

Spring 2012

ONLINE MONITORING USING KISMET

Sumit Kumar
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Kumar, Sumit, "ONLINE MONITORING USING KISMET" (2012). *Master's Projects*. 243.

DOI: <https://doi.org/10.31979/etd.rexc-dkr7>

https://scholarworks.sjsu.edu/etd_projects/243

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

ONLINE MONITORING USING KISMET

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Sumit Kumar

May 2012

© 2012

Sumit Kumar

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

ONLINE MONITORING USING KISMET

by

Sumit Kumar

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2012

Dr. Mark Stamp Department of Computer Science

Dr. Chris Pollett Department of Computer Science

Dr. Cay Horstmann Department of Computer Science

ABSTRACT

Online Monitoring using Kismet

by Sumit Kumar

Colleges and universities currently use online exams for student evaluation. Students can take assigned exams using their laptop computers and email their results to their instructor; this process makes testing more efficient and convenient for both students and faculty. However, taking exams while connected to the Internet opens many opportunities for plagiarism and cheating.

In this project, we design, implement, and test a tool that instructors can use to monitor the online activity of students during an in-class online examination. This tool uses a wireless sniffer, Kismet, to capture and classify packets in real time. If a student attempts to access a site that is not allowed, the instructor is notified via an Android application or via Internet. Identifying a student who is cheating is challenging since many applications send packets without user intervention. We provide experimental results from realistic test environments to illustrate the success of our proposed approach.

ACKNOWLEDGMENTS

I would like to thank Dr. Mark Stamp for trusting me with this idea and for his continued support and guidance throughout the course of the project.

I would like to thank my committee members Dr. Chris Pollett and Dr. Cay Horstmann for providing their valuable feedback. I would also like to thank Kevin Ross for his help in setting up the lab environment to test this project and Debra J. Caires for helping me in project report.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
1.1	Project Motivation	2
1.2	Aim and Objective	6
2	Background and Related Work	8
2.1	Types of Online Test	8
2.2	Types of Cheating	9
2.2.1	Using Spyware	9
2.2.2	Using IM	10
2.2.3	Connecting to Different Network	10
2.2.4	Changing IP/MAC Address	10
2.2.5	Using Proxy Server	11
2.2.6	Using VPN	11
2.3	Sniffing and Packet Sniffers	11
2.4	Sniffing Tools	13
2.4.1	Wireshark	13
2.4.2	TCPDump	13
2.4.3	NetStumbler	14
2.4.4	Kismet	15
2.5	Related Work	17
2.5.1	SeCOnE	17

2.5.2	Triangulation	19
3	Design	22
3.1	Approach	22
3.2	Development Tools	23
3.2.1	Kismet	23
3.2.2	Tshark	30
3.2.3	XAMPP Server	31
3.2.4	Eclipse for Android application	32
3.2.5	Python	33
3.3	Software Design	33
3.3.1	Online Test Design	33
3.3.2	Kismet .pcapdump Parser	36
3.4	System Design	36
3.5	Database Design	38
3.6	Mobile Application Design	40
4	Implementation and Results	45
4.1	Experiment 1	45
4.2	Experiment 2	47
4.3	Experiment 3	49
4.4	Experiment 4	52
5	Conclusion	59
6	Future Work	60

LIST OF FIGURES

Figure		Page
1	MAC address changing software	11
2	Wireshark Snapshot	14
3	TCPDump capture snapshot	15
4	NetStumbler screenshot	16
5	Kismet Screenshot	17
6	System architecture of SeCOnE	18
7	Students cheating by taking help from outside	20
8	Students cheating being at a particular distance	20
9	Student trying to cheat by communicating with neighbor student	20
10	Kismet architecture	24
11	Kismet Startup GUI	25
12	Kismet Startup GUI	26
13	Kismet GUI for adding sources	27
14	Kismet GUI for adding sources	28
15	Kismet GUI showing packet capturing	29
16	XML log file showing card type	29
17	XML log file showing network information	30
18	XML log file showing clients information	31
19	Home page of the online test	35
20	System Design	38
21	Database Design	39

22	Android Application main screen	41
23	Adding a website to the white-list	42
24	Adding a website to the black-list	43
25	Student Details	44
26	Experiment 1 System Architecture	46
27	Experiment 2 System Architecture	48
28	Experiment 3 System Architecture	51
29	Database entries for all the students	52
30	Results of Experiment 4	54
31	Results of Experiment 4	55
32	Results of Experiment 4	55
33	Results of Experiment 4	56
34	Results of Experiment 4	56
35	Results of Experiment 4	57
36	Results of Experiment 4	57
37	Results of Experiment 4	58
38	Results of Experiment 4	58

CHAPTER 1

Introduction

An online test is an activity where an individual uses a computer connected to Internet and completes an exam. For example, there may be a portal or website where students submit their answers to exam questions. In some cases, the answers may be evaluated automatically and results are shown for students to view immediately.

The benefits of an online test include the following.

- Online tests are more efficient, from both the instructor's and student's perspectives.
- The online exams are more secure as each student might have a different set of questions so they are not be able to help one another during an exam. Although a paper based text can also have different question sets, which requires substantial human effort, an online test can be easily configured to generate sets in an automated manner.
- The administration of online exams is easier, provided monitoring can be completed using a computer. The instructor would be freed from spending excess time trying to detect cheating.
- Online tests are environmentally friendly, as they reduce need for paper.

Increasingly, colleges and universities are using online exams for student evaluation [4, 13, 23, 25]. These online tests are generally taken by students using their laptops. Since students are connected to the Internet during the tests, this creates

many opportunities for cheating. The main aim of this project is to develop an application that can be used by instructors to monitor an online exam and, thereby, detect possible cheating. In our application, the instructor provides a white-list of websites that students are allowed to access. Any attempt to access a website that is not on the white-list is logged and the instructor is notified in real time.

1.1 Project Motivation

At San Jose State University (SJSU) most instructors use paper exam. Very few instructors administer their exams online and essentially this is very restricted and the environment is manually controlled. After considering the above benefits regarding online tests and the current scenario of our college, a tool was designed that can be easily used by professors for administering online exams.

Online exams can be helpful for students and professors even though they give students an opportunity to cheat during the exam. Since students will now be given a test on their own laptop, they can at any time connect to the Internet and look for the answers to the exam's questions. If cheating, the instructor can challenge the student and can easily close their browser. Thus, it becomes essential to stop students from cheating. There are many ways students can cheat:

- Connecting to the Internet and searching on common search engines such as Google, Yahoo, Bing and/or DogPile.
- Connecting to a proxy server and searching for answers from a search engine.
- Using a peer-to-peer chat client during the exam for chatting among other students.
- Connecting to IRC or forums in order to ask questions regarding exam solutions.

The above list is not complete as there are more ways a student can cheat. Most cases revolve around being connected to an external IP. If there is a way this activity can be detected, we can successfully make the test more secure.

Following the above guidelines there has been an attempt to solve the problem of cheating during exams within our college. A tool was developed [32] which can monitor applications running on remote systems. The main aim [32] was to use client/server architecture to develop a tool that displays the student's details and any websites visited during the test to the instructor. Using this tool, the instructor can add sites to the white-list and the blacklist as well as monitor all the student's activities during the exam. By viewing at the logs and the reports, the instructor can easily determine if any student was trying to cheat.

For this project students were asked to connect to a specific SSID which was created by a wireless USB router called Windy31. This router was connected to the instructor's machine and students were required to connect solely to Windy31 router. The students are required to have WindowsTM Operating System for the test. Once connected to Windy31 router, students were asked to switch off their WindowsTM firewall and anti-virus. They were also required to change the WindowsTM workgroup to MSHOME. After setting the above changes students were asked to install a program that was written in Java; they also were required to have Java installed on their machines. This tool uses technology Remote Method Invocation RMI (explained below) to call remote methods on the server. Using this technology Java RMI can send all student's computer details to the server, such as how many processes (software) are running, and how many windows in the browser are open.

The tool was divided into two parts client and the server program. The client

program needs to run on student computers; each student was given an executable file to run on their own machine. Once installed successfully, the server part of the software was also installed on the instructor's machine. The client program uses the WindowsTM "tasklist.exe" to get information regarding all the running processes on the students' computers. It then makes a list of all the applications running on the student's machine along with their headers and then sends it to the server. The client program sends this data by using a Remote Method Invocation [17, 26]. RMI, a Java application programming interface, enables the programmer to create distributed Java applications where the methods of remote Java objects can be invoked by any other Java virtual machine. This process means that the students computer can call a method or function of an object that is present on the server (instructor's computer). The server exposes these methods and objects that can be called by any client that is connected. By these methods the students can obtain a list of all the running programs on their computers and the server is able to extract useful information.

