verbose', help='set logging to COMM',
106                     action='store_const', dest='loglevel',
107                     const=5, default=logging.INFO)
108
109     # JID and password options.
110     optp.add_option("-j", "--jid", dest="jid",
111                     help="JID to use")
112     optp.add_option("-p", "--password", dest="password",
113                     help="password to use")
114
115     opts, args = optp.parse_args()
116
117     # Setup logging.
118     logging.basicConfig(level=opts.loglevel,
119                         format='%(levelname)-8s %(message)s')
120
121     if opts.jid is None:
122         opts.jid = raw_input("Username: ")
123     if opts.password is None:
124         opts.password = getpass.getpass("Password: ")
125
126     # Setup the EchoBot and register plugins.
127     xmpp = EchoBot(opts.jid, opts.password)
128     xmpp.register_plugin('xep_0030') # Service Discovery
129     xmpp.register_plugin('xep_0004') # Data Forms
130     xmpp.register_plugin('xep_0060') # PubSub
131     xmpp.register_plugin('xep_0199') # XMPP Ping
132
133     # Connect to the XMPP server and start processing XMPP stanzas.
134     if xmpp.connect():
135
136         xmpp.process(block=True)
137         print("Done")
138     else:
139         print("Unable to connect.")
```

project/code/alexbot.py

6.2.1 Modified Turing Test

We conducted a POC Test with real users and here are the screenshots

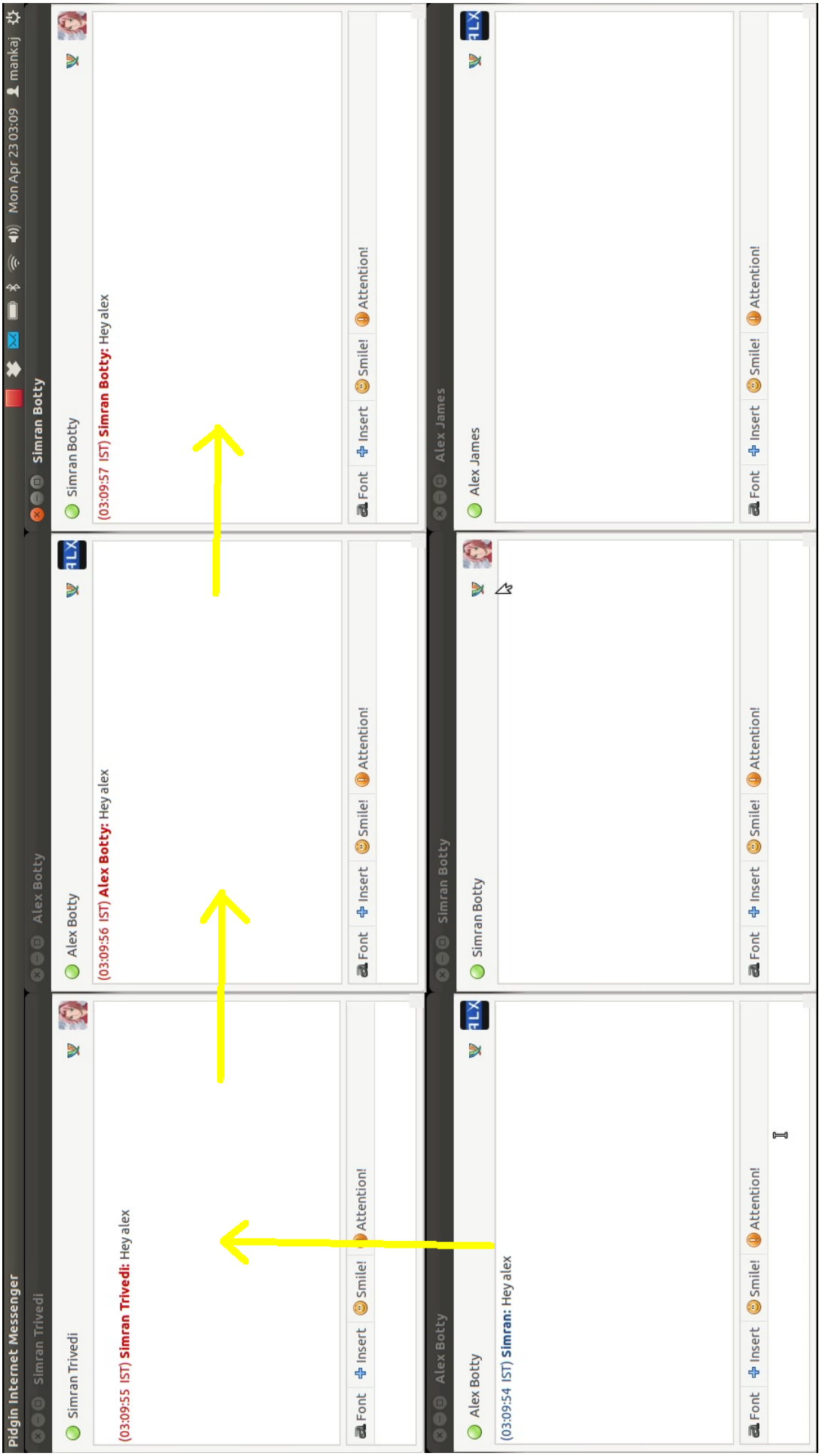

Figure 6.1: Facebook Chat Attack - 1

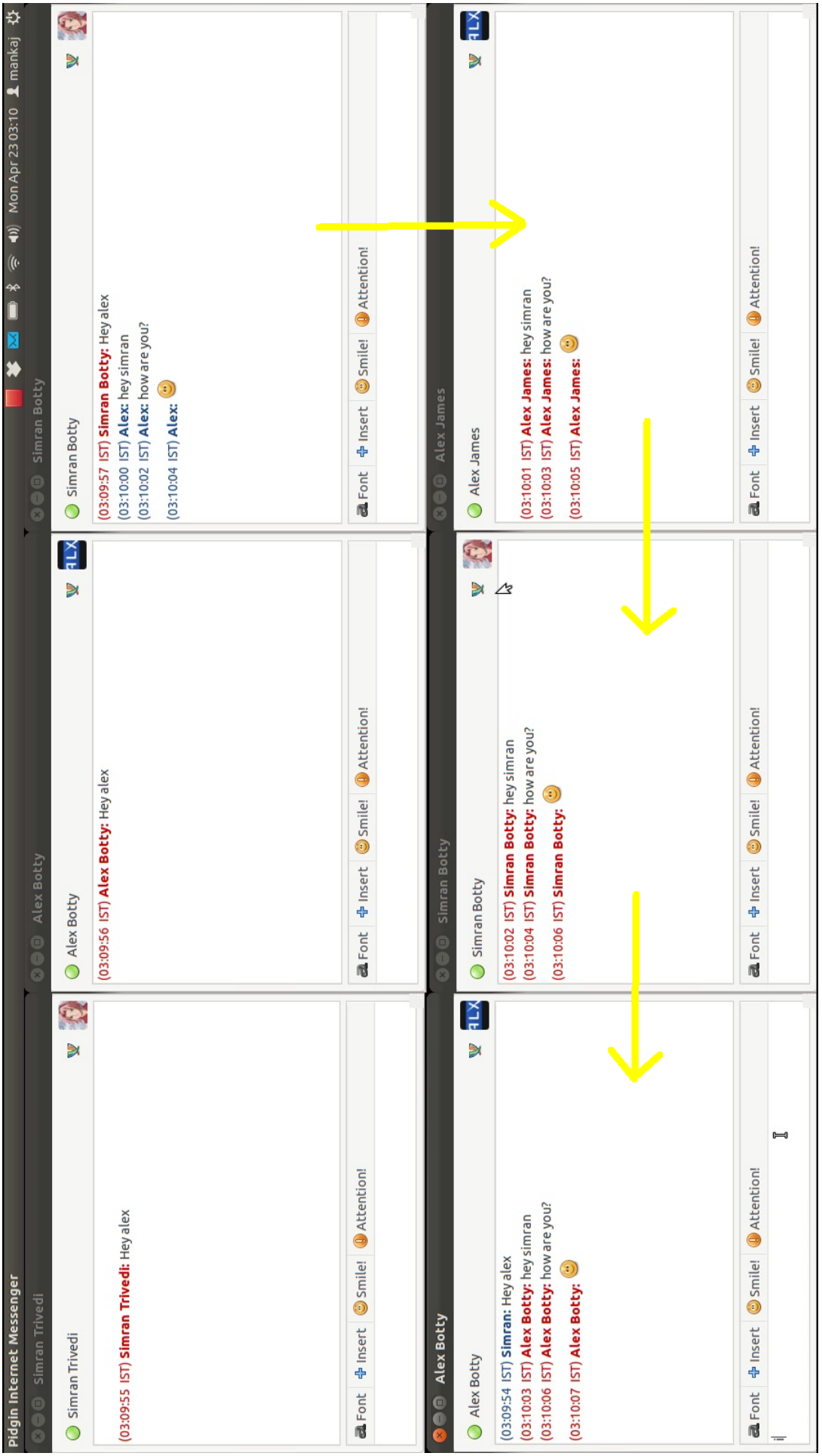

Figure 6.2: Facebook Chat Attack - 2

