Experiment No. 8

Aim: Write a program in C++ /Python to analyze email header.

TITLE	Email Header Analysis
PROBLEM STATEMENT / DEFINITION	Write a program in C++ / Python to analyze email header.
OBJECTIVE	• To understand the parameters of email header • To analyze the life of an email • Able to analyze email header • To apply client – server programming model
S/W PACKAGES AND HARDWARE APPARATUS USED	1. Operating System : Latest version of 64 –Bit operating systems open source Fedora 20 or higher equivalent or Windows 8 with Multicore CPU equivalent to Intel i5 / 7 generation onwards 2. Debain based OS (Ubuntu 13.04) , g ++ , Python 2.7 3. Programming Tools : Latest 64 –Bit Programming languages such as Microsoft Visual Studio (ver. 12 or Higher) or equivalent open source , Eclipse 64 –Bit platform / IDLE editor 4. Networked computer with internet access 5. Valid e–mail account 6. 32 / 64 –bit LATEX
REFERENCES	• 'Internetworking with TCP/IP Principles, Protocols & Architectures '; Pearson Education, Vol. 1, Douglas E. Comer • 'An Introduction to Computer Networking '; Kenneth C. Mansfield & James L. Antonakos , Pearson Education • RFC 822 & 2822 • http://www.python.org/download (suitable installation for Windows 8 along with IDLE editor) • M. Tariq Banday , " Analysing E-Mail Headers for Forensic Investigation " , Journal of Digital Forensics, Security and Law, Vol. 6(2)
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning outcome 6. Related Mathematics 7. State transition diagram 8. Concepts related Theory 9. Program code with proper documentation. 10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes)

- **Aim**
  Write a program in C++ / Python to analyze email header.

- **Prerequisites**
  - Object oriented programming features.
  - Eclipse framework
  - File handling

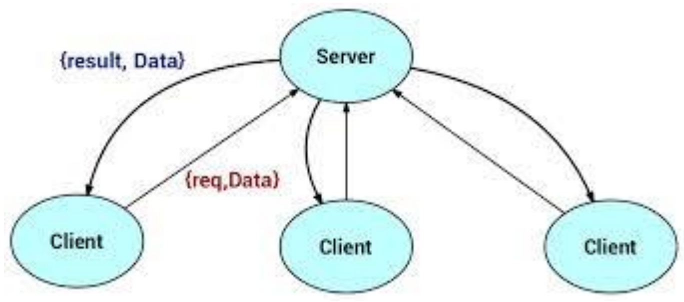
- **Learning Objectives**
  - Ability to identify parameters of email header
  - To analyze the lifetime of an email.
  - Able to analyze email header fields.

- **Learning Outcome**
  - After successfully completing this assignment, you should be able to
  - Demonstrate email header analysis based on various header fields and its values.

- **Related Mathematics**

Mathematical Model
Let us consider **S** as a solution perspective of the email header analysis system

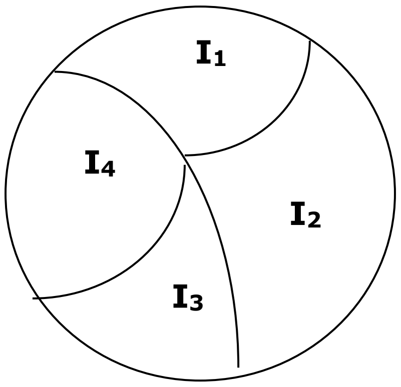
$S = \{ St , E , I , O , F_{me} , DD , NDD , S_u , F \mid \varnothing \}$
where ,
St = initial state i.e. connection establishment
St = { client , server , ip , port }
    where ip = netid + hosted = 32 bits
        port = 2^{16} combination $^{n}C_{r}$



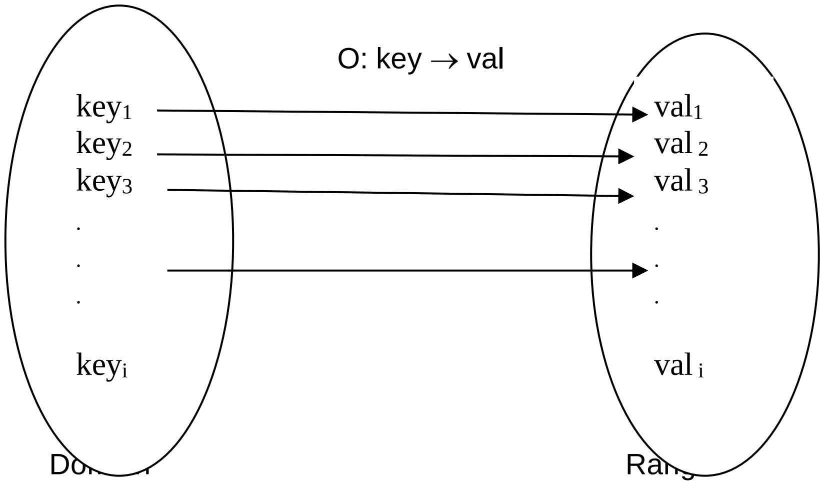
E = end state = close connection

I = unstructured header ($[a - z]^{*} \mid [A - Z]^{*} \mid [0 - 1]^{*} \mid$ @...)
I = Full email header – body i.e. capture email header up to < content type >
$I_t = \{ x_{junk} , x_{noreply} , x_{forw} , x_{cc} \}$ = types of email headers captured

$I = \{ I_4 , I_3 , I_2 , I_1 \mid I_4 \cap I_3 \cap I_2 \cap I_1 = \varnothing \}$i.e. I divided into chunks
    where ,
        I_1 , I_2 , I_3 , I_4 are chunks of email header in bottom – up approach
$\Pi = \{ \{ I_4 \} , \{ I_3 \} , \{ I_2 \} , \{ I_1 \} \}$



O = analyzed e–mail header i.e. $\{ (key_i , val_i) \mid 0 < i < 13 \}$
where i be number of fields of header
$| key_i | = | val_i |$
$key_i \ R \ val_i$; where R is the symmetric relation with **surjective mapping**



Let f_i be various fields of headers where $0 < i \leq 13$ (important for investigation purpose)
f_i = { ReturnPath , Received$^+$, To$^+$, From , MessageID , num_hops , Date , MIME – version , Subject , Content – Type$^+$, X – Apparently – To , X – Originating – IP }
where + indicated 1 or more occurrence

NDD: { Received$^+$, X – Apparently – To , To$^+$, num_hops } (depends on I_t & routing technique)

DD: $f_i \oplus$ NDD = ($f_i \cup$ NDD) – ($f_i \cap$ NDD) i.e. symmetric difference

F_{me} : { parse w.r.t. grammars from RFC 822 } – CRLF (carriage return & line feed) & WSP

F_{chunks} – I_4 , I_3 , I_2 , $I_1 \subseteq I$
F_{alt} – Rule for local alternatives
F_* – Rule for repetition
$F_{[]}$ – Rule for optional elements
$F_{\#}$ – Rule for interval boundaries
$F_;$ – Rule for comment

F_{uf} – unfolding – make use of CRLF
$F_{compare}$

Input Analysis:
Input ‘I’ must be e – mail header file & not other.

Algorithm:

Step 1: Start – Establish the connection between client & server for further communication

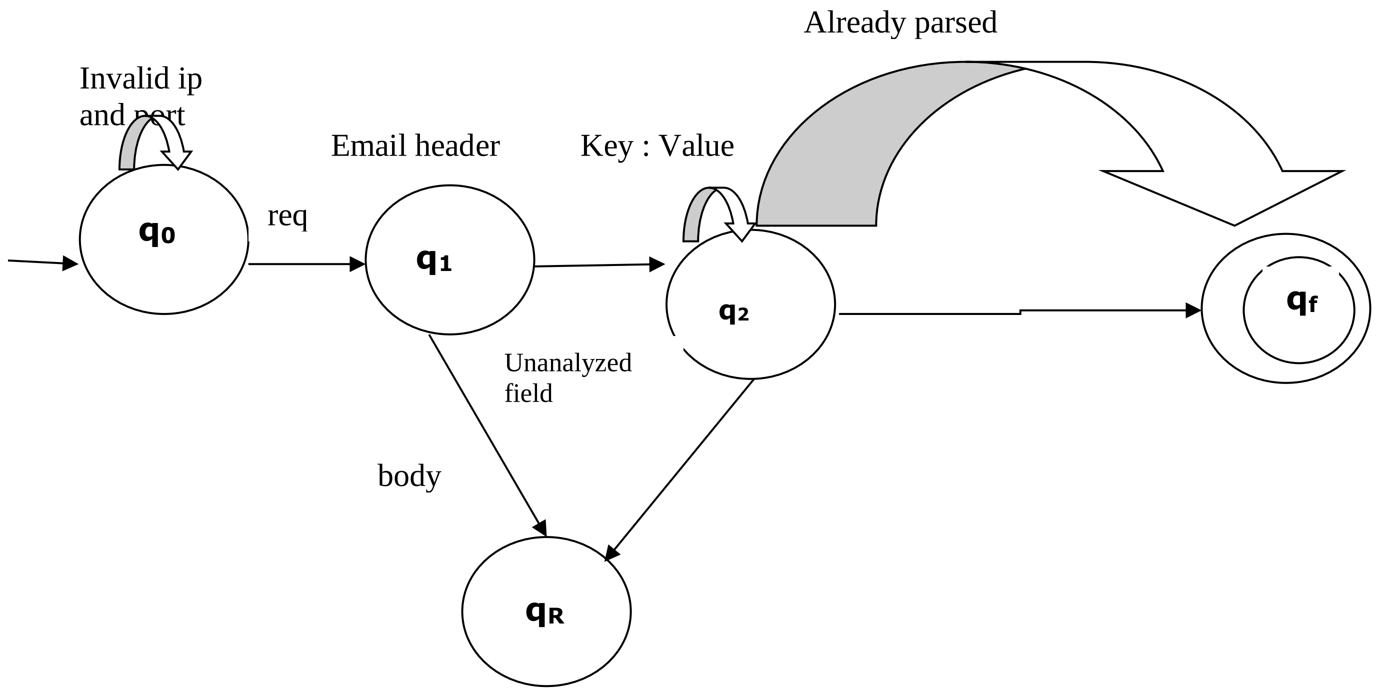
Step 2: Input – Find or capture few samples of full email headers (up to content type)

- Using API / library used by any email system & openly available
- Using packet capturing tool
- Using manual email header capturing of mail system & storing on central repository in txt format (file handling)

Step 3: Analysis with reference to RFC 822 / 2822 (bottom – up approach)