A Graphical User Interface (GUI) was also developed for instructors so they can see and monitor their students who are connected to the router and what programs were running on their system. Using this GUI, the instructor can add websites to white and black lists. The GUI that was on the server, also on the instructors computer, was able to generate notifications by opening a pop-up window displaying details of students activity. Any student who connects to the system or disconnects from the system or opens any restricted websites opens a pop-up window on the server. This way instructor was able to monitor any given test.

Using GUI professors were able to track student activities; however instructors were not able to have full control over student activity. Students were still able to disconnect from the router and connect to another wireless network. By connecting

to another network students can easily search on the Internet for answers to their questions and then again re-connect to the instructor's router. Their activities during that period remain hidden from the instructor.

Currently installing software on student laptops, restricting students to a WindowsTM operating system, and connecting to an MSHOME workgroup, switching off the firewall and anti-virus for the duration of the test does not seem practical. It may not be practical for our college, as students have different operating systems such as MAC OSX, Ubuntu and others. Installing software on multiple systems raises the issue of privacy and security concerns. Also, the server was only able to connect a few students at one time.

An idea presented [32] looks very interesting and the approach to solve the problem of monitoring online test seems good; however solution is not practical enough to be implemented within our college. The barrier against implementing within our college is that students use many different operating systems: Windows 7, Mac OS-X, Ubuntu, Fedora and others. Students using the latest version of WindowsTM(Windows 7) can't use this software as the User Access Control (UAC) built-in will not allow the software to obtain the list of processes from the task manager. The older version of WindowsTM(Windows XP) does not allow students to change the name of the running application. The tool will always report the wrong name to the server. By looking at the logs, the instructor will never know if the name is a real application or if it is some other application. The Windy31 wireless adapter is also a major reason why this can't be used within our college. There is a limit on the number of computers connecting to the USB wireless adapter. In a class where more than 25 students have to take test, this tool will not be able to connect all the computers to the server. If students are not connected to the server, then they cannot

be monitored and the main purpose of this project is defeated. Students can easily disconnect from the wireless network, connect to another wireless network, search for their answers, and then come back and connect to the original network. This activity will only generate an alert message on their instructor's computer; the instructor cannot tell what the student did during that period when he/she was disconnected from the test network. Also, if the student has more than one network card, they can connect to the test server with one and another network with the other; the student will still be able to cheat. The above shortcomings were taken as a challenge and became the main motivation for our project. There were other projects taken into consideration for our project. These projects will be discussed in detail in Chapter 2 Related work.

1.2 Aim and Objective

The main aim of this project is to build a monitoring tool that can help instructors conduct online tests within the classroom and monitor students in real time. Students will take an online test using their laptops and will be connected to the Internet. Although students have Internet access, they will be allowed to use only a few websites. The tool should be able to monitor all the sites a student visits and should alert and generate proper notification, if a student tries to open any restricted site.

To monitor all the activities in the network, the monitoring tool should be able to capture all the packets flowing through the network. To capture all the packets, a wireless sniffer should be used. Only those packets should be filtered out that belong to the student taking the exam and these packets should be scanned for any restricted activity. The packet's data will tell us all the details that we want to know

about the student's activity. The main objective of our project is to use a wireless sniffer to capture all the packets when students are taking online tests and then scan them to find out if any student has tried to open any link that was not allowed. An additional mobile application will be developed that can help instructors obtain instant notifications on their Android phones if any suspicious activity occurs during the exam.

CHAPTER 2

Background and Related Work

To design an online monitoring tool it is essential to know the required components. Therefore, in this section we will explore and study the different components that may be required for our project. We discuss the following in the remainder of this chapter:

- Types of online tests
- Various types of cheating possible in an online test
- Packet sniffers
- Related work

2.1 Types of Online Test

The online tests can be referred to as Computer-Assisted Assessment or Computer-Aided Assessment (CAA) and Computer Based Assessment (CBA) [13]. The term CAA is mainly used to refer to a test where a computer is used in the assessment process. CBA is mainly used to refer to tests where automated responses are generated by the system after evaluating the student's answers. These two objective based online tests can be referred to as formative and summative tests [13]. The formative tests are the exams that a tutor gives while teaching a course that gives information about the learning state of each student. This type of information gathering helps students, as well as tutor, to improve a student's learning process. Summative tests are conducted at the end of the course/module and give a measure-

ment of how much a student has learned from that course/module. There are many tools available on the market today, that are designed according to the above points. The main features of any good online tests should be:

- The exam should be secure so that the content of the exam is not be revealed until the exam starts.
- The exam should be able to create multiple question sets from a pool of questions so that each student gets a different set of questions.
- The exam should set the same level of difficulty for all created sets.
- Supervising, managing and controlling the exam should be easy.
- If possible, an automated evaluation of the answers is used for giving instant feedback about the student's performance.

2.2 Types of Cheating

Online tests are useful but they are also vulnerable. A student can try and cheat on the exam and pass which can discourage honest students. There are many ways a student can cheat in an online test. The main methods used are as follows:

2.2.1 Using Spyware

Spyware [34, 35] is a type of malware that can secretly gain access inside a computer and can collect information about that computer. The computer user might not be aware of the spyware and that it is collecting the user's personal data without their knowledge. Collected data can then be used by the person who installed the spyware on that computer. Using a malware program, a student can spy on a professor's

computer containing exam questions. Students can also gain more information about quizzes and other tests that a professor might be planning to give. After obtaining the information, students might use it for themselves to perform well on their exam or they might also try to sell the information to others for monetary gains [25].

2.2.2 Using IM

Students can use instant messaging (IM) software to chat among themselves during the exam. They can discuss exam questions and collaborate on answers.

2.2.3 Connecting to Different Network

Students can disconnect from the exam network (one they are not allowed to use the Internet) and can connect to another open network in order to search the Internet. After searching, they can again connect to the original exam network and can submit their answers.

2.2.4 Changing IP/MAC Address

Students can change their IP/MAC address while taking the exam; thereby they are not visible to any monitoring system. Changing the network adapter settings can help a student become untraceable. IP addresses can be changed easily, but a MAC address is a unique value associated with the network card. Although it's unique, there are a few software programs that can help students change it. For example in Figure 1, a tool known as "Technitium MAC Address Changer" [36] can help a user change their MAC address.

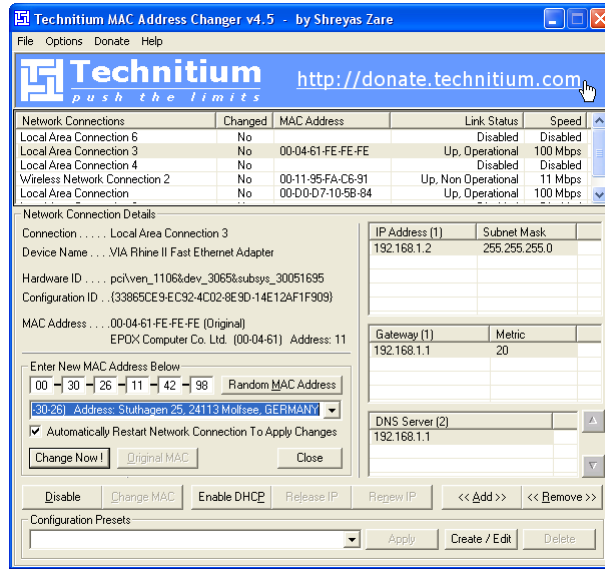


Figure 1: MAC address changing software

2.2.5 Using Proxy Server

Using a proxy server [28] a student can redirect Internet traffic through a computer or a router. Students can also setup their own proxy or they can use one of the many free proxies available on Internet to redirect traffic. In this way, they can redirect all their traffic for searching and finding answers during the exam.

2.2.6 Using VPN

Students can setup a virtual private network (VPN) at their home that is encrypted. Using this encrypted VPN students can search for the answers to exam questions and still remain unnoticed.

2.3 Sniffing and Packet Sniffers

Sniffing [37] is a technique where data flows through a network and can be captured and analyzed for information gathering. The tools that help capture the data

flowing within a network are known as Sniffers. These tools are either software or hardware [33]. The sniffers are also capable of reading packets flowing through the network if the network is not encrypted. There are mainly three types of sniffing methods [31]:

Type 1 - IP based Sniffing: in this type of sniffing a network card is put into promiscuous mode. In this mode all the packets flowing around the network card are passed to a computer rather than frames. These packets can then be analyzed based on an IP address. For example, a tool known as an “IP Sniffer 1.99.3” [16] can sniff all the IP packets flowing around within the network.