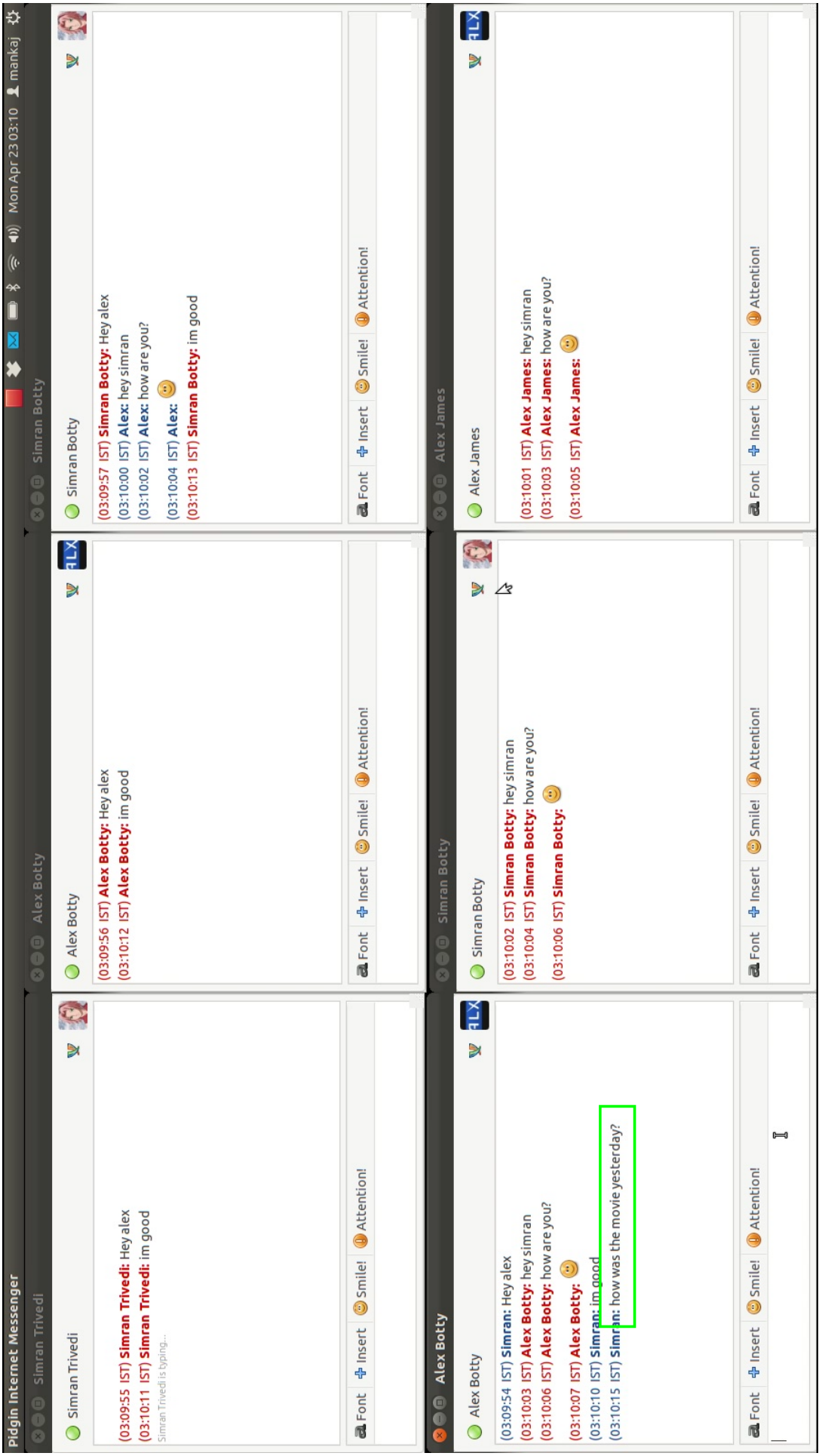

Figure 6.3: Facebook Chat Attack - 3

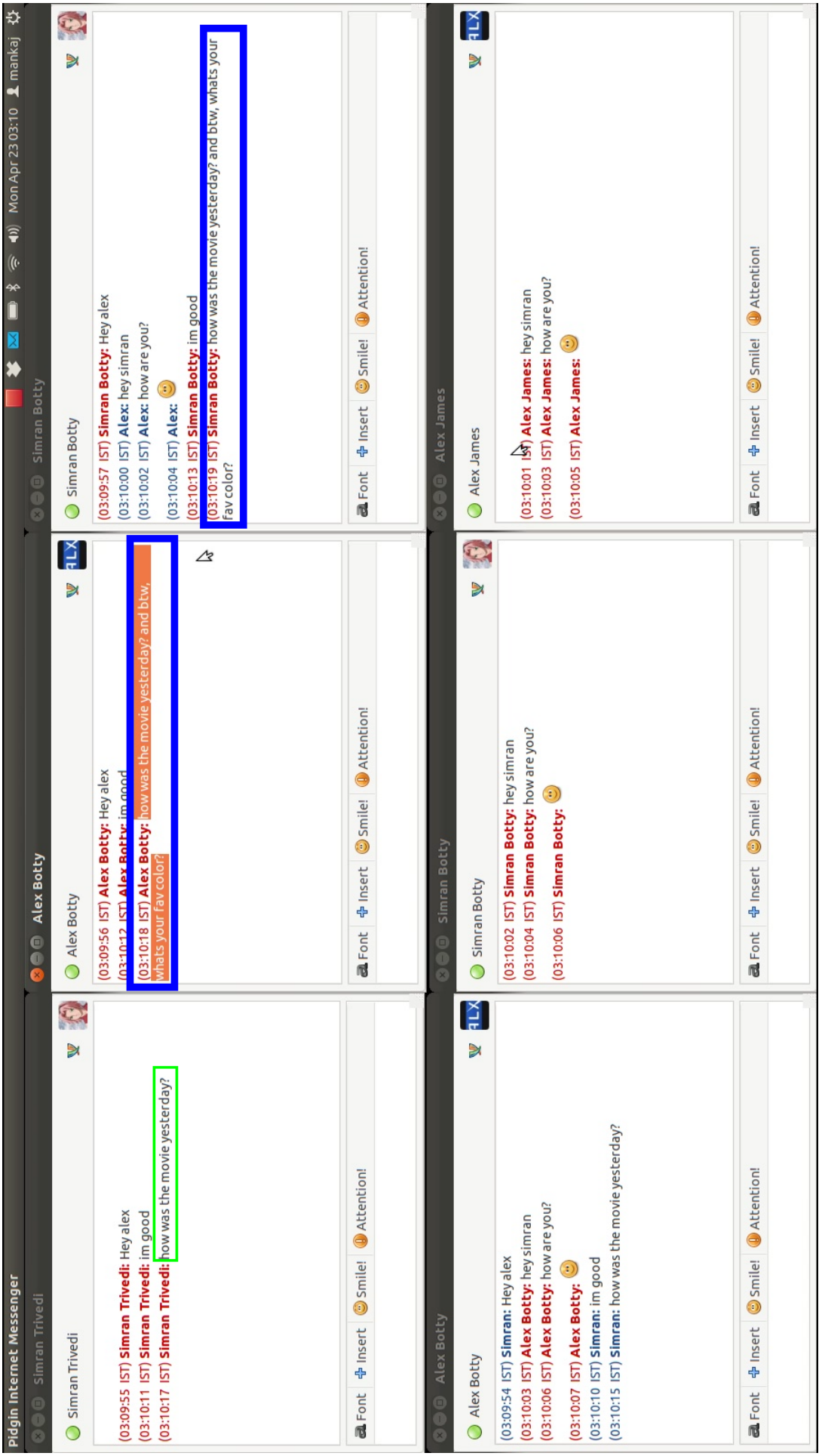

Figure 6.4: Facebook Chat Attack - 4

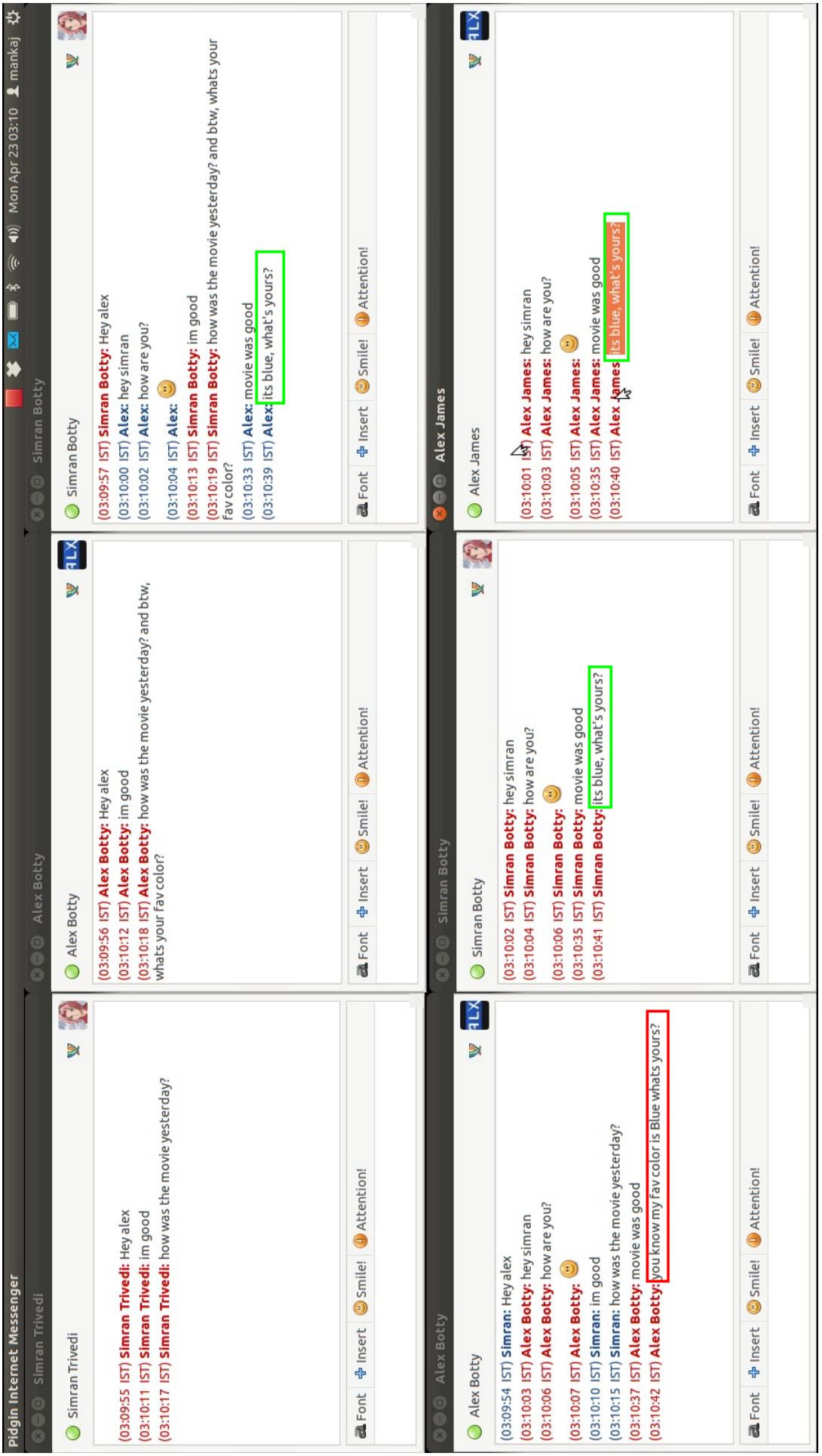

Figure 6.5: Facebook Chat Attack - 5

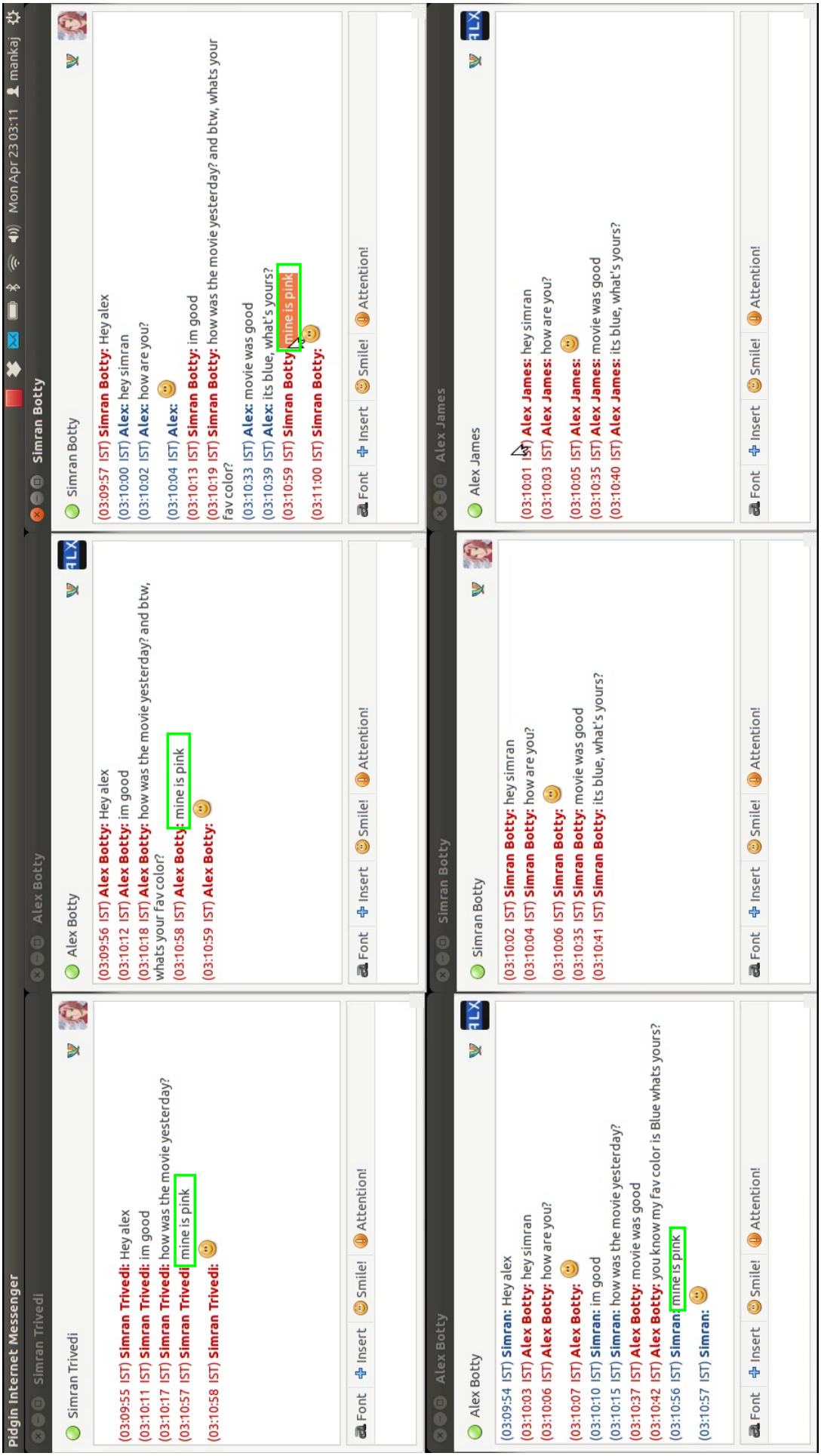

Figure 6.6: Facebook Chat Attack - 6

Chapter 7

Scheduling

7.1 Proposed Modules

- Passive Information Gathering
- Facebook Chat Attack
- Modification of chat on the fly

7.2 Scheduling

August-September 2011 : Finalized the Problem Statement

September-October 2011 : Prepared SRS

November-December 2011 : Exam Break

December-January 2011-12 : Passive Information Gathering Module

February 2012 : Learned XMPP, tried XMMPPPY Library and failed

March 2012 : Learned SleekXMPP Python Library.