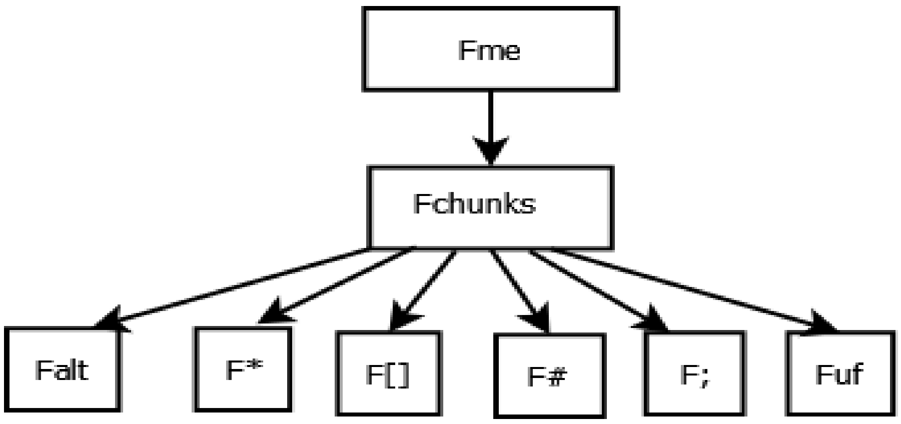
Step 4: Connection Tear–off

State Transition Diagram:



where,
q_0 is init state (connection establishment)
q_1 is handle request / response
q_2 is parser of email header analyzer
q_4 is connection close (end state)
q_R rejection state for invalid

Function Dependency Diagram:



S_u :

F: Parsing on file other than email header (invalid input) i.e. I = ¬ I

- **Concepts and Related Theory:**

  Introduction:

Understanding how e–mail system work is extremely important if one actually wants to be able to solve e–mail related threats. All e–mail communication on Internet is governed by rules & regulations laid down by SMTP (port 25) & POP (port 110). HTTP is used to transfer displayable web pages & related files whereas SMTP is used to transfer E–mail. POP3 specifies how user can retrieve the contents of mailbox. MIME adds lines to the header to define the type of the data & the encoding used. IMAP4 provides extended functionality for message retrieval & processing. A user can obtain information about a message or examine header fields without retrieving the entire message. While tracing E–Mails, "header information" is included along with E–mails either at the beginning or end of E–Mail messages.

Journey of an E–mail:

Sender Outbox → Source Mail Server → Interim Mail Servers → Destination Mail Server → Destination Inbox.

E–mail Header:

An e–mail consists of two parts; header & body. In an e-mail, the body (content text) is always preceded by header lines. *Message header* provides the entire path of E–mail's journey from its origin to destination. Most cyber–crime investigators turn to electronic mail headers for evidence in any kind of e–mail related crime. E–mail headers are automatically generated & embedded into an e–mail message both during composition & transfer between systems. Header information varies with E–mail service provider, E–mail applications & system configuration. Header part carries information i.e. needed for E–mail routing, subject line & time stamps. The mail message format 822 uses blank line to separate message header & body.

The header contains several mandatory & optional fields, trace information & heading fields. E–mail header is a sequence of fields (not in a particular order).

Forensic Analysis of E–Mail (Bottom – up approach to trace E–Mail source):

The E–Mail was handed from the machines (MTA) at the bottom of the E–Mail header to one at the top of it. The control information i.e. envelope & headers including headers in the message body that contain information about the sender & / or the path along which the message has traversed represents the metadata of an e–mail message. The analysis of this metadata called header analysis can be used to determine genuineness of a message.

Divide the header information into separate chunks, examine each chunk as an independent entity& then put back the entire individual puzzle piece together. Grammar for analysis distinguishes between:

- Header fields from the message body,
- Beginnings of fields from lines which continue fields,
- Field–names from field–contents.

Capture Email Header:

Open source SHIVA (Spam Honeypot with Intelligent Virtual Analyzer) provides capability of collecting & analyzing all spam thrown at it. It has two parts :

1. Receiver
2. Analyzer

Receiver acts as open relay SMTP server, collects all spam thrown at it & dumps them into a local directory. SHIVA can be used as SMTP server that will dump all the emails that it receives , into local directory . If only receiver part is started SHIVA acts as an SMTP server & all mails are dumped into queue directory.

Following packages to be installed on system before user starts installation procedure.

- Debian based OS (Preferably Ubuntu 13.04)
- Python 2.7
- exim4-daemon-light
- g++
- python-virtualenv

- python-dev
- libmysqlclient-dev
- mysql-client (optional, only if user wants to save spam data in database)

Case Study for Email header actual analysis:

I. The e–mail header of the forged e – mail created :

```
Return – Path: <billgates@microsoft.com>
Received: from pobox4.Standford.EDU ([unix socket]) bypobox4.Standard.EDU (Cyrus v2.1.16)
with LMTP; Wed, 24 Nov 2004 05:58:42 –0800
X – Sieve: CMU Sieve 2.2
Received: from smtp-roam.Standford.EDU (smtp–roam.Standford.EDU [171.64.10.152])
by pobox4.Standford.EDU (8.12.11/8.12.11) with ESMTP id iAODwgWI020583
for <afadia@pobox4.standford.edu>; Wed, 24 Nov 2004 05:58:42 –0800 (PST)
Received: from ankit.com ([203.94.218.178])
by smtp–roam.Standford.EDU (8.12.11/8.12.11) with SMTP id iAODvchL015286 for
afadia@standford.edu; Wed 24 Nov 2004 05:58:15 –0800
Date: Wed, 24 Nov 2004 05:58:15 –0800
Message–ID: <200411241358.iAODvchL015286@smtp–roam.Standford.EDU>
From: bill.gates@standford.edu (Unverified)
To: afadia@standford.edu
Subject: Hi
Hi
```

Analysis:

At a first glance, above e–mail looks quite authentic & nothing suspicious. However, a closer inspection reveals a number of gaping tell – tale signs that should arouse the suspicion of the victim:

- Since domain names of the two commands did not match. Sendmail actually entered the sender's e – mail address as billgates@microsoft.com
- In the FROM line, the sender's e – mail address is followed with the keyword " Unverified " in brackets
- On the other hand, in the first line of the e – mail headers, the sender's e – mail address has been displayed as billgates@microsoft.com, which does not match with the 2nd point mentioned above
- The last RECEIVED line tells us that the e – mail has been received from the domain ankit.com (which does not match with the domain of the sender's e – mail address)

II. The e–mail header of the forged e – mail created i.e. Investigating E – mail Crimes

A sample header set of an e – mail message sent by tariq@tariq.com pretending to be alice@alice.com and sent to bob@bob.com is shown below. In this e – mail, the sender's address, date e – mail was sent, reply – to address, and various other fields have been spoofed.

X – Apparently – To: *bob@bob.com via a4.b4.c4.d4; Tue, 30 Nov 2010 07:36:34 -0800*
Return-Path: *< alice@alice.com >*
Received-SPF: *none (mta1294.mail.mud.bob.com: domain of alice@alice.com does not designate permitted sender hosts)*
X – Spam – Ratio: *3.2*
X – Originating – IP: *[a2.b2.c2.d2]*
X – Sieve: *CMU Sieve 2.3*
X – Spam – Charsets: *Plain='utf-8' html='utf-8'*
X – Resolved-To: *bob@bob.com*
X – Delivered – To: *bob@bob.com*
X – Mail – From: *alice@alice.com*
Authentication Results: *mta1294.mail.mud.bob.com from=alice.com; domainkeys=neutral (no sig); from=alice.com; dkim=neutral (no sig)*
Received: *from 127.0.0.1 (EHLO mailbox-us-s-7b.tariq.com) (a2.b2.c2.d2) by mta1294.mail.mud.bob.com with SMTP; Tue, 30 Nov 2010 07:36:34 -0800*
Received: *from MTBLAPTOP (unknown [a1.b1.c1.d1]) (Authenticated sender: tariq@tariq.com) by mailbox-us-s-7b.tariq.com (Postfix) with ESMTPA id 8F0AE139002E for <bob@bob.com>; Tue, 30 Nov 2010 15:36:23 +0000 (GMT)*
From: *"Allice" Alice@a.com*
Subject: *A Sample Mail Message*
To: *"Bob Jones" <bob@bob.com>*
Content – Type: *multipart / alternative ; charset = " utf – 8 " ; boundary = " KnRl8MgwQQWMSCW6Q5 =_HgI2hw*
Adah5NLY"
MIME-Version: *1.0*
Content – Transfer – Encoding: *8bit*
Content – Length: *51*
Reply – To: *"Smith" <smith@smith.com>*
Organization: *Alices Organization*
Date: *Tue, 28 Nov 2010 21:06:22 +0530*
Return – Receipt – To: *smith@smith.com*
Disposition Notification – To: *jones@jones.com*
Message – Id: *<20101130153623.8F0AE139002E@mailbox - us-s-7b.tariq.com>*

- **Test Methods**
    - Black box Testing
    - White box testing
    - Manual Testing & / or Unit testing

Sr. No.	Test Case	Actual output	Expected output
1.	Invalid input (File not containing email header format)		Should not parse
2.	Header field containing list of		

	senders (valid input)		

Review Questionnaire:

1) What are email headers?
2) What information is present in an Email Header that can be used to track a sent mail?

Extra Assignment:
1. Write a program in Python of online email header capturing & Analyze various header fields
2. Extend the same program in which client – server model is reflected by request – response of e–mail header fields analysis

Conclusion:
Thus, after successfully completing this assignment, you should be able to understand & implement header analysis C++ / Python.

Experiment No. 9

Aim: Implement a program to generate and verify CAPTCHA image

CAPTCHA Explained

Your task in this assignment to write a C program which automatically recognizes the digits in a CAPTCHA image.

A CAPTCHA is an attempt to determine whether or not a user is human. It is in effect a reverse Turing test. CAPTCHA are designed to be difficult to recognize with a computer - the design and assessment of this assignment recognizes this difficulty.

The input to your program will be a black-and-white (monochrome) image in a simple format described below. Each image will contain either 1 or 4 digits. The output of your program should be the digits in the images

Image Format

Common image formats such as JPEG and PNG are complex, and decoding them would too difficult a task for this assignment.

Instead this assignment uses portable bitmap format (PBM) for images. This is very simple ASCII format. The first line of each file will contain the characters "P1" identifying its format. The next line will contain 2 integers, the width and height of the images. The remainder of the lines in the file contain '1's and '0's specifying the pixel values of the image. Here is an example

Here is **read_pbm.c** which contains a function to read PBM files:

```
int read_pbm(char filename[], int height, int width, int pixels[height][width]);
```

It is strongly recommended you use **read_pbm** in your assignment rather than writing your own code.

Part 1 - Digit Cracking

The first part of this assignment is to write a C program **crack_digit.c**.
crack_digit will be given one command line argument, an image filename.

The image will contain a single digit.

You program should print only a single line of output

This line of output should contain only the digit in the images.

For example:

```
./crack_digit digit/3_42.pbm

3

./crack_digit digit/7_99.pbm

7

./crack_digit digit/0_12.pbm

0
```

A dataset of 1000 example digit images is available to help you develop your program.
The week 7 lab exercises take you through getting started on **crack_digit.c**.

Part 2 - CAPTCHA Cracking

The second part of this assignment is to write a C program **crack_captcha.c**.
crack_captcha will be given one command line argument, an image filename.

The image will contain 4 digits.

You program should print only a single line of output

This line of output should contain only the 4 digit in the images.