Type 2 - MAC based Sniffing: is a type of sniffing similar to the IP based sniffing however the packets are now analyzed by matching a MAC address. For example, a tool known as “Wake-on-LAN Packet Sniffer” [22] can help in sniffing all the packets by matching a MAC address in a given network.

Type 3 - ARP based Sniffing: is a method of sniffing that is a little different from the above two. In this method the network interface card is not put into a promiscuous mode. This is because the ARP packets are stateless. First, the ARP cache of the host whom we want to sniff is poisoned. Next, the ARP packets are sent to the computer sniffing packets directly, rather than going to the destination directly. This method is used in a switched network. The IP of the sniffer computer is put into the victim’s computer in such a way that it will always send the packet to the sniffer computer first; the sniffer can then store the packet and later can analyze it. This type of sniffing is also known as, Man in the middle attack. Tools such as Ettercap and AntiSniff can help in ARP sniffing.

2.4 Sniffing Tools

There are many different tools that are used to sniff packets within a network for either managing networks or for finding weakness within the network. These tools are also used by hackers to gather information about a given network. Next, we discuss the most common sniffers.

2.4.1 Wireshark

Wireshark [38] is the most well-known and used open source network packet analyzer. It was originally known as Ethereal. Wireshark provides a good user interface to show the details of captured packets. As shown in Figure 2, Wireshark can give detailed information about the packet captured such as source address, protocol name, header details and the body of the packet. Wireshark is used for many purposes such as:

- Testing network security
- Troubleshooting network problems
- Studying network protocols
- Debugging network software

There is also a command line version of Wireshark called TShark which we have used in our project.

2.4.2 TCPDump

TCPDump is an UNIX based tool that is also used for packet sniffing. TCPDump also works similar to Wireshark [10]. It is a network analyzer that was developed by

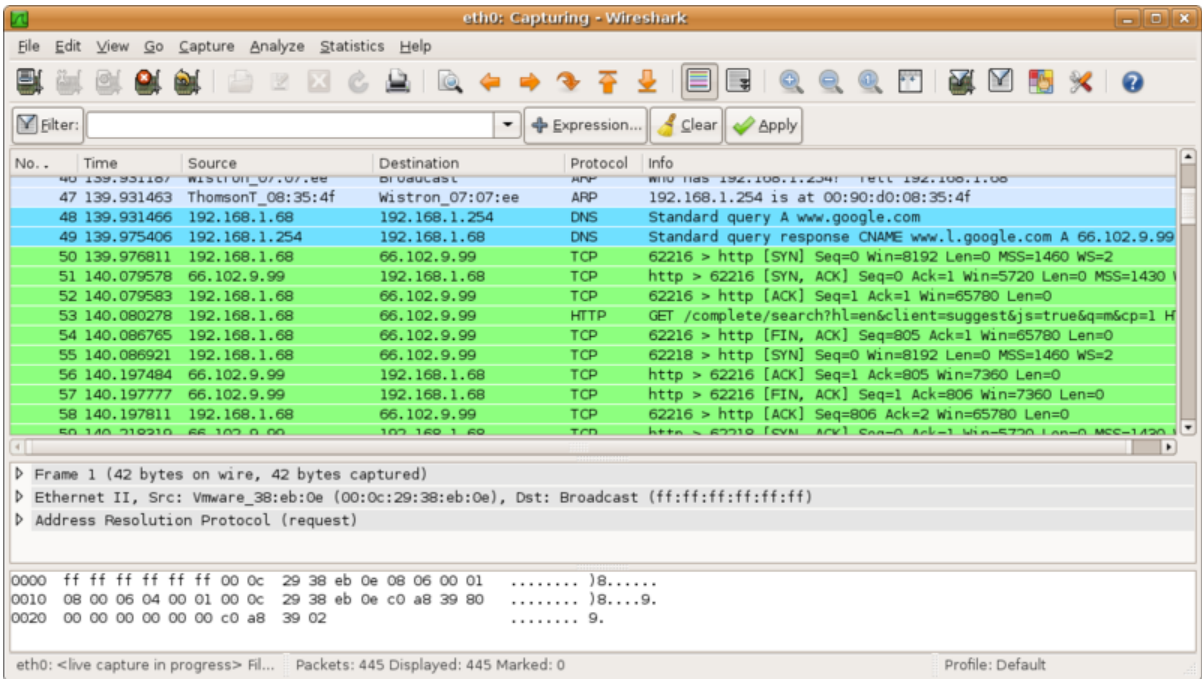


Figure 2: Wireshark Snapshot

Van Jacobson and is mainly operated through a command line interface. TCPDump analyses all important network protocols such as TCP, UDP, IPV4, ICMPv4, IPv6, ICMPv6 and SNMP. As shown in Figure 3, we can see that TCPDump can capture packets within the network as well as display the source and destination IP addresses.

2.4.3 NetStumbler

NetStumbler [14] is also a packet sniffer designed for Windows™ that can detect LANs and WANs using 802.11b, 802.11a and 802.11g standards. NetStumbler is mainly used for:

- Wardriving
- Verifying network connections
- Finding network strengths in any region


```

root@Homeserver:~# tcpdump -c 6 tcp and dst host 192.168.1.52
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
21:31:52.397327 IP 74.125.230.145.http > 192.168.1.52.43567: . 3105919908:3105921326(1418) ack 92059312 win 221 <nop,nop,timestamp 2275048695 48435831>
21:31:52.399478 IP 74.125.230.145.http > 192.168.1.52.43567: . 1418:2836(1418) ack 1 win 221 <nop,nop,timestamp 2275048695 48435831>
21:31:52.400901 IP 74.125.230.145.http > 192.168.1.52.43567: P 2836:3866(1030) ack 1 win 221 <nop,nop,timestamp 2275048695 48435831>
21:31:52.403619 IP 74.125.230.145.http > 192.168.1.52.43567: . 5866:5284(1418) ack 1 win 221 <nop,nop,timestamp 2275048695 48435831>
21:31:52.405224 IP 74.125.230.145.http > 192.168.1.52.43567: P 5284:6410(1126) ack 1 win 221 <nop,nop,timestamp 2275048695 48435831>
21:31:52.604279 IP 74.125.230.146.http > 192.168.1.52.54852: P 3109908297:3109908443(146) ack 91234453 win 135 <nop,nop,timestamp 2275058567 48436071>
6 packets captured
7 packets received by filter
0 packets dropped by kernel
[root@Homeserver ~]#

root@Homeserver:~# tcpdump -c 6 tcp and dst host 192.168.1.52 -q
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
21:31:52.397327 IP 74.125.230.145.http > 192.168.1.52.43567: t cp 1418
21:31:52.399478 IP 74.125.230.145.http > 192.168.1.52.43567: t cp 1418
21:31:52.400901 IP 74.125.230.145.http > 192.168.1.52.43567: t cp 1030
21:31:52.403619 IP 74.125.230.145.http > 192.168.1.52.43567: t cp 1418
21:31:52.405224 IP 74.125.230.145.http > 192.168.1.52.43567: t cp 1126
21:31:52.604279 IP 74.125.230.146.http > 192.168.1.52.54852: t cp 146
6 packets captured
7 packets received by filter
0 packets dropped by kernel
[root@Homeserver ~]#

```

Figure 3: TCPDump capture snapshot

- Detecting unauthorized access points

NetStumbler sends a probe request every second and listen for the response. The response contains information about the network such as SSID, BSSID and MAC. This type of sniffing is also known as active sniffing since the probe requests that are sent to the wireless access point can be tracked easily. NetStumbler can also detect the physical location of network devices by using a GPS device. As shown in Figure 4, we can see that NetStumbler can sniff networks as well as help in wardriving shown by the graph (below).