March-April 2012 : Implemented Facebook Chat Attack

April 2012 : Finished Project Report

Chapter 8

Conclusion and Future Scope

8.1 Future Scope

The project certainly has a lot of scope. Let's take a look at some of them

- We could extend the capabilities of extracting information to gather personal information as well like About Me page of the user.
- Parallizes the HTTP request to make it faster
- Deploy the passive information extractor on Amazon EC2 Servers.
- Hook up AI bots like megahal[17] in Facebook Chat Attack and train our scripts to learn when/what to modify
- Feature to clone profiles
- Scale the architecture where Facebook Chat Attack would work on more then just two people at the same time

8.2 Conclusion

The project was sucessfully implemented and a Proof of Concept was established that Social Engineering can be done on Social Networks.
We were able to extract out public information from Facebook user's profile on a mass scale. Also we demonstrated that Trust between two users can be used to extract out information which is not available publicaly.

References

[1] *Honeybot, Your Man in the Middle for Automated Social Engineering*;Tobias Lauinger, Veikko Pankakoski, Davide Balzarotti, Engin Kirda; EURECOM Sophia-Antipolis, France. LEET10 Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats.

[2] *All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks*; Leyla Bilge, Thorsten Strufe, Davide Balzarotti, Engin Kirda EURECOM Sophia Antipolis, France. WWW 09 Proceedings of the 18th international conference on World Wide Web.

[3] *Eight Friends Are Enough Social Graph Approximation via Public Listings*; Joseph Bonneau, Jonathan Anderson, Frank Stajano, Ross Anderson. SNS 09: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems.

[4] *Towards Automating Social Engineering Using Social Networking Sites*;Huber, M.; Kowalski, S.; Nohlberg, M.; Tjoa, S.; Computational Science and Engineering, 2009. CSE 09. International Conference.

[5] Social Engineering : The art of Human Hacking

[6] Link Name
`http://goliath.ecnext.com/coms2/gi_0199-7186209/The-human-element-the-weakest`

[7] Market Share of Social Network Sites
`http://techcrunch.com/2011/12/22/googlesplus`

[8] Web Scraping
`http://en.wikipedia.org/wiki/Web_scraping`

[9] Extensible Messaging and Presence Protocol
`http://en.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol`

[10] Urllib2 Python Library
http://docs.python.org/library/urllib2.html

[11] Regular Expression
http://www.zytrax.com/tech/web/regex.html

[12] XMPPPY Python Library
http://xmpppy.sourceforge.net/

[13] Issues using XMPPPY Link 1
http://stackoverflow.com/questions/4732230/
xmpppy-and-facebook-chat-integration

[14] Issues using XMPPPY Link 2
http://stackoverflow.com/questions/8841367/
using-xmpp-for-facebook-chat-in-python

[15] Issue Resolved using XMPPY
http://superuser.com/questions/387504/
unable-to-connect-to-facebook-chat-via-python-using-xmpppy-library

[16] SleekXMPP Python Library
http://sleekxmpp.com/

[17] MegaHAL
http://en.wikipedia.org/wiki/MegaHAL