For example:

```
./crack_captcha captcha/4224.pbm
```

```
4224
./crack_captcha captcha/9264.pbm
9264
./crack_captcha captcha/0053.pbm
0053
```

A dataset of 1000 example captcha images is available to help you develop your program.

Challenge CAPTCHA Cracking

The challenge part of this assignment is to identify more difficult captcha images with **crack_captcha.c**.
A dataset of 1000 example challenge captcha images is available to help you develop your program.

Testing

The script ~cs1511/bin/captcha_test will automatically test your programs on a random subset of a specified size of the supplied images:

```
~cs1511/bin/captcha_test --digit -n 10 crack_digit.c captcha.h other_C_files
dcc crack_digit.c read_pbm.c -o crack_digit
dcc crack_digit.c read_pbm.c --valgrind -o crack_digit-valgrind
Running 10 tests
Test digit/5_95.pbm passed
...
~cs1511/bin/captcha_test --captcha -n 20 crack_captcha.c captcha.h other_C_files
dcc crack_captcha.c read_pbm.c --valgrind -o crack_captcha-valgrind
Running 20 tests
Test captcha/8119.pbm passed
...
~cs1511/bin/captcha_test --challenge -n 30 crack_captcha.c captcha.h other_C_files
cc crack_captcha.c read_pbm.c -o crack_captcha
```

```
dcc crack_captcha.c read_pbm.c --valgrind -o crack_captcha-valgrind

Running 30 tests

Test captcha_challenge/1936.pbm passed

...
```

Experiment No.10:

Aim: A person on a nearby road is trying to enter into a Wi-Fi network by trying to crack the Password to use the IP Printer resource; write a program in Java/Python/C++ to detect such attempt and prohibit the access. Develop the necessary scenario by Using an IEEE 802.11; configure a Wi-Fi adapter and Access Point.

TITLE	**To detect an attempt to crack the password to use the resources in a Wi-Fi network.**
PROBLEM STATEMENT /DEFINITION	A person on a nearby road is trying to enter into a Wi-Fi network by trying to crack the Password to use the IP Printer resource. Write a program in Java/Python/C++ to detect such attempt and prohibit the access. Develop the necessary scenario by Using an IEEE 802.11, configure a Wi-Fi adapter and Access Point.
OBJECTIVE	• To understand the Wi-Fi networks and security in Wi-Fi network
S/W PACKAGES	**Operating Systems** 64-BIT Fedora 20 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards 2. Programming Tools (64-Bit) Scapy(Python)/Libcrafter(C++) APIs for packet capturing.
HARDWARE APPARATUS USED	• Wireless Access Point • WiFi adapter
REFERENCES	• Kurose, Ross "Computer Networking a Top Down Approach Featuring the Internet", Pearson; 6th edition (March 5, 2012), ISBN-10: 0132856204,ISBN-13: 978-0132856201. • Dipankar Raychaudhari, Mario Cerla,"Emerging Wireless Technologies and the Future Mobile Internet", Cambridge University Press, ISBN-13: 978-1-107-67864-4
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition

	4. Learning objective 5. Learning Outcome 6. Related Mathematics 7. Class Diagram 8. Concepts related Theory 9. Program code with proper documentation. 10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes).

- **Aim**

A person on a nearby road is trying to enter into a WiFi network by trying to crack the Password to use the IP Printer resource; write a program in Java/Python/C++ to detect such attempt and prohibit the access. Develop the necessary scenario by Using an IEEE 802.11, configure a Wi-Fi adapter and Access Point.

Prerequisites

- Concept of IEEE 802.11 Wi-Fi standards
    1. Access point
    2. SSID
    3. Authentication mechanisms
    4. Configuring Static IP on PC and Wireless Router
    5. Secure your network by configuring WAP key on Router
    6. Connect PC by using WAP key

- **Learning Objectives**
    - To understand the basics of Wifi networks.
    - To understand the functioning of AP.
    - To understand the Authentication of Wifi network.
    - Configure Static IP on PC and Wireless Router
    - Secure your network by configuring WAP key on Router
    - Connect PC by using WAP key
- **Learning Outcome:**

After successfully completing this assignment you should be:

- Able to configure Wifi Network.
- Able to capture Wi-Fi traffic and analyze it to detect intruders in the network.
- Provide necessary security mechanism to protect from unauthorized access.

- **Related Mathematics**

Mathematical Model

Let us consider **S** as a system for Wifi network.

S= {...

1. Identify the inputs

S = Start state

I = { Network Mode, Network Name, Security Mode}

Network Mode = {Mixed, Wireless-B/G only, Wireless-G only, Wireless-B only, Wireless-N only, Disabled }

Network Name (SSID) ={ You need to decide on a network name that is different than any other networks in your area, but remember that this information is public }
Security Mode = {WEP, WPA Personal, WPA2 Personal, and WPA2/WPA Mixed Mode}
DD = Deterministic data
NDD = Non deterministic data

Functions:

- **accessWifiNetwork():** input is beacon frame from access point available in the area.
- **captureAuthenticationFrames():** input is network interface in monitor mode to capture Wi-Fi frames.
- **getDevicesList():**from the authentication frames identify the list of devices with their MAC addresses trying to connect to Wifi network by sending multiple authentication frames with different passwords. Prepare the list of MACs of such devices to be restricted.
- **prohibitAccess(): configure access point to restrict the devices listed in getDevicesList() function.**

Output:

$O= \{O_1, O_2, O_3\ldots, O_n$ |'O' outcome of operations performed by the system**}**
Hence final S will comprise of:

$S= \{S, I, F_s, O, DD, NDD\}$ ϕ

<u>Data Structures used:</u>

For 'A': Network name to which host is to be connected.

'B': access to wifi network using password cracking tool.

'C': list containing the list of all the devices which are connecting to AP in the area.

'D': Entry is restricted for intruders.

Sr. No.	Mathematical Model	Description	Observation

1.	I = { Network Interface Mode, Network Name(SSID), Security Mode}	I be the configuration settings required for Wi-Fi network	AP is configured with: the security modes like WEP, WPA Network interface is changed to “Monitor” mode for packet capture.
2.	O= {O_1 }	**’O’** will store output of system	List of devices trying to connect to the AP and AP configuration to prohibit access to Wi-Fi network.
3.	F1(**accessWifiNetwork()=A**'	A’**=** {ssid’| ssid’ contains the network name**}**	Device is to be connected to the wifi network.
4.	F2 **captureAuthenticationFrames():** = B'	B’**=** {connected to Wifi network and WiFi (802.11) frames are captured, authentication frames are identified}	Using brute force attack, the intruder device is trying to crack wifi network password.
5.	F3(**getDevicesList()**)=C’	C’= {d | d device which are trying to connect to AP, Counted number of attempts from each device trying to connect to AP}	List of devices trying to connect to the AP.
6.	F4 (**prohibitAccess()**)= D'	D’={s| s security mechanism for access protection, if number of attempts exceeding the threshold AP is configured to restrict the access to WIFi network}	Entry restricted for intruders.

Success:

Finally, we can restrict intruder to access the Wi-Fi network by using the steps prescribed above successfully.

Failure:

With the help of the above procedure wifi is still insecure and not protected.

Experimental Setup:

1. **Access point (AP) connected to Internet**
2. **Wi-Fi adapter connected to Workstation**
3. **Intruder device trying to connect to AP with brute force attack**

Algorithm:

1. Setup Wireless network (802.11) with AP.
2. Configure Wi-Fi adapter in monitor mode to capture Wi-Fi frames.
3. Capture frames and identify its type and subtype.
4. Count number of authentication frames sent from each of the device in the Wi-Fi network.
5. Prepare the list of MAC addresses of the devices crossing some threshold limit of authentication requests.
6. Configure AP to restrict the access in the Wi-Fi network for the list of devices obtained in step 6.

Theory and Related Concepts:

IEEE 802.11 is a set of media access control (MAC) and physical layer (PHY) specifications for implementing wireless local area network (WLAN) computer communication in the 2.4 GHZ, 3.6 GHZ, 5 GHZ, and 60 GHz frequency bands.

Three major frame types exist in Wi-Fi networks:

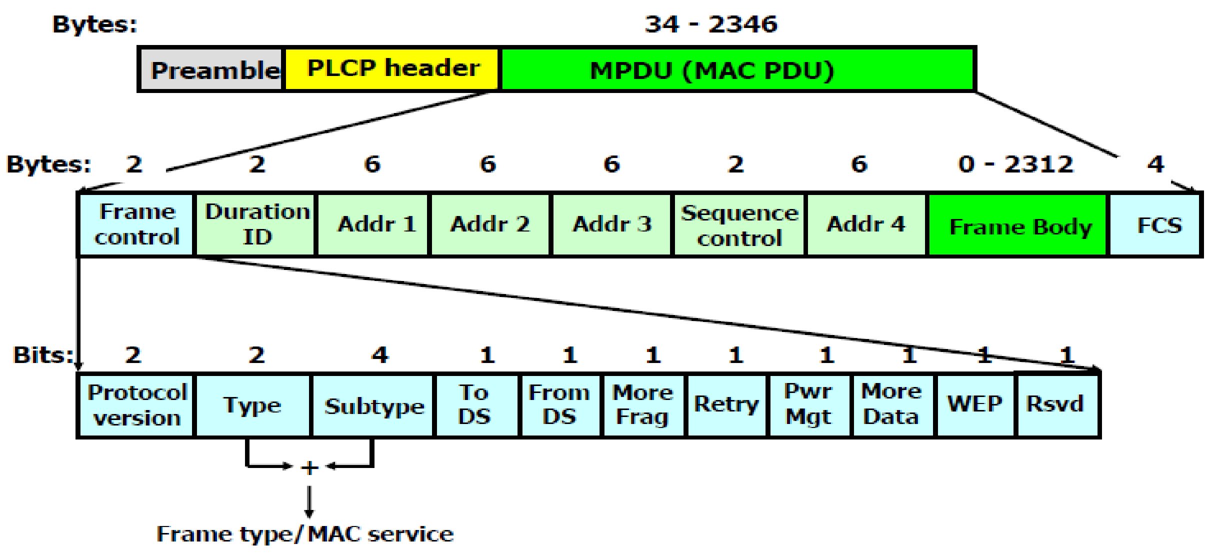
1. Data frames: data from station to station.
2. Control frames are used in conjunction with data frames to perform following operations.

- Area clearing operations
- channel acquisition
- carrier-sensing maintenance functions
- positive acknowledgment of received data

3. Management frames perform supervisory functions

- Join and leave wireless networks.
- Move associations from access point to access point.

802.11 frame format



Frame Control field:
Protocol Version: zero for 802.11 Standards
Type= frame type: a. Data, b. Management, c. Control
Subtype = frame sub-type
ToDS: When bit is set indicate that destination frame is for DS
FromDS: When bit is set indicate frame coming from DS

802.11 Frame type and subtypes: Listed in the table 1.

Authentication frame:

This is a frame signifying to the network membership within the wlan topology. 802.11 authentications is a process whereby the access point either accepts or rejects the identity of a radio NIC to create resources. Authentication restricts the ability to send and receive on the network. It is the first step for a device attempting to connect to an 802.11 WLAN. The function is handled by an exchange of management packets .Authentication is handled by a request/response exchange of management packets. The number of packets exchanged depends on the authentication method employed.