2.4.4 Kismet

Kismet [21] is an open source network packet sniffer, packet analyzer, network detector and also an intrusion detection system. It works with any card that supports

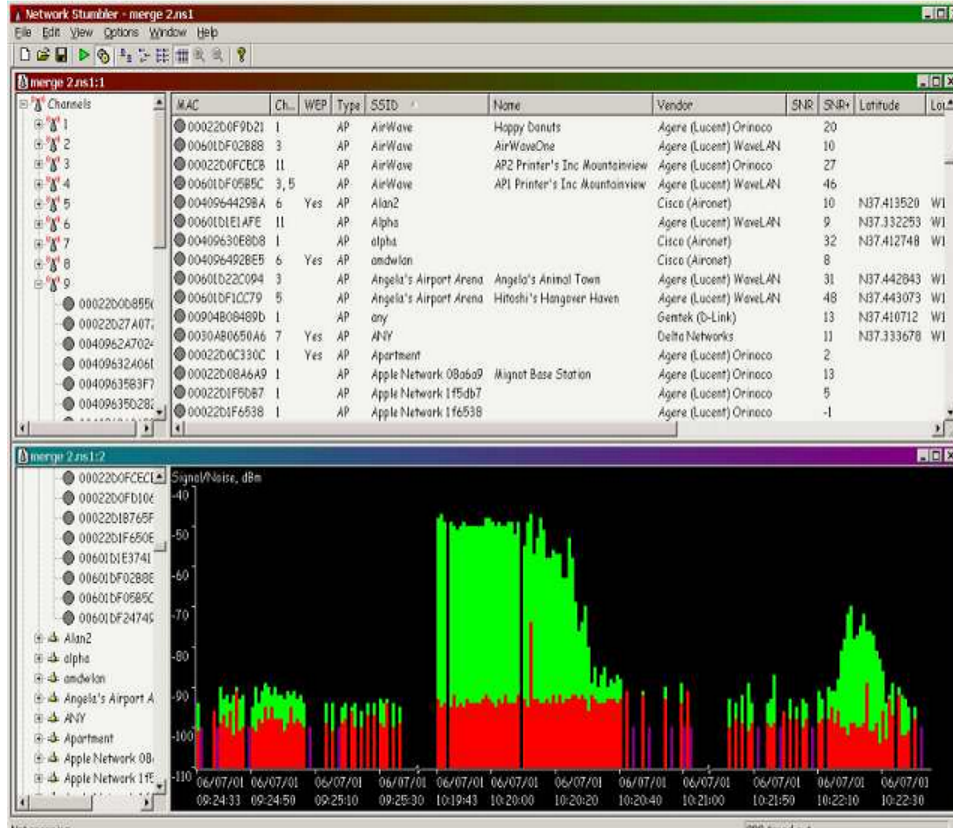


Figure 4: NetStumbler screenshot

a raw network monitoring mode also known as a rfmon mode. Kismet can also sniff packets following the 802.11b, 802.11g, or 802.11n wireless standards. It is very easy to configure Kismet; there is a very detailed packet capture data since it can hop through all the channels and can analyze the entire wireless network. Kismet can also detect hidden networks as shown in Figure 5. The main benefits of Kismet over others are as follows:

- It puts the card into a monitoring mode which is not attached to any network

- It scans all the wireless networks passively so it remains undetected
- It can scan the entire spectrum and all the wireless networks nearby
- It generates different types of logs thus giving full information about the network

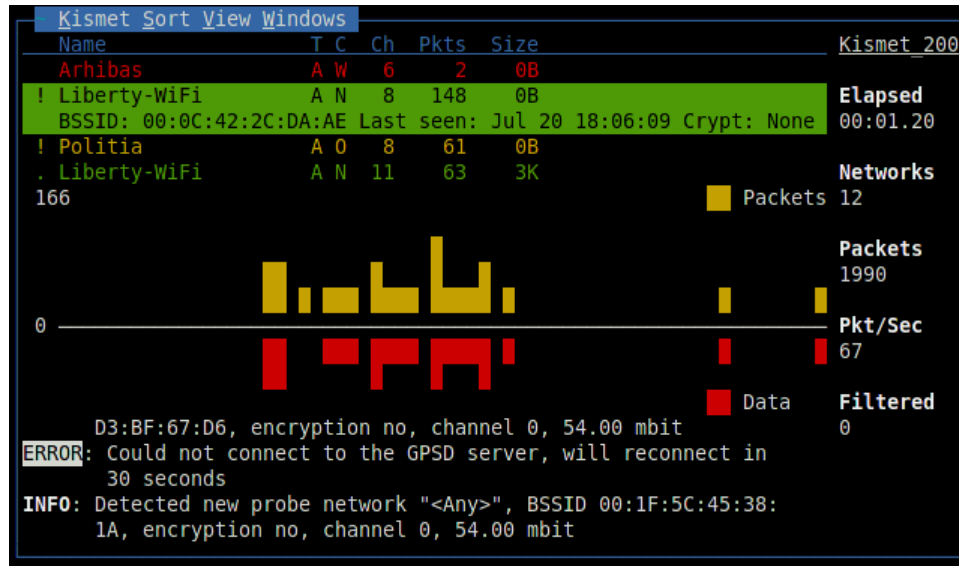


Figure 5: Kismet Screenshot

2.5 Related Work

There are many projects aimed at detecting online exam cheating. Some examples that were studied for this project are as follows:

2.5.1 SeCOne

The research work as described in [15] helps in monitoring students who are taking online tests at a remote location where a proctor is not present. Since the proctor is not present, a computer system “Exam Admin Group Agent” and “Examinee Group Agent” are used to monitor these exams as shown in Figure 6. This research attempted to remove human presence completely by using a secure online

exam management. The researchers used an enhanced Security Control system during the Online Exam (SeCO_nE), which is based on group cryptography and e-monitoring scheme [15].

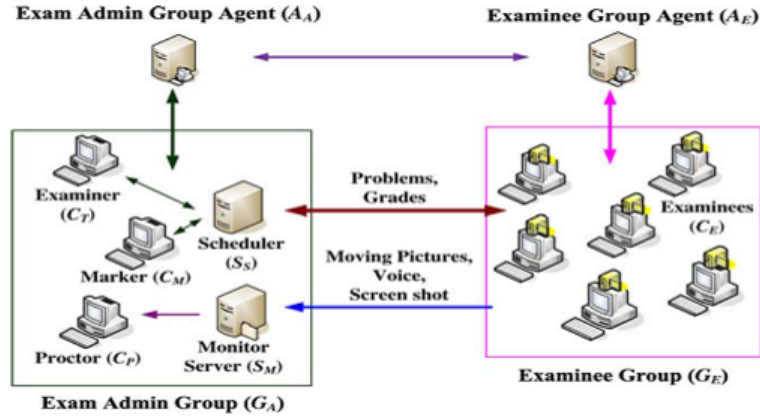


Figure 6: System architecture of SeCO_nE

The components of a SeCO_nE system can be explained as follows: This system uses a webcam to identify and authenticate the student. The photos taken during registration, and the current image from the webcam is verified to authenticate the student taking the exam. The verified data is saved for the exam and the photo can be used easily identify the student taking exam. The audio and video of the person taking the exam is continuously recorded. This audio and video data is then saved during the exam for later analysis. This is done to reduce cheating during the exam. Many screen shots of the student taking the online exam are taken and stored in parallel with the audio and the video. Audio and video captured makes it easier for the proctor to determine what exactly an examinee is doing on their computer. During the exam all computer ports are also disabled on the student's computer taking the online test except for the ports required for the online exam. This prevents the student from accessing the Internet and finding answers online or through popular search engines. The port that is open is then used to send student's

IP address of the student to the exam administrator.

This method places quite a few restrictions on the student's computer during the exam. The use of a webcam to record audio and video, along with blocking all the ports of a student's computer, is just not practical for an online test experience.

2.5.2 Triangulation

The tool in [23] is designed to prevent cheating when students take online exam on their laptops. When students use their laptops to take an exam, the risk of cheating increases and it becomes more important to monitor these exams. Using their laptop students can use many methods to cheat during an exam. The first half of the report explained possible ways a student can cheat online during their exam. The second half of the report discussed the various methods and solutions that can help in detecting and preventing online cheating. This report will now explain how directional antennas were used to measure the wireless signal strength and a methodology called "triangulation" in order to pin-point any student's computer. By measuring the signal strength and location of a student, small tests were performed to determine whether the system can detect any cheating within an exam hall.

As shown in Figure 7, students can seek help from an outside resource. The outside resource may be a wireless access point having open Internet access, or a friend or an expert that will help to find solutions for the questions on the examination.

As shown in Figure 8, students can communicate with each other within the examination hall. Students can be seated at some distance from one another. By communicating with each other they can discuss exam questions and collaborate on answering them.