Association request frame:

802.11 associations enable the access point to allocate resources for and synchronize with a radio NIC. A NIC begins the association process by sending an association request to an access point. This frame carries information about the NIC (e.g., supported data rates) and the SSID of the network it wishes to associate with. After receiving the association request, the access point considers associating with the NIC, and (if accepted) reserves memory space and establishes an association ID for the NIC. Packets can show the current association of the sender. Association and Reassociation are handled by request/response management packets.

Association response frame:

An access point sends an association response frame containing an acceptance or rejection notice to the radio NIC requesting association and will include the Association ID of the requester. If the access point accepts the radio NIC, the frame includes information regarding the association, such as association ID and supported data rates. If the outcome of the association is positive, the radio NIC can utilize the access point to communicate with other NICs on the network and systems on the distribution (i.e., Ethernet) side of the access point.

Disassociation frame:

A station sends a disassociation frame to another station if it wishes to terminate the association. For example, a radio NIC that is shut down gracefully can send a disassociation frame to alert the access point that the NIC is powering off. The access point can then relinquish memory allocations and remove the radio NIC from the association table. Disassociation is a simple declaration from either an access point or a device.

Beacon frame:

The access point periodically sends a beacon frame to announce its presence and relay information, such as timestamp to help synchronize member stations with the BSS, , SSID, and other parameters regarding the access point to radio NICs that are within range. This purpose of this frame is to announce the beginning of a Contention Free period (CF), during which the right to transmit is conferred by the access point by polling. Radio NICs continually scan all 802.11 radio channels and listen to beacons as the basis for choosing which access point is having the best signal and availability to get associate with.

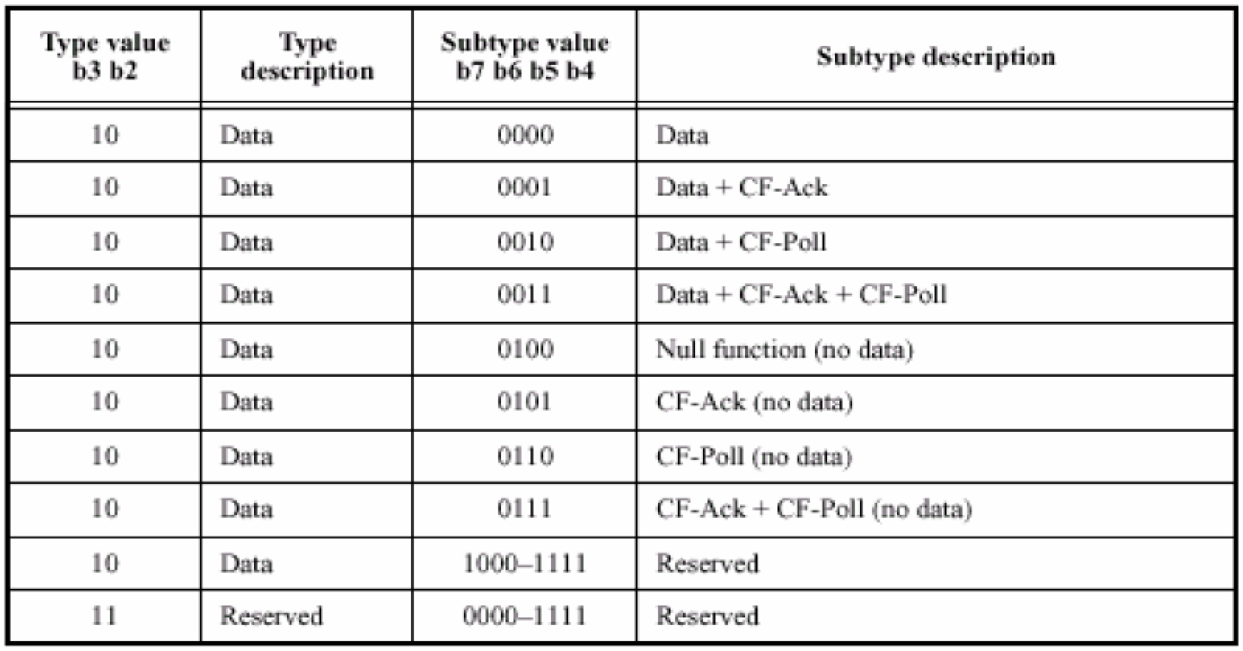
Probe request frame:

A station or client becomes active or on a PC when the wlan card it enabled it becomes active sends a probe request frame when it needs to obtain information from another station or access point. For After a radio NIC sends out a probe request to determine which access points are within range. The probe request frame is sent on every channel the client supports in an attempt to find all access points in range that match the SSID and client-requested data rates .Its upto the client to determine which access point to associate to by weighing various factors like supported data rates and access point load to select optimal access point thus moves to the authentication phase of 802.11 network after getting responses from Aps as probe response. This mechanism support also helps in roaming station the ability to move between cells while remaining connected in the search for new access point.

Table 1

Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
00	Management	0000	Association request
00	Management	0001	Association response
00	Management	0010	Reassociation request
00	Management	0011	Reassociation response
00	Management	0100	Probe request
00	Management	0101	Probe response
00	Management	0110–0111	Reserved
00	Management	1000	Beacon
00	Management	1001	Announcement traffic indication message (ATIM)
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101–1111	Reserved

Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
01	Control	0000–1001	Reserved
01	Control	1010	Power Save (PS)-Poll
01	Control	1011	Request To Send (RTS)
01	Control	1100	Clear To Send (CTS)
01	Control	1101	Acknowledgment (ACK)
01	Control	1110	Contention-Free (CF)-End
01	Control	1111	CF-End + CF-Ack



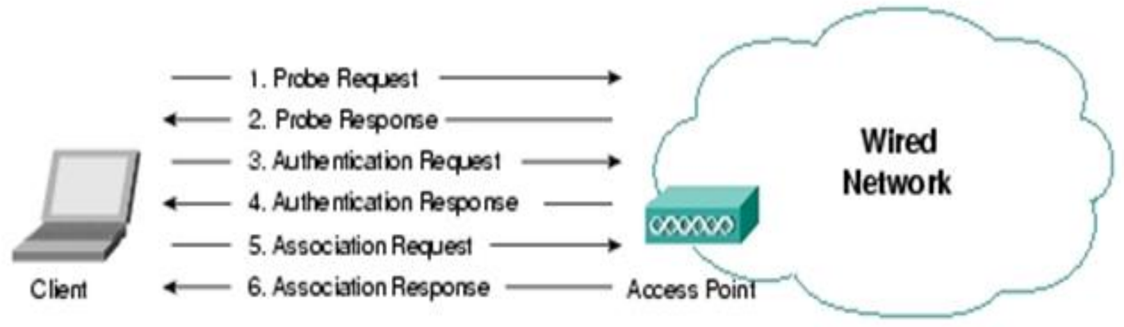
Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
10	Data	0000	Data
10	Data	0001	Data + CF-Ack
10	Data	0010	Data + CF-Poll
10	Data	0011	Data + CF-Ack + CF-Poll
10	Data	0100	Null function (no data)
10	Data	0101	CF-Ack (no data)
10	Data	0110	CF-Poll (no data)
10	Data	0111	CF-Ack + CF-Poll (no data)
10	Data	1000–1111	Reserved
11	Reserved	0000–1111	Reserved

802.11 Client Authentication Process:

Authentication in the 802.11 specification is based on authenticating a wireless station or device instead of authenticating a user.

As per the 802.11 specification client authentication process consists of the following transactions as mentioned below:

1. The Access points continuously send out Beacon Frames which are picked up by the nearby wlan clients.
2. The client can also broadcast on its own probe request frame on every channel
3. Access points within range respond with a probe response frame
4. The client decides which access point (AP) is the best for access and sends an authentication request.
5. The access point will send an authentication reply.
6. Upon successful authentication, the client will send an association request frame to the access point.
7. The access point will reply with an association response.
8. The client is now able to pass traffic to the access point.



Conclusion:

After successfully completing this assignment, you should be able to configure Wi-Fi Network and provide necessary security mechanism to protect from unauthorized access.

Experiment No.11

Aim: Write a computer forensic application program in Java/Python/C++ for Recovering Deleted Files and Deleted Partitions.

TITLE	Recovery of File & Partition
PROBLEM STATEMENT / DEFINITION	Write a computer forensic application program in Java / Python / C++ for Recovering Deleted Files and Deleted Partitions
OBJECTIVE	• Understanding the file system, how information is arranged, giving insight into where it can be hidden & how it can be recovered & analyzed. • To understand fundamentals of file systems such as FAT, EXT3, EXT4. • To demonstrate file recovery on files using metadata stored by journal on ext including merits & demerits. • To analyze what happens with ext FS with its journal on external device & how to deal with this situation.
S/W PACKAGES AND HARDWARE APPARATUS USED	1. Operating System : Latest version of 64 –Bit operating systems open source Fedora 20 or higher equivalent or Windows 8 2. Programming Tools : Latest 64 –Bit Programming languages or equivalent open source , Eclipse IDE / QT 64 –Bit platform / IDLE editor 3. Python 2.7 , Java , GCC , g++ 4. TSK 5. 32 / 64 –bit LATex software for documentation
REFERENCES	• Unix & Linux System Administration Handbook, Evi Nemeth , Garth Snyder , et al, Person Publication

	• Guide to Computer Forensics & Investigation, Bill Nelson, Amelia Phillips, christopher Steuart, Cengage Learning, Fourth Edition, ISBN 13 : 978-1435498839, ISBN 10 : 1435498836 • Digital Evidence& Computer Crime, Eoghan Casey Bs Ma Ac, ELSEVIER – Academic Press, 3rd Edition, ISBN 13 : 978-0123742681 , ISBN 10 : 0123742684
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning outcome 6. Related Mathematics 7. State transition diagram 8. Concepts related Theory 9. Program code with proper documentation. 10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes)

- **Aim:** Write a computer forensic application program in Java / Python / C++ for Recovering Deleted Files and Deleted Partitions.

- **Prerequisites**

  - Object oriented programming features.
  - FAT , EXT , Windows & Linux File System

- **Learning Objectives**

  - Understanding the file system, how information is arranged, giving insight into where it can be hidden & how it can be recovered & analyzed.
  - To understand fundamentals of FAT, EXT3, EXT 4.
  - To demonstrate file recovery on files using metadata stored by journal on ext including merits & demerits
  - To analyze what happens with ext FS with its journal on external device & how to deal with this situation

- **Learning Outcome**

  - After successfully completing this assignment, you should be able to
    - Able to use & demonstrate creation of filesystems
    - Able to create partitions.
    - Able to use file recovery mechanism.