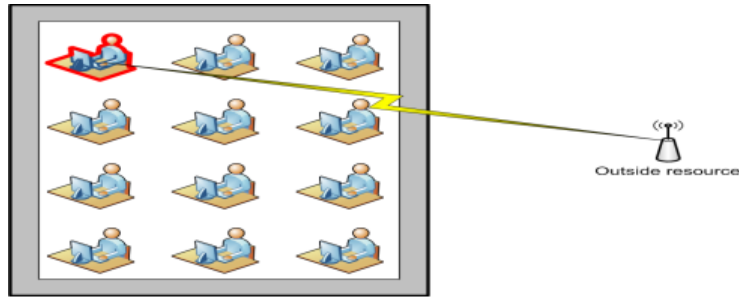


Figure 7: Students cheating by taking help from outside

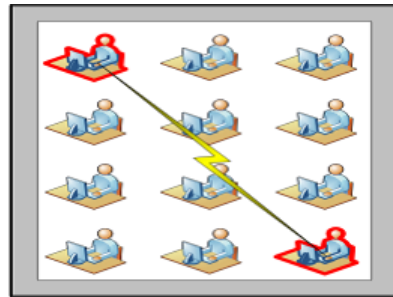


Figure 8: Students cheating being at a particular distance

As shown in Figure 9, students can also communicate with neighbouring students in the same exam hall. In a similar situation mentioned above, students can communicate with each other, but not with others outside the network to discuss exam questions.

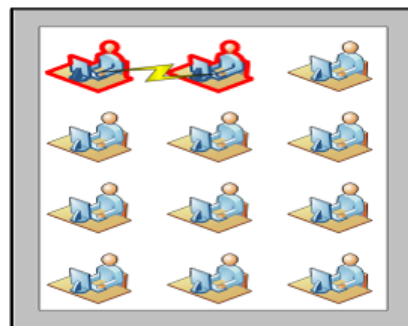


Figure 9: Student trying to cheat by communicating with neighbor student

All three types of attack were resolved by using Triangulation. This technology is used for detecting the location of a particular student by using basic trigonometry.

Different types of antennas were used to find the location of a wireless signal which was the devices used by students taking the exam. The antenna and an USB wireless card was used to measure the signal strength. Kismet on BackTrack Linux was used for capturing packets. The packets captured by Kismet were saved in a .pcapdump file. Wireshark was used to filter the captured packets based on a MAC address of a wireless network card and signal strength.

CHAPTER 3

Design

3.1 Approach

The goal of our project is to build an online test monitoring tool that students can take exams using their laptops. Students will be allowed to access only a few websites that are white-list. Websites, other than white-list sites will not be allowed. The monitoring tool should detect if any student tries to cheat by opening any website which is not on the white-list. Currently students can cheat by either opening blacklisted websites, which are not allowed, or they can connect to some other wireless network in order to open those websites. In either case, they would be opening a restricted website on their laptop.

The main approach for making the Online Test Monitoring tool is to notify the instructor when:

- a student has disconnected from the test network
- a student opens up a non-white listed web site
- a student switches between wireless networks
- storage of the IP/MAC address of all students connected to the test server is complete

The above points can be achieved by sniffing packets within the network, used by students taking the test. For this Kismet [5, 19, 21] was used to sniff all the packets. As discussed in Chapter 2, Kismet is a wireless packet sniffer that can

analyze network traffic used by students taking the test. Although “Kismet” can detect which IP/MAC address is connected to which access point, it still cannot detect which IP/MAC address belongs to an individual student. Thus, we need to obtain a method to identify their IP/MAC address. For capturing the name/IP/MAC address of a student, students were given a login page before the online test begins in order to type in their name and student ID then submit. As soon as they submit, the PHP code on the server will capture their name, IP, and MAC address then store this information within the database. We now have obtained a mapping of each student’s IP/MAC address. We now read the Kismet log files for submitted IP/MAC addresses in order to get information about the students who are taking the test. The student’s Mac address (from the database) is looked up in the “.netxml” file obtained from Kismet. This xml file has the list of all wireless clients connected to the network. The student’s IP address obtained from the database is matched with the IP address present in the logs (.pcapdump) which was obtained from Kismet. The .pcapdump is the raw dump of all the packets flowing through the network and needs further processing to extract useful information. For this feature we used another tool called Tshark to read the Kismet pcapdump file. Tshark is a command line version of Wireshark with all the same features. By converting the log from pcapdump to text, we obtained details regarding the activities of every IP/MAC address taking the test. Based on the activities of each IP address, the list of websites viewed by a student was extracted from the logs and further reports and notifications were generated.

3.2 Development Tools

3.2.1 Kismet

In our project Kismet is used to capture all the packets flowing within the network. Since Kismet remains completely passive while capturing packets, it was the

best choice for our project. Kismet identifies the network by passively collecting data packets and detecting hidden networks via data traffic as shown in Figure 10.

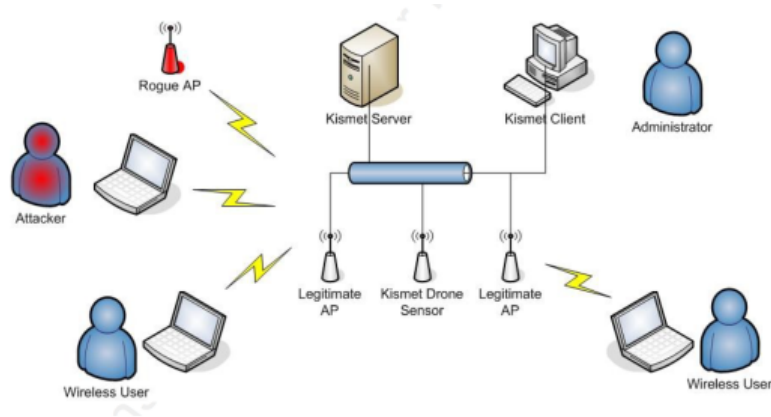


Figure 10: Kismet architecture

Kismet can also operate in a distributed architecture, can have a server called “kismet_server,” and a client called “kismet_client” that can be connected to individual servers. There can be many servers and clients running at the same time, configured to monitor different parts of the network.

Kismet can generate several types of log files such as: “.dump,” “.csv,” “.xml,” “.netxml,” “.nettxt” and “.pcapdump”. Using these log files one can gain quite a bit of information about the wireless network hardware as well as the software in use.

In summary, the features of Kismet include [1]:

- can scan wireless access points passively
- can detect hidden access points
- can detect “cloaked” access points
- also provides GPS support to geo-locate access points

- logs all information in the form of log files (.csv, .netxml, .nettext, .gps, and .pcapdump)
- raw packets flowing through the network are stored in .pcapdump file

The following procedure is used to run Kismet:

Step 1 - To start Kismet we can either click on the icon or type the command “sudo kismet” into the command prompt.

Step 2 - As shown in Figure 11 and 12 (below), a GUI will open in the terminal and will prompt the user to start the Kismet server. “Yes” and “Start” is pressed to start the program.

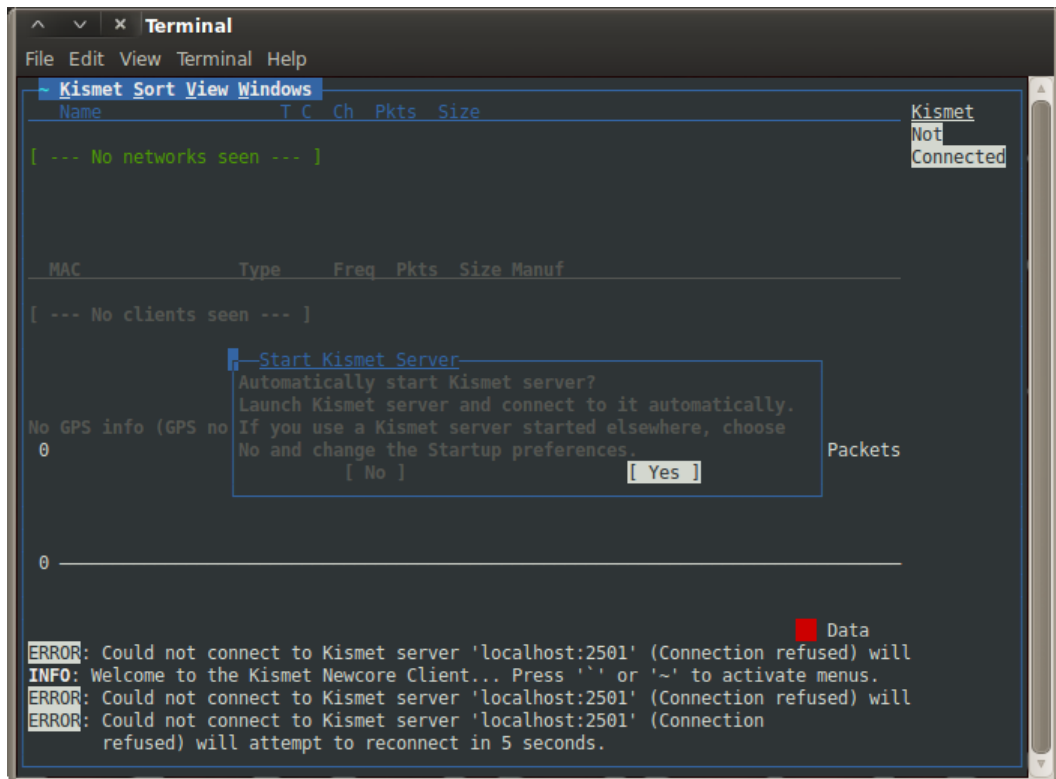


Figure 11: Kismet Startup GUI

Step 3 - As shown in Figure 13, Kismet will prompt the user that there are no

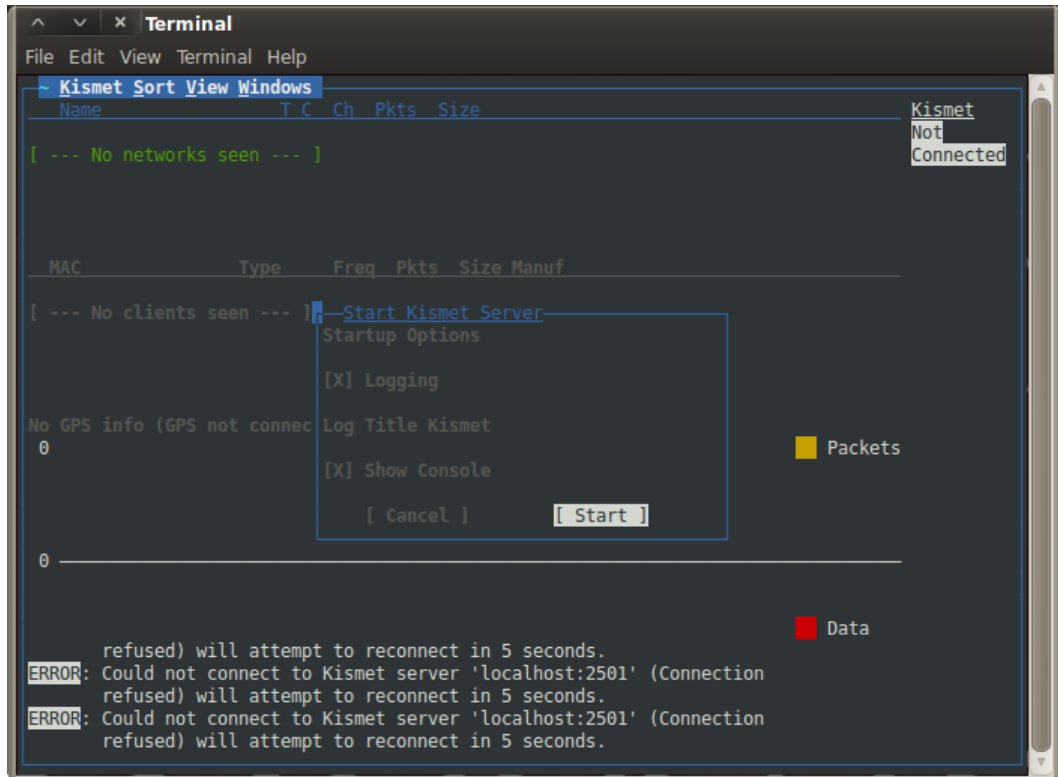


Figure 12: Kismet Startup GUI

sources defined on which Kismet will capture packets. After “Yes” is pressed user is prompted to “Add Sources” as shown in Figure 14. In our project, the wireless interface is used so the source will become “wlan0”. To configure Kismet on any other source we can use the command “iwconfig” to discover different network sources on that computer and then add that particular source.

The sources can also be defined in the Kismet configuration file “kismet.conf” as follows: “ncsource=wlan0” if the source is a wireless card.

Step 4 - Once the source is added, Kismet will show the list of all the visible nearby wireless network and start capturing packets as shown in the Figure 15. The list of networks will also contain any hidden networks and will appear as <Hidden SSID> in the list. By clicking on the name of the wireless network a table below

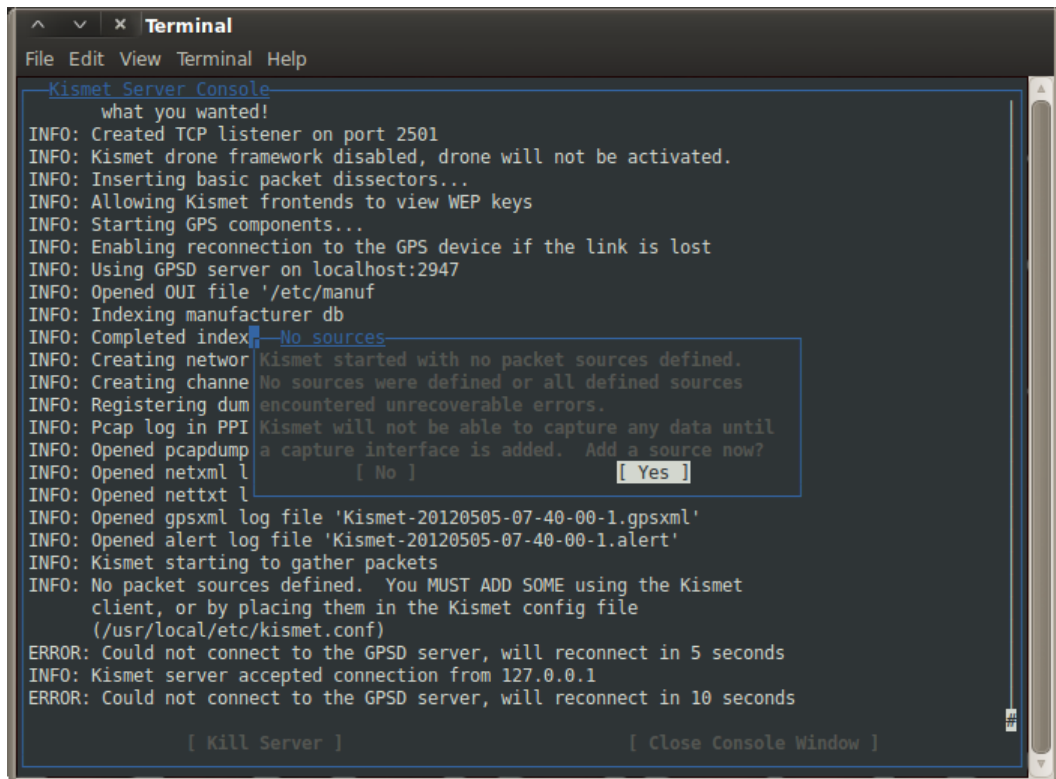


Figure 13: Kismet GUI for adding sources

the list will show the number of devices connected to that network. There is also a horizontal bar at the bottom of the Kismet UI that shows data packets passing through the network. This bar shows that Kismet is capturing all the packets from any nearby visible wireless networks.

When Kismet runs, it generates multiple log files one of which is as “.netxml” log file. This log file contains details about the network infrastructure and hardware. It consists of the following three important parts:

Part 1 Wireless card type information

This part of XML shows information about the computer’s wireless card which has the Kismet server running on it. As shown in Figure 16, wlan0 was the card interface and sjsu was the card name.