Related Mathematics

Mathematical Model:
Let us consider **S** as a solution perspective of file & partition recovery system

$S = \{ St , E , I , O , F_{me} , DD , NDD , S_u , F \mid \varnothing \}$
where ,
St = initial state **image of the file / disk parititon**
Let L be the list of connected devices
$\therefore$ L = { inter , exter }
where inter is the set of disk partitions = { sda 1(windows) , sda 2 , … sda 10 }
Scope = { sda 7 , sda 9 , sda 10 }
exter = { fat 32 , … }

Assume Input I = { fname**,** inode no, }
Constraints = file recently deleted inode not allocated
here O be the output
O = { R_{full} , R_{part} } where R_{full} is fully recovered data & R_{part} be partially recovered data

NDD: the name of the partition in which file was created in Linux O.S.

Let F be the set of following functions
F_{jls}: to get list of inodes of deleted files
$F_{jcat:}$ to recover deleted file

- **Concepts and Related Theory**

Filesystem: It helps the devices to maintain physical location of file
Four main components of file system:
namespace: way to name things & organize them in hierarchy
API: set of system calls for navigating & manipulating objects
Security Model: Scheme of protecting , hiding & sharing things
implementation: s/w to tie logical model to the h/w

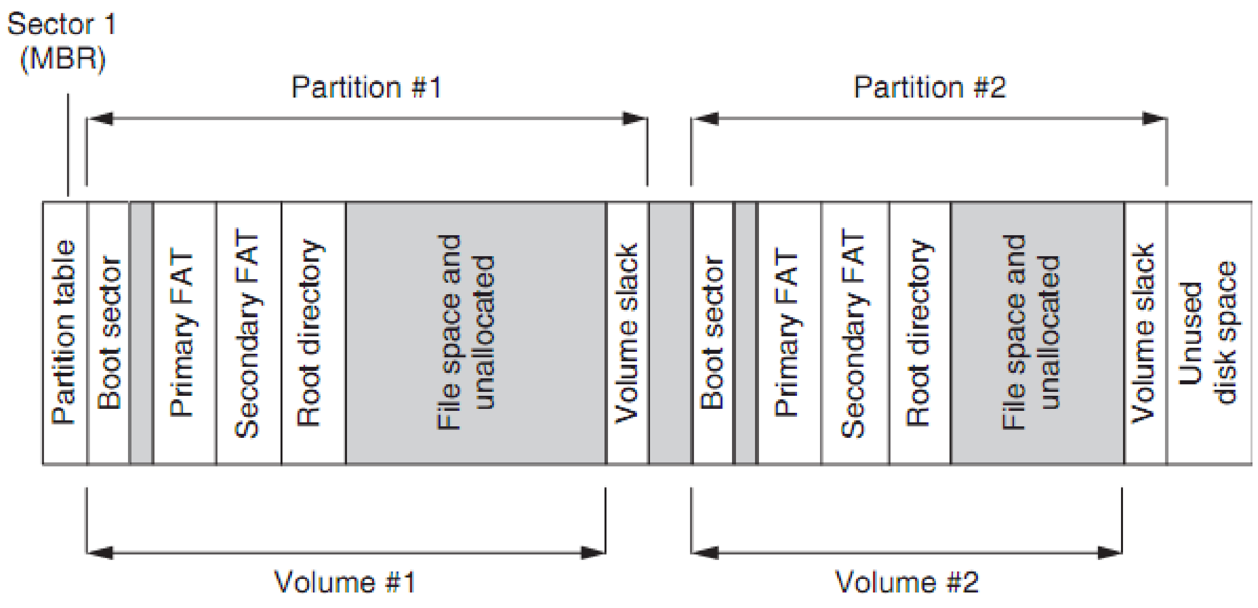


Fig: Simplified depiction of disk structure with two partitions, each containing a FAT formatted volume.

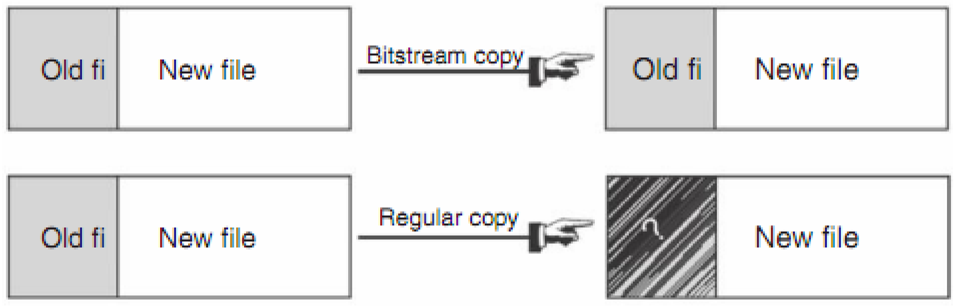
Five Layer Model:
1. File System
2. File Name
3. Metadata
4. Data / Content
5. Application

TSK Layered Approach:
File System Layer fsstat
Content Layer dcat dls
Inode / Metadata Layer icat ifind
Human Interface / File Layer

FAT : It is a section of disk is set aside to store FAT that first block in the file.
Entry in the first table corresponds to first block has a pointer to next block & so on until EoF

- ♣ Sequential file reads – very slow
- ♣ no security
- ♣ no well with large files
- ♣ simple
- ♣ records last access date & not last
- ♣ access time
- ♣ does not show starting cluster
- ♣ FAT12 (32 MB) , FAT 16 (2 – 4 GB) , FAT 32 (32 GB) exFAT (> 64 GB)

When old data are overwritten with new data, some of the old data can remain.



<u>**Journal Fundamentals:**</u>

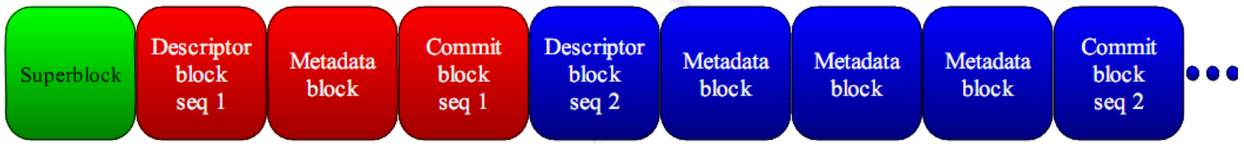
- ❖ Journaling File System is a type of file system that allows the OS to keep a log of all file system changes before writing the data to disk
- ❖ This type of file system offers better probabilities to avoid corruption in case of a power outage or system crash
- ❖ Journaling reduces time to perform filesystem consistency checks
- ❖ When file system operation occurs , required modifications are first written to journal if crash occurs during update, filesystem uses journal log to reconstruct

Superblock:
Record that describes characteristics of file system:

It contains:

- length of disk block
- size & location of inode tables
- disk block map
- usage information
- size of block groups
- important parameters of file system

Ext3 Journal general structure:



Steps for Installing TSK & Autopsy

*****Preferred installation in HOME DIRECTORY *****

The first step is to download and install TSK.

You can download TSK here:

https://sourceforge.net/projects/sleuthkit/files/latest/download

As of the time of writing this tutorial, the newest version is version 3.2.3.

Next, we need to extract TSK, compile it, and install it.

First, to extract TSK, open a terminal and change to the directory where TSK was downloaded to.

Then, run:
$ tar -xf sleuthkit-3.2.3.tar.gz to extract the files to sleuthkit-3.2.3

Now we need to compile TSK. To do that, first run:
$ ./configure

If there were no errors, run 'make'.
$ make

if there are any errors related to this jar file like -

```
make[2]: *** [all-local] Error 1

make[2]: Leaving directory `/home/tecomp/workspace/3228/sleuthkit-4.1.3/bindings/java'

make[1]: *** [all-recursive] Error 1

make[1]: Leaving directory `/home/tecomp/workspace/3228/sleuthkit-4.1.3/bindings/java'
```

make: *** [all-recursive] Error 1

THEN DOWNLOAD ivy.jar file from below link and copy is to /root/.ant/lib/ivy.jar

```
Getting: http://repo2.maven.org/maven2/org/apache/ivy/ivy/2.3.0-rc2/ivy-2.3.0-rc2.jar
    [get] To: /root/.ant/lib/ivy.jar
```

THEN TRY DOING MAKE AGAIN !

Because we plan on using Autopsy, we need to install TSK. To do that, run:

```
$ make install
```

Yay, we've installed the TSK. Now, time to do it for Autopsy.

As before, we need to download and compile Autopsy. However, this time we don't need to install it.

You can download Autopsy here:

https://sourceforge.net/projects/autopsy/fWiles/latest/download

As of the time of writing this tutorial, the newest version is version 2.24

Now again, extract the files with:

```
$ tar -xf autopsy-2.24.tar.gz
```

As part of its installation we will need to give Autopsy a location to store the evidence. Let's create that now in /home/<username>/evidence

```
$ mkdir /home/<username>/evidence
```

Now we need to compile autopsy. To do that, first run:

```
$ make
```

For now, select that you don't want to use the NSRL hash database (just hit enter)

Then it will ask for a folder to use as the evidence locker. This folder must exist.

Let's use the folder we made before:
/home/<username>/evidence

Now we can start Autopsy with `./autopsy`

If you start Autopsy this way you will need to open a web browser (like Firefox or Chrome) and open

http://localhost:9999/autopsy

Comparison:

Properties	**FAT**	**ext3**	**ext4**	**NTFS**
Year Introduced	1977	1999	2006	2001
Last access / read timestamps	✓	✓	✓	✓

ACL	×	✓	✓	✓
Security	×	✓	✓	✓
Journaling	×	✓	✓	×
	Windows Linux (with 3rd party driver)	✓	✓	✓

Conclusion: By performing this assignment students will be able to recover deleted files.

Experiment No.12

Aim: Write a C++/Java program for Log Capturing and Event Correlation.

TITLE	Log Capturing and Event Correlation.
PROBLEM STATEMENT /DEFINITION	Write a C++/Java program for Log Capturing and Event Correlation.
OBJECTIVE	• Log Capturing and Event Correlation.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards 2. Programming Tools (64-Bit) Latest Open source update of Eclipse Programming frame work, GTK+
REFERENCES	• Kurose, Ross "Computer Networking a Top Down Approach Featuring the Internet", Pearson; 6th edition (March 5, 2012), ISBN-10: 0132856204, ISBN-13: 978-0132856201. • Karen Kent, Murugiah Souppaya,"Guide to Computer Security Log Management", National Institute of Standards and Technology. • David Swift, "Successful SIEM and Log Management Strategies for Audit and Compliance", National Institute of Standards and Technology. • http://man7.org/linux/man-pages/man3/syslog.3.html
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Related Mathematics 7. Class Diagram 8. Concepts related Theory 9. Program code with proper documentation.

	10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes).

- **Aim:** Write a C++/Java program for Log Capturing and Event Correlation.

Prerequisites:

- Concepts of file handling in C++/Java
- Concept of Regular expressions
- Eclipse framework

Learning Objectives:

- To understand the event log concepts
- To understand the places where event logs are maintained.
- To capture and analyze the log files.
- To correlate the events.

Learning Outcome: After successfully completing this assignment, you should be able to:

- Capture and analyze the log files.
- Correlate the events.