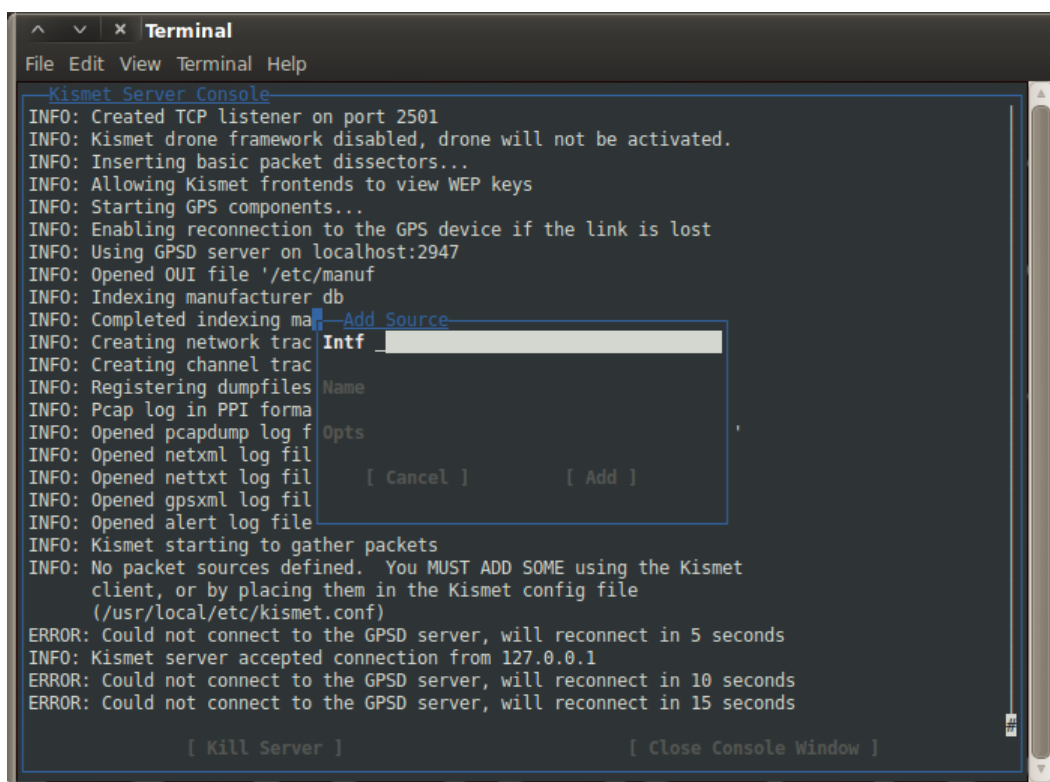


Figure 14: Kismet GUI for adding sources

Part 2 List of wireless networks

This part of XML gives information about the list of wireless networks visible to the Kismet server. As shown in Figure 17, the XML file also provides other useful information about available networks such as the BSSID, UUID, Channel, and Frequency.

Part 3 List of wireless clients connected

This part of an XML gives information about all the wireless clients connected to the network. There can be multiple wireless clients connected to a single wireless network; this XML file will give us a complete list of all clients connected to each access point. As shown in Figure 18, this XML file also shows us each client's MAC address. Using this MAC address, our tool will search for all the access logs from the

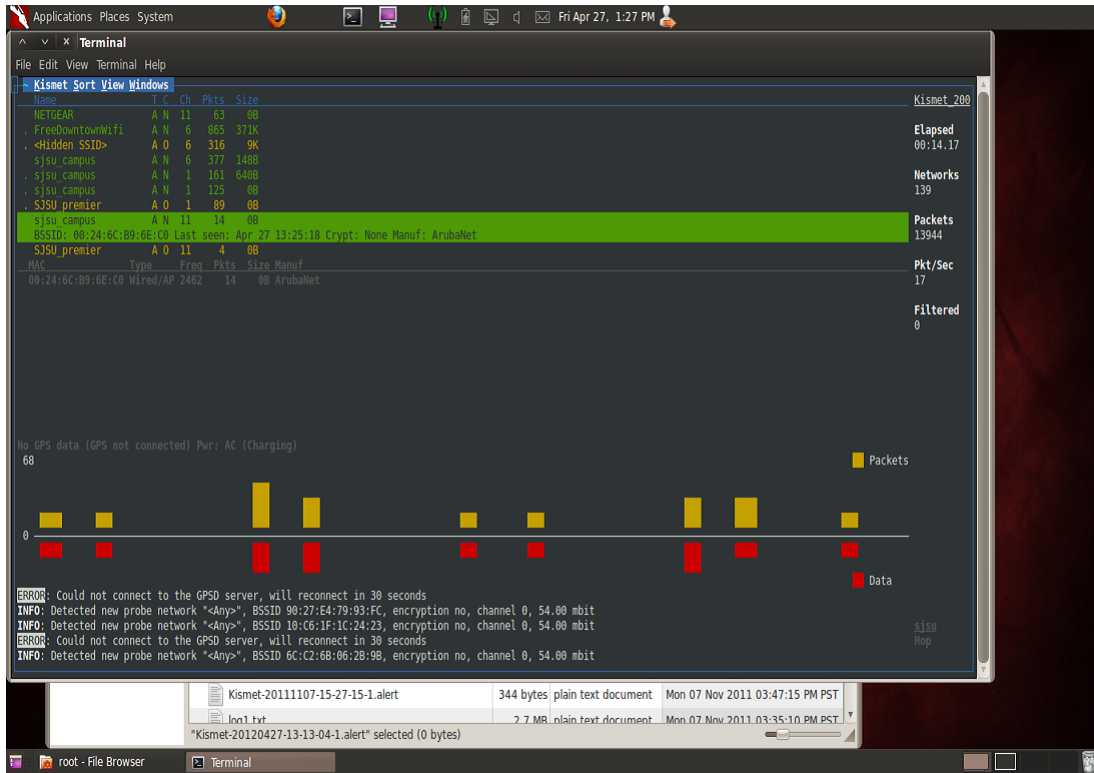


Figure 15: Kismet GUI showing packet capturing

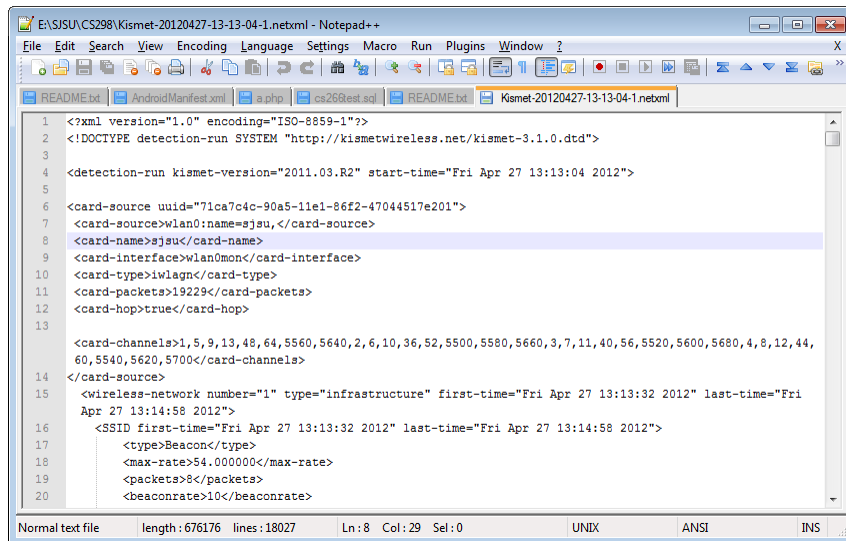
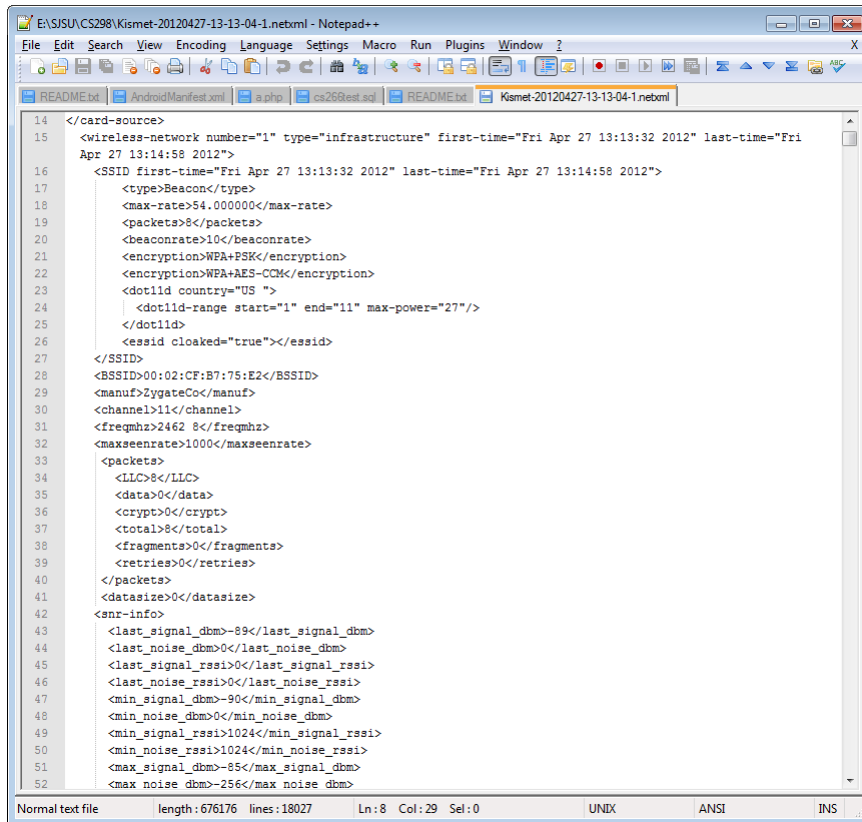