- **Related Mathematics**

Mathematical Model
Let us consider **S** as a system for Wifi network.
S= {...
Identify the inputs
S = Start state
I = {Linux System logs}
O = {Events, Event Correlation}
DD (Deterministic data): Format of log files is deterministic
NDD (Non deterministic data): Number of logs generated

Operations Performed:

{readLogRecords(), analyzeLogs(), eventCorrelation() }

- **readLogRecords():** takes input as log file in the system for applications like ftp, yum logs, http logs, login attempts etc.
- **analyzeLogs():** Log file analysis for determining the patterns and store these patterns in the file.
- **eventCorrelation():** Correlate the events based on the output of analyzeLogs().

Output: O**= {O_1 |'O'** outcome of operations performed by the system**}** = Event correlation and display.

Hence final S will comprise of:

S= {S, I, F_s, O, DD, NDD} ϕ

Sr. No.	**Mathematical Model**	**Description**	**Observation**
7.	I = {Linux based System with network connection and syslog utility}	System is configured with syslog	System has various log files such as: var/log/maillog /var/log/httpd/access_log /var/log/mrtg /var/log/httpd **/var/log/yum.log**
8.	O= {O_1 }	**'O'** will store output of system	Output of log analysis and event correlation**.**
9.	F1 **readLogRecords()=A**'	A'**=** {logfile| logfile contains the application specific logs**}**	Application generates the logs. For example yum will generate the install logs in yum.log file.
10.	F2(**analyzeLogs()**) = B'	B'**=** {logs are analyzed based on various parameters like timestamps, success, failure etc. **}**	Log analysis output will be the number of attempts, total successes or failures, type of data for the application etc.
11.	F3(**eventCorrelation()**)=C'	C'={ Correlate the events based on the output of analyzeLogs()}	Generate the events based on log analysis. e.g. a *security attack* event based on 10 login failures.

Success:

Finally, we are able to do event correlation successfully.

Failure:

If unable to open system log files, we cannot do processing.

Concepts and Related Theory:

Event logs play a crucial role in security of the IT system. Today, many applications, operating systems, network devices and other system components are capable of writing security related event messages to log files. The BSD syslog protocol is an event logging standard supported by majority of OS and network equipment vendors, which allows one to set up a central log server for receiving and storing event messages from the whole IT system.

Since event logging is a widely accepted and well-standardized practice, there is a high chance that after a security incident has occurred in an IT system, there is (are) also event log message(s) for it in some log file(s).

In most cases event messages are appended to event logs in real-time as they are emitted by system components, event logs are an excellent source of information for monitoring the system, including security conditions that arise in it.

Event – a change in the system state, e.g., a disk failure; when a system component (application, network device, etc.) encounters an event, it could emit an event message that describes the event.

The central unit of information for any event correlation engine is the event. Events can be viewed as a generalized log records produced by various agents including standard UNIX syslog. They can be related to any significant change in the state of the operating system or application. Events can be generated for not only for problems but for successful completions of scheduled tasks.

For example, a host is being rebooted, attempt to log as an administrator, or a hard drive being nearly full.

- Event logging – a procedure of storing event messages to a local or remote (usually flat-file) event log.
- Many system components like applications, servers, and network devices have a builtin support for event logging (with the BSD *syslog* protocol being a widely accepted standard),
- In most cases event messages are appended to event logs in real-time, event logs are an excellent source of information for monitoring the system (a number of tools like Swatch and Logsurfer have been developed for log monitoring),
- Information that is stored to event logs can be useful for analysis at a later time, e.g., for audit procedures.

Syslog:

syslog is a utility for tracking and logging all manner of system messages from the merely informational to the extremely critical. Each system message sent to the syslog server has two descriptive labels associated with it that makes the message easier to handle.

- The first describes the function (facility) of the application that generated it. For example, applications such as mail and cron generate messages with easily identifiable facilities named mail and cron.

- The second describes the degree of severity of the message. There are eight in all and they are listed in Table 1:

Syslog Facilities:

Security Level	**Keyword**	**Description**
0	emergencies	System unusable
1	alerts	Immediate action required
2	critical	Critical condition
3	errors	Error conditions
4	warnings	Warning conditions
5	notifications	Normal but significant conditions
6	informational	Informational messages
7	debugging	Debugging messages

Table 1

The /etc/rsyslog.conf File:

The files to which syslog writes each type of message received is set in the /etc/rsyslog.conf configuration file. In older versions of Fedora this file was named /etc/syslog.conf.

This file consists of two columns. The first lists the facilities and severities of messages to expect and the second lists the files to which they should be logged. By default, RedHat/Fedora's /etc/rsyslog.conf file is configured to put most of the messages in the file /var/log/messages.

Sample:

```
*.info;mail.none;authpriv.none;cron.none           /var/log/messages
```

In this case, all messages of severity "info" and above are logged, but none from the mail, cron or authentication facilities/subsystems. You can configure syslog's /etc/rsyslog.conf configuration file to place messages of differing severities and facilities in different files. Almost all log files are located under /var/log directory and its sub-directories on Linux. Certain applications will additionally log to their own application specific log files and directories independent of the syslog.conf file. Here are some common examples:

/var/log
/var/log/maillog : Mail
/var/log/httpd/access_log : Apache web server page access logs
/var/log/samba : Samba messages
/var/log/mrtg : MRTG messages
/var/log/httpd : Apache webserver messages
/var/log/yum.log **:** Contains information that are logged when a package is installed using yum

Event correlation – a conceptual interpretation procedure where new meaning is assigned to a set of events that happen within a predefined time interval. During the event correlation process, new events might be inserted into the event stream and original events might be removed.

Event correlation is a procedure where a stream of events is processed, in order to detect (and act on) certain event groups that occur within predefined time windows.

Examples:

- if 10 *login failure* events occur for a user within 5 minutes, generate a *security attack* event.
- if both *device internal temperature too high* and *device not responding* events have been observed within 5 seconds, replace them with the event *device down due to overheating*.

Approaches: A number of approaches have been proposed for event correlation (rule-based, codebook based, neural network based etc.

A number of event correlation products are available on the market (HP ECS, SMARTS, NerveCenter, RuleCore, LOGEC, etc.

- **Rule-based** (HP ECS, IMPACT, RuleCore, etc.) – events are correlated according to the rules *condition → action* that are specified by the human analyst.
- **Codebook based** (SMARTS) – if a set of events $e_1,...,e_k$ must be interpreted as event A, then $e_1,...,e_k$ are stored to the *codebook* as a bit-vector pointing to A. In order to correlate a set of events, look for the most closely matching vector in the codebook, and report the interpretation that corresponds to the vector.
- **Graph based** – find all dependencies between system components (network devices, hosts, services, etc.) and construct a graph with each node representing a system component and each edge a dependency between two components. When a set of fault events occurs, use the graph for finding possible root cause(s) of fault events (e.g., 10 “HTTP server not responding” events were caused by the failure of a single network link).
- **Neural network based** – a neural net is trained for the detection of anomalies in the event stream, root cause(s) of fault events, etc.

Open-source tools developed for monitoring event logs in real-time:

- Swatch
- Logsurfer

Simple tasks performed by the tool:

Raise an alarm immediately after a certain message has been appended to a log file.

Event processing tasks involve:

Event correlation is a conceptual interpretation procedure where new meaning is assigned to a set of events that happen within a predefined time interval.

Event correlator: A software application that implements event correlation

During the interpretation procedure, the correlator might create new events and hide original events from the end user.

Example: Processing of login failure events.
Although an individual login failure event might be a symptom of a password cracking attempt, it could also indicate that the user accidentally typed a wrong password.
Therefore, one can't simply configure the log file monitoring tool to send an immediate alert on the occurrence of login failure log message, since this could result in a high number of false positives.
To reduce the number of false alarms, one or both of the following event correlation schemes can be used:

1. Once N login failure for user X events have been observed during the last T seconds, generate the excessive number of login failures for user X event and send it as an alarm to the security administrator.
2. If the login failure for user X event appears and during the next T seconds no successful login for user X event will appear, generate the login failure not followed by success for user X event and send it as an alarm to the security administrator.

Linux applications use the syslog utility to export all their errors and status messages to files located in the /var/log directory. This can be used in correlating the timing and causes of related events on your system. It is also important to know that applications frequently don't display errors on the screen, but will usually log them somewhere.

SEC - simple event correlator:

SEC is an event correlation tool for advanced event processing which can be harnessed for event log monitoring, for network and security management, for fraud detection, and for any other task which involves event correlation.

SEC is a lightweight and platform-independent event correlator which runs as a single process. SEC reads lines from files, named pipes, or standard input, matches the lines with patterns (like regular expressions or Perl subroutines) for recognizing input events, and correlates events according to the rules in its configuration file(s).
SEC can produce output by executing external programs (e.g., *snmptrap* or *mail*), by writing to files, by sending data to TCP and UDP based servers, by calling precompiled Perl subroutines, etc.

Conclusion:

After successfully completing this assignment, you should be able to correlate the logs to events.

Experiment No.13

Aim: Configure and demonstrate use of vulnerability assessment tool like Wireshark or SNORT

TITLE	Configure and demonstrate use of vulnerability assessment tool like Wireshark or SNORT

PROBLEM STATEMENT / DEFINITION	Implementation of following spoofing assignments using C++ multicore Programming a) IP Spoofing b) Web spoofing
OBJECTIVE	• To understand & determine the relevance of spoofing attack • To understand the flaws of TCP/IP packets • To understand IP header fields , 3 – way handshaking concept & difference between stateful & stateless protocols • To ethically perform IP & web spoofing • To understand 802.11
S/W PACKAGES AND HARDWARE APPARATUS USED	1. Operating System : Latest version of 64 –Bit operating systems open source Fedora 20 or higher equivalent or Windows 8 with Multicore CPU equivalent to Intel i5 / 7 generation onwards or Kali Linux 2. Programming Tools : Latest 64 –Bit Programming languages such as Microsoft Visual Studio (ver. 12 or Higher) or equivalent open source , Eclipse / QT 64 –Bit platform 3. Networked computer with internet access 4. Wireless Access Point & wireless LAN card 5. Wireshark 5.0 – Ethereal software / LanExplorer 6. G++ , libcrafter , Crawlers / Web spiders 7. 32 / 64 –bit LATXT
REFERENCES	• ‘ Simple Active Attack Against TCP ’, Laurent Joncheray; April 1995 • RFC 793 (for TCP) • https://github.com/pellegre/libcrafter (install crafter 0.3) • www.kali.org (bootable Kali Linux)
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning outcome 6. Related Mathematics 7. State transition diagram 8. Concepts related Theory 9. Program code with proper documentation. 10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes)

a) **Aim:** Configure and demonstrate use of vulnerability assessment tool like Wireshark or SNORT

- **Prerequisites**
    - Object oriented programming features.
    - Understanding QT framework
    - File handling

- Understanding web hacking requires an appreciation for the strong conceptual links between web resources & their functionality

- **Learning Objectives**
  - To understand & determine the relevance of spoofing attack
  - To understand the flaws of TCP/IP packets
  - To understand IP header fields & 3 – way handshaking concept
  - To ethically perform IP & web spoofing

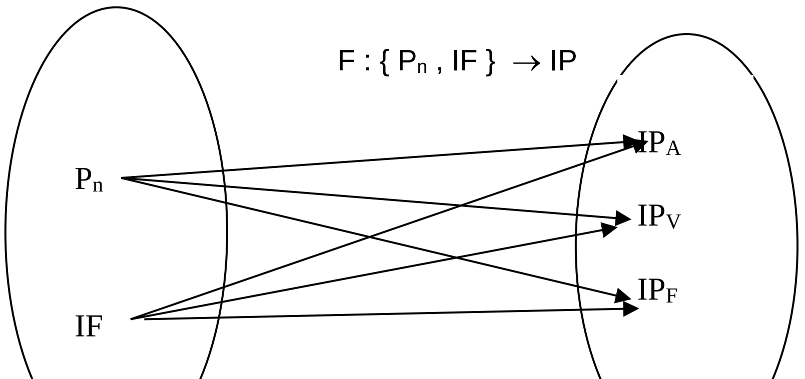
- **Learning Outcome**: After successfully completing this assignment, you should be able to
  - Able to implement IP & Web spoofing & certain ways to perform the same.
  - Able to identify reasons of IP spoofing & web spoofing attacks, to locate vulnerabilities.

- **Related Mathematics**

Mathematical Model
Let us consider **S** as a solution perspective of the email header analysis system

$S = \{ St , E , I , O , F_{me} , DD , NDD , S_u , F \mid \varnothing \}$
Where,
St = attacker machine in LAN , ISN
LAN represented in terms of graph G = (V , E) or Tree T = (V , E) based on T_{topo}
Here
$V = \{ v_1 , v_2 , \ldots , v_n \}$ vertices denotes host machines
$\therefore \quad n = | V |$

$E = \{ e_1 , e_2 , \ldots , e_n \}$ edges denotes links for connection or possible communication
$T_{topo} = \{ star , mesh , bus \}$
E = end state = close connection
Let $I = \{ IP , P_n , IF , LO \}$
where ,
$IP \rightarrow$ netid + hosted (32 bits)
$IP = \{ IP_A , IP_V , IP_F \}$
Assume ,
IP_A = IP address of Attacker machine in LAN where program of spoofing resides
IP_V = IP address of Victim machine in LAN
IP_F = IP address of Fake machine in LAN
Assume, UV_{url} List of unvisited URL
O be the output = exploited trusted relation among entities
Suppose P_n is the set of unspecified port numbers = 2^{16}
$IF = \{ IF_{eth0} , IF_{wlan0} \}$ denotes necessary interface for wired & wireless respectively
LO = 127.0.0.1 i.e. local host



domain range

F_{me} : { S_n ,

Where f (S_n) denotes generalized relative sequence number – MIT GNU scheme

e.g. | V | = | E | = 4 in mesh topology

G can be represented by M of size n × n –

i. Incident Matrix based on topology

viz; In Mesh Topology we get Identity Matrix under legitimate nodes

$M = [M_{ij}]$

`

$$M_{ij} = \begin{cases} 1 & \text{; when edge } e_j \text{ is incident with } v_i \\ 0 & \text{; otherwise} \end{cases}$$

$$M_I = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline v_1 & 1 & 1 & 1 & 1 \\ v_2 & 1 & 1 & 1 & 1 \\ v_3 & 1 & 1 & 1 & 1 \\ v_4 & 1 & 1 & 1 & 1 \end{array}$$

if we assume v_3 as attacker machine

v_1 as victim machine

v_4 as fake machine

$$M_I = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline v_1 & 1 & 1 & 1 & 1 \\ v_2 & 1 & 1 & 1 & 1 \\ v_3 & 0 & 0 & 0 & 0 \\ v_4 & 1 & 1 & 1 & 1 \end{array}$$

ii. Path Matrix / Reachability Matrix P

$$P_{ij} = \begin{cases} 1 & \text{; when } \exists \text{ path / communication possibility from } v_i \text{ to } v_j \\ 0 & \text{; otherwise} \end{cases}$$

Rewriting start state as St

$$P_{ij} = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 1 & 1 & 0 & 1 \\ v_2 & 1 & 1 & 0 & 1 \\ v_3 & 0 & 0 & 1 & 0 \\ v_4 & 1 & 1 & 0 & 1 \end{array}$$

Reflexive relation = loop back i.e. 127.0.0.1

O =

$$P_{ij} = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 1 & 1 & 1 & 1 \\ v_2 & 1 & 1 & 0 & 1 \\ v_3 & 1 & 0 & 1 & 0 \\ v_4 & 0 & 1 & 0 & 1 \end{array}$$

Algorithm:

1. Monitor the traffic in between the legitimate nodes i.e. nodes having trust relationship
2. Know the IP addresses of Victim machine & Fake machine
3. Generate a trust relationship between Attacker & Victim by sending SYN & getting SYN / ACK
4. Store ISN of Victim machine at Attacker machine
5. Terminate the connection (RST / FIN)
6. Initiate SYN connection from attacker using any other IP address of the same domain (Fake machine)
7. Make DoS attack on Fake machine with the purpose to block SYN/ACK to or NACK (RST) from Fake machine
8. Make a bluff ACK within RTT by adding next sequence number of Victim from it forged ISN
9. Perform the spoofing with multiple clients which may lead to multicore way of programming

Algorithm :

1) Crawler will copy the site on some host
2) The hosted fake sites link will be shared with victims
3) Victim clicks on fake site link
4) Victim is presented with same UI on the fake site as of legitimate site
5) Web spoofing achieved

Writing a Crawler:

```
while( list of unvisited URLs is not empty )
{
        take URL from list
        fetch content
        record whatever it is you want to about the content
        if content is HTML
        {
                parse out URLs from links
                for each URL
                {
                if it matches your rules & its not already in either the visited or unvisited list
                        add it to the unvisited list
                }
        }
}
```

- **Concepts and Related Theory**
  Introduction to Spoofing :

Computer on a network pretends to have identity of another computer, usually one with special access privileges, so as to obtain access to the other computers on the network.

Types of Spoofing:

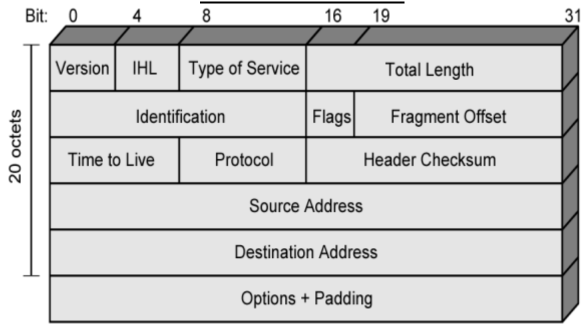
- IP spoofing
- ARP spoofing
- Email spoofing
- Web spoofing
- DNS spoofing

IP Spoofing:

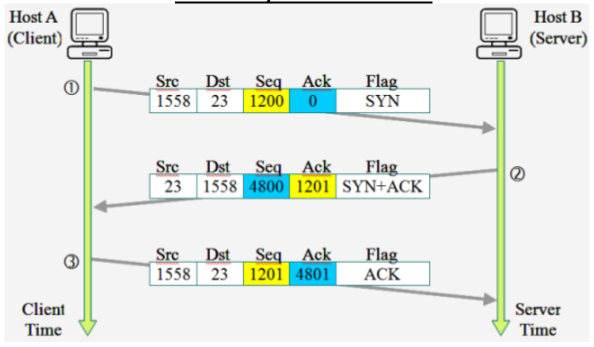
Typically involves sending packets with spoofed IP addresses to machines to fool the machine into processing the packets. It is a process of that enables attacker to hide his real identity when communicating with the target system; therefore , the data packets the attacker send will appear to originate at another system. It is the creation of IP packets using somebody else's IP source addresses.

It is required to study :

IP header fields



3 – way Handshake

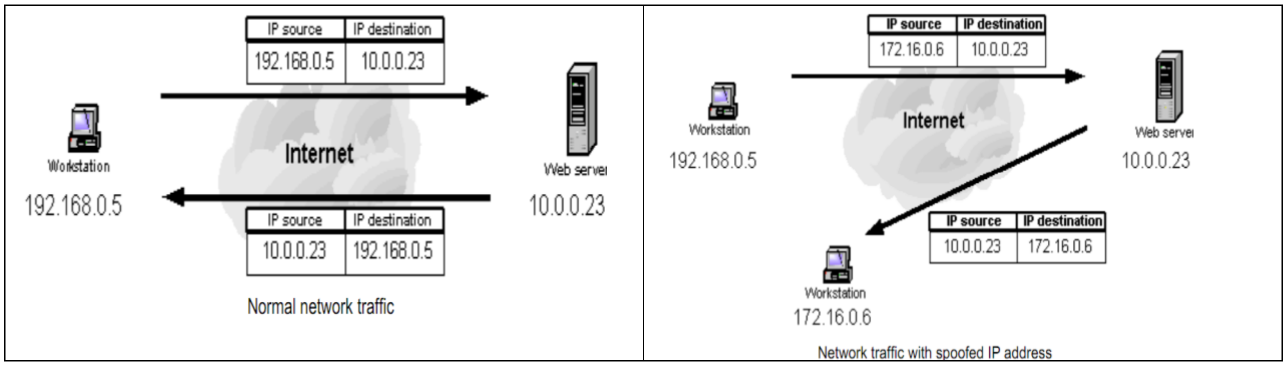


Examining the IP header, we can see that the first 12 bytes contain various information about the packet. The next 8 bytes contains the source & destination IP addresses.

20 bytes ≤ Header Size < 2^4 x 4 bytes = 60 bytes
20 bytes ≤ Total Length < 216 bytes =65536 bytes

With the program an attacker can easily modify these addresses – specifically the "source address” field. Two general techniques are used during IP spoofing :

i. A hacker uses an IP address that is within the range of trusted IP addresses.
ii. A hacker uses an authorized external IP address that is trusted

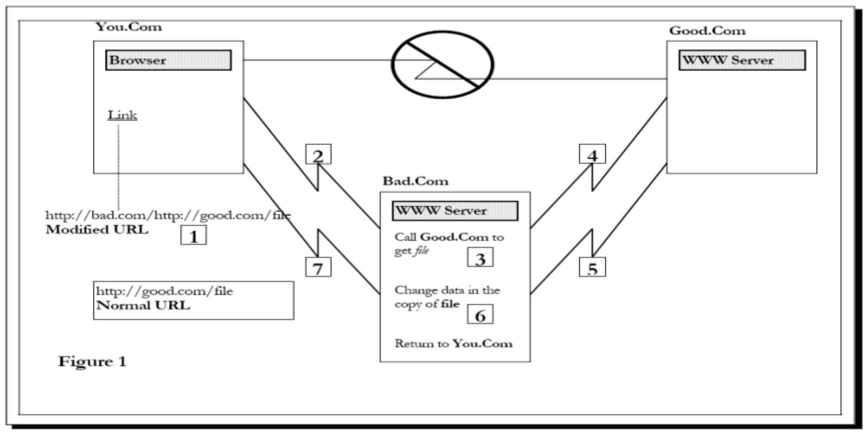


IP Spoofing is easy because Routers look at Destination addresses only & authentication is based on Source addresses only. To change source address field in IP header field is easy. It is the smoothest & hardest attack. IP spoofing can also be a method of attack used by network intruders to defeat network security measures, such as *authentication* based on IP addresses. IP spoofing attack can be classified into –

i. Non – blind spoofing
ii. Blind spoofing

Web Spoofing:

Here is how a spoofed web session flows.



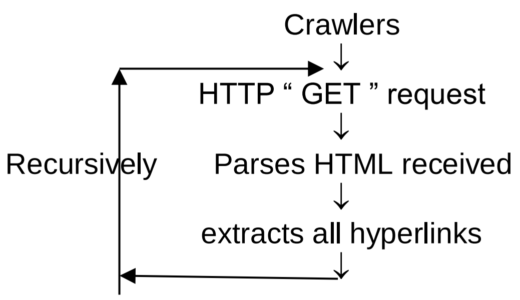
1. Somehow, the URL that your client browser is referencing on You.Com has been prefaced with the address of the intermediary web site – Bad.Com.
2. Your browser determines that Bad.Com should handle the URL request .
3. The web server on Bad.Com reads the URL and determines that the you are actually trying to reach a file on Good.Com.
4. The Bad.Com server calls Good.Com and asks for the specified file.
5. The Good.Com server returns the page as requested.
6. At this point, the Bad.Com server can change its copy of the page you asked for.
7. After these changes, the Bad.Com server returns the page to you.

Web Crawler / ant / spider / bot / web scutter …

Crawling Website begins with first web page & follow every link found. It is a program to automate

Hyperlinking
<a href = “url”>   </a>
<form action = “ url ”> </form>
<object>
<applet>

<u>Crawling a site means BFS on connected directional graph.</u> Drawback of Crawling is that it can’t interpret client side scripting.



Applications of crawlers:
- ✓ most crawlers are helpful
- ✓ help search engines index pages & perform routine maintenance by testing HTML code & hyperlinks
- ✓ crawlers can identify hyperlinks on webpages – particularly email addresses
- ✓ create lists of URL to visit, also known as “ crawler frontier ”

Different Type of Web Spoofing:
1. DNS server spoofing attack
2. Content theft
3. Subdomain spoofing

Wireshark:

Wireshark is a free network protocol analyzer for UNIX, Linux, Mac OS X, & Windows. It allows the investigator to examine data from a live network or from a capture file on disk. The investigator can see all traffic being passed over the network interface into promiscuous mode.

Features of Wireshark:

- Capture data can be browsed via GUI or command line
- Capture files can be programmatically edited
- Display filters can be used to selectively highlight & color packet summary information
- Hundreds of protocols can be dissected

Hence, the output of IP spoofing can be checked in any protocol analyzer tool e.g. Wireshark. To install it
```
$ yum install Wireshark 5.0 gnome
```

Print of a program along with its output should be highlighted for
- Legitimate IP address
- Spoofed IP address

- Sequence number
- Spoofed list mapping to multicore programming

Spoofing in Multicore way:

- Many PCs are connected in an internetwork connection with trust relation among them
- Assume multiple victims / targets to be attacked with spoofing
- Create different threads; in every thread the attacker machine will be attacking different victims
  Pthread_create (thread_id ,   , fun_spoof, NULL)
- Acquire the availability of several cores by get affinity
- These created different threads can be assigned to different cores (CPU mask)
- The spoof function is given as 3rd parameter to the threads created with different victims as input
- Therefore, spoofing achieved simultaneously for multiple victims parallelly

Conclusion:

Thus, after successfully completing this assignment, you should be able to understand Wireshark Tool.

Experiment No.14

Aim: Design and implementation of Honeypot.

TITLE	Honeypot
PROBLEM STATEMENT / DEFINITION	Design and Implement a Honeypot.
OBJECTIVE	• To understand working of honeypot • To learn how to design and implement it.
S/W PACKAGES AND HARDWARE APPARATUS USED	• Operating System : Latest version of 64 –Bit operating systems open source Fedora 20 or higher equivalent Multicore CPU equivalent to Intel i5 / 7 generation onwards • Programming Tools- Eclipse 64 –Bit platform • Networked computer with internet access • Wireless LAN card • G++ , libcrafter, Python 2.7 , Java • 32 / 64 –bit LATXT
REFERENCES	• "Honeypots: tracking Hackers", Lance Spitzner, Addison Wesley, ISBN-13: 007-6092017530, 2003 • "Data Communications and Networking", Behrouz A Forouzan, McGraw Hill, 2007 • http:// www.en.wikipedia.org/wiki/Honeypot_(computing) • www.os3.nl/_media/2005-2006/rp1/js_mm_report.pdf

STEPS	Refer to theory, algorithm, create a database, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	• **Date** • **Assignment no.** • **Problem definition** • **Learning objective** • **Learning outcome** • **Related Mathematics** • **State transition diagram** • **Concepts related Theory** • **Program code with proper documentation.** • **Output of program.** • **Conclusion and applications (the verification and testing of outcomes)**

Aim: Design and Implement a Honeypot.

- **Prerequisites**
- Concepts of IDS, IP addresses and headers.
- C++/ python programming knowledge
    - **Learning Objectives**
    - To understand working of honeypots
    - To understand design and implementation of a honeypot.

- **Learning Outcome**:
    - Students should be able to design a honeypot
    - Students should be able to implement a honeypot

Related Mathematics

Mathematical Model
Let S= { S, I, Fs, ∂, O, DD, NDD}
Where,
S= Start state
I= Input to the system= request( ) to server from hacker
∂=Transition on different Input (I*i*) by the system
Fs=Final State
O= {O1, O2, O3…, On |'O' outcome of operations performed by the honeypot}
DD= Deterministic data= IP address of honeypot, IP address of server.
NDD= Non-deterministic data= attack patterns

Related Theory:
In computer terminology, a honeypot is a trap set to detect, deflect, or, in some manner, counteract attempts at unauthorized use of information systems. Generally, a honeypot consists

of a computer, data, or a network site that appears to be part of a network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers.

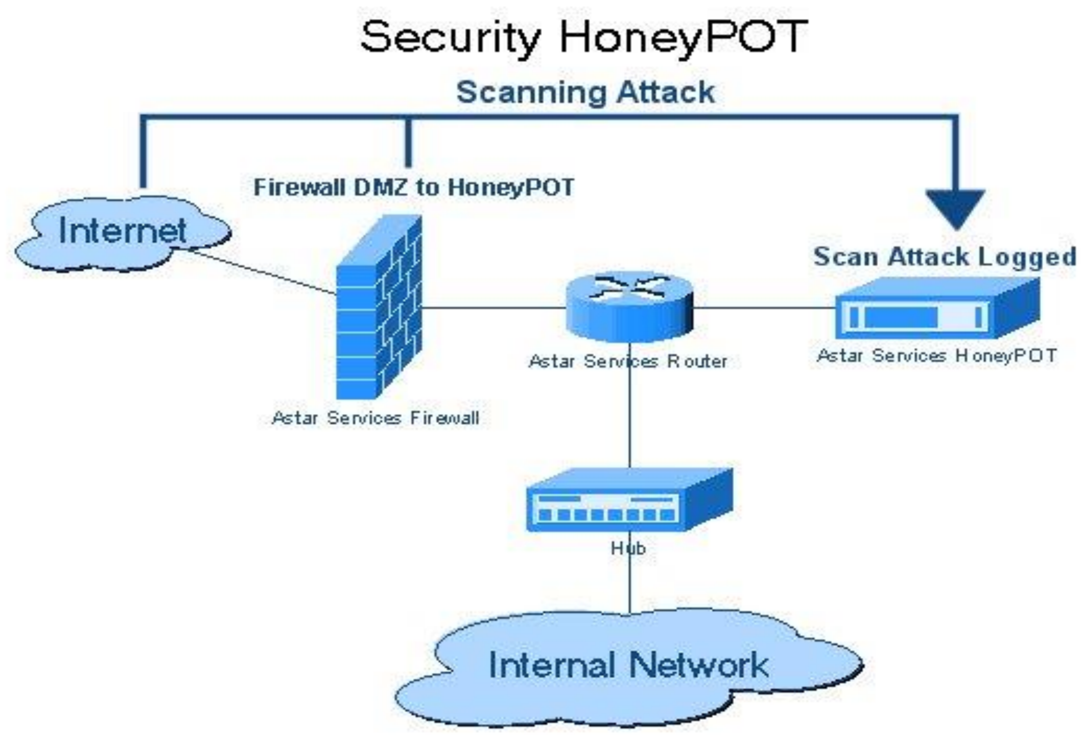


Fig: Scenario of honeypot

Honeypots can be classified based on their deployment (use/action) and based on their level of involvement. Based on deployment, honeypots may be classified as:

- Production honeypots
- Research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots do.

Research honeypots are run to gather information about the motives and tactics of the Blackhat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats.[1] Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations.

Based on design criteria, honeypots can be classified as:-

- Pure honeypots
- High-interaction honeypots
- Low-interaction honeypots

Pure honeypots are full-fledged production systems. The activities of the attacker are monitored by using a casual tap that has been installed on the honeypot's link to the network. No other software needs to be installed. Even though a pure honeypot is useful, stealthiness of the defense mechanisms can be ensured by a more controlled mechanism.

High-interaction honeypots imitate the activities of the production systems that host a variety of services and, therefore, an attacker may be allowed a lot of services to waste his time. By employing virtual machines, multiple honeypots can be hosted on a single physical machine. Therefore, even if the honeypot is compromised, it can be restored more quickly. In general, high-interaction honeypots provide more security by being difficult to detect, but they are expensive to maintain. If virtual machines are not available, one physical computer must be maintained for each honeypot, which can be exorbitantly expensive. Example: Honeynet.

Low-interaction honeypots simulate only the services frequently requested by attackers. Since they consume relatively few resources, multiple virtual machines can easily be hosted on one physical system, the virtual systems have a short response time, and less code is required, reducing the complexity of the virtual system's security. Example: Honeyd.

Detection of honeypots:

Just as honeypots are weapons against spammers, honeypot detection systems are spammer-employed counter-weapons. As detection systems would likely use unique characteristics of specific honeypots to identify them, a great deal of honeypots in use makes the set of unique characteristics larger and more daunting to those seeking to detect and thereby identify them. This is an unusual circumstance in software: a situation in which "versionitis" (a large number of versions of the same software, all differing slightly from each other) can be beneficial. There's also an advantage in having some easy-to-detect honeypots deployed.

Honeynets:

Two or more honeypots on a network form a honeynet. Typically, a honeynet is used for monitoring a larger and/or more diverse network in which one honeypot may not be sufficient. Honeynets and honeypots are usually implemented as parts of larger network intrusion detection systems. A honeyfarm is a centralized collection of honeypots and analysis tools.

Conclusion: After successfully completing this assignment, we can detect an attack, re-route it and track the attacker.