Figure 16: XML log file showing card type

packet dump log file.



```
14 </card-source>
15 <wireless-network number="1" type="infrastructure" first-time="Fri Apr 27 13:13:32 2012" last-time="Fri
16 Apr 27 13:14:58 2012">
17 <SSID first-time="Fri Apr 27 13:13:32 2012" last-time="Fri Apr 27 13:14:58 2012">
18 <type>Beacon</type>
19 <max-rate>54.000000</max-rate>
20 <packets>8</packets>
21 <beaconrate>10</beaconrate>
22 <encryption>WPA+PSK</encryption>
23 <encryption>WPA+AES-CCM</encryption>
24 <dot11d country="US ">
25 | <dot11d-range start="1" end="11" max-power="27"/>
26 </dot11d>
27 <ssid cloaked="true"></ssid>
28 </SSID>
29 <BSSID>00:02:CF:B7:75:E2</BSSID>
30 <manuf>ZygateCo</manuf>
31 <channel>11</channel>
32 <freqmhz>2462 8</freqmhz>
33 <maxseenrate>1000</maxseenrate>
34 <packets>
35 <LLC>8</LLC>
36 <data>0</data>
37 <crypt>0</crypt>
38 <total>8</total>
39 <fragments>0</fragments>
40 <retries>0</retries>
41 </packets>
42 <dataseize>0</dataseize>
43 <snr-info>
44 <last_signal_dbm>-89</last_signal_dbm>
45 <last_noise_dbm>0</last_noise_dbm>
46 <last_signal_rssi>0</last_signal_rssi>
47 <last_noise_rssi>0</last_noise_rssi>
48 <min_signal_dbm>-90</min_signal_dbm>
49 <min_noise_dbm>0</min_noise_dbm>
50 <min_signal_rssi>1024</min_signal_rssi>
51 <min_noise_rssi>1024</min_noise_rssi>
52 <max_signal_dbm>-85</max_signal_dbm>
53 <max_noise_dbm>-256</max_noise_dbm>
```

Figure 17: XML log file showing network information

3.2.2 Tshark

The command line version of Wireshark is known as Tshark [38]. Kismet, while capturing packets, generates a “.pcapdump” file. This .pcapdump file contains raw packets captured from the network. The information present in .pcapdump file is not in a readable format as it’s a raw dump of all the packets flowing through the network. When this file is opened using Wireshark, the packet details can be seen, however for an automated solution this will not work as it requires manual scanning of the logs by Wireshark. Thus tshark was used to filter the .pcapdump file and show packet details.

Tshark was configured to filter only the HTTP and TCP protocols by changing


```
63 </seen-card>
64 <wireless-client number="1" type="fromds" first-time="Fri Apr 27 13:14:58 2012" last-time="Fri Apr 27
13:14:58 2012">
65 <client-mac>00:02:CF:B7:75:E2</client-mac>
66 <client-manuf>ZygateCo</client-manuf>
67 <channel>11</channel>
68 <freqmhz>2462 8</freqmhz>
69 <maxseenrate>1000</maxseenrate>
70 <packets>
71 <LLC>8</LLC>
72 <data>0</data>
73 <crypt>0</crypt>
74 <total>8</total>
75 <fragments>0</fragments>
76 <retries>0</retries>
77 </packets>
78 <datasize>0</datasize>
79 <snr-info>
80 <last_signal_dbm>-89</last_signal_dbm>
81 <last_noise_dbm>0</last_noise_dbm>
82 <last_signal_rssi>0</last_signal_rssi>
83 <last_noise_rssi>0</last_noise_rssi>
84 <min_signal_dbm>-90</min_signal_dbm>
85 <min_noise_dbm>0</min_noise_dbm>
86 <min_signal_rssi>1024</min_signal_rssi>
87 <min_noise_rssi>1024</min_noise_rssi>
88 <max_signal_dbm>-85</max_signal_dbm>
89 <max_noise_dbm>-256</max_noise_dbm>
90 <max_signal_rssi>0</max_signal_rssi>
91 <max_noise_rssi>0</max_noise_rssi>
92 </snr-info>
93 </seen-card>
94 <seen-uuid>71ca7c4c-90a5-11e1-86f2-47044517e201</seen-uuid>
95 <seen-time>Fri Apr 27 13:14:58 2012</seen-time>
96 <seen-packets>8</seen-packets>
97 </seen-card>
98 </wireless-client>
99 </wireless-network>
100 <wireless-network number="2" type="infrastructure" first-time="Fri Apr 27 13:13:25 2012" last-time="Fri
Apr 27 13:33:00 2012">
```

Figure 18: XML log file showing clients information

the configurations in Wireshark. After filtering, the file is saved as a .txt file. The following command is used to convert .pcapdump file into .txt file.

```
$ tshark -r Kismet.pcapdump > log.txt
```

The above command will take the Kismet.pcapdump file and convert it to a normal readable text file. This text file is then parsed to extract the access logs of the students giving the tests.

3.2.3 XAMPP Server

The online test has to be hosted on a web server. The web server should also have an active connection with a database server in order to store student's IP/MAC

addresses, as well as online test questions and answers. XAMPP [9, 40] was chosen for this purpose as it is a complete bundle of Apache web server and MySQL database server. XAMPP also has support of PHP, perl, and Python so the server side scripting can be easily done on this server. XAMPP is also very easy to install and configure on any machine as its open source; therefore XAMPP was the best choice for our project. There are many tools that come with XAMPP server that aids in rapid software development. phpMyAdmin is one such tool that gives an easy web interface to a MySQL database. The database creations and other operations are seamless.

Thus, an online test was hosted on this web server and the database was also created on the same machine.

3.2.4 Eclipse for Android application

A mobile application was also designed for Android to help instructors gain instant notification of any unwanted activity on their Android Smart Phone [27]. The main idea behind designing the Android application was to help instructors roam in the classroom and still be able to get student's notification (alerts) about any unwanted activity. Instructors could stay at a computer running Kismet to watch for notification alerts, however having an application on their Smart Phone means an instructor can remotely monitor the exam. They can leave the examination hall and still be able to monitor the exam.

Eclipse was used to build an Android application that can obtain notifications [17] from the Kismet server. The application was designed in such a way that it can help instructors remotely control an online test. The various features of the application are as follows:

- Any website can be added within a white-list or a blacklist;
- The list of websites in the white-list or a blacklist can be viewed;
- The list of students taking an exam can be seen;
- The details of each student can be retrieved;
- And Kismet server logs can also be seen.

Proper notification will be generated on the application if there is suspicious activity.

3.2.5 Python

The logs, generated by Kismet are usually large in size and converting them into text that can be parsed and processed is challenging. To accomplish this, Python [29] was chosen as it is a scripting language and has an automatic memory management that can help in parsing large sized files. Python is open source and is also easy to code in Python.

3.3 Software Design

In this section the design of the online test and the design of the autonomous Kismet .pcapdump parser will be discussed.

3.3.1 Online Test Design

The online test was designed in such a way that the student's details, along with their IP/MAC address, are stored within the database for use with a Kismet log parser. The online test was designed using AMP (Apache MySQL PHP). PHP was

used since it's easy to write server side code and its easily configured with Apache. The database was created in MySQL. The following are the key factors for the online test design:

- The site should let the student Login/Register with their name and their unique student ID;
- The site should capture the MAC/IP address of every student;
- The IP/MAC address should also be captured as soon as the student begins taking the test;
- The server should allow long polling from all the students taking the test;
- And the database should be accessible by the machine running Kismet.

Keeping the above things in mind, the online test site was designed with a front page as shown in Figure 19.

This front-end was designed using HTML/JavaScript/JQuery, along with PHP as the server side scripting language. The database used was MySQL. XAMPP was used to host this online test therefore the database and the web server were on the same machine.

Students have to first register on the website. As soon as the student registers on the site, their IP/MAC address was stored in the database and also a session variable is set with these values. From this point forward whenever a student logs-out and logs-in again to the test, their IP address and MAC address are captured and stored within the database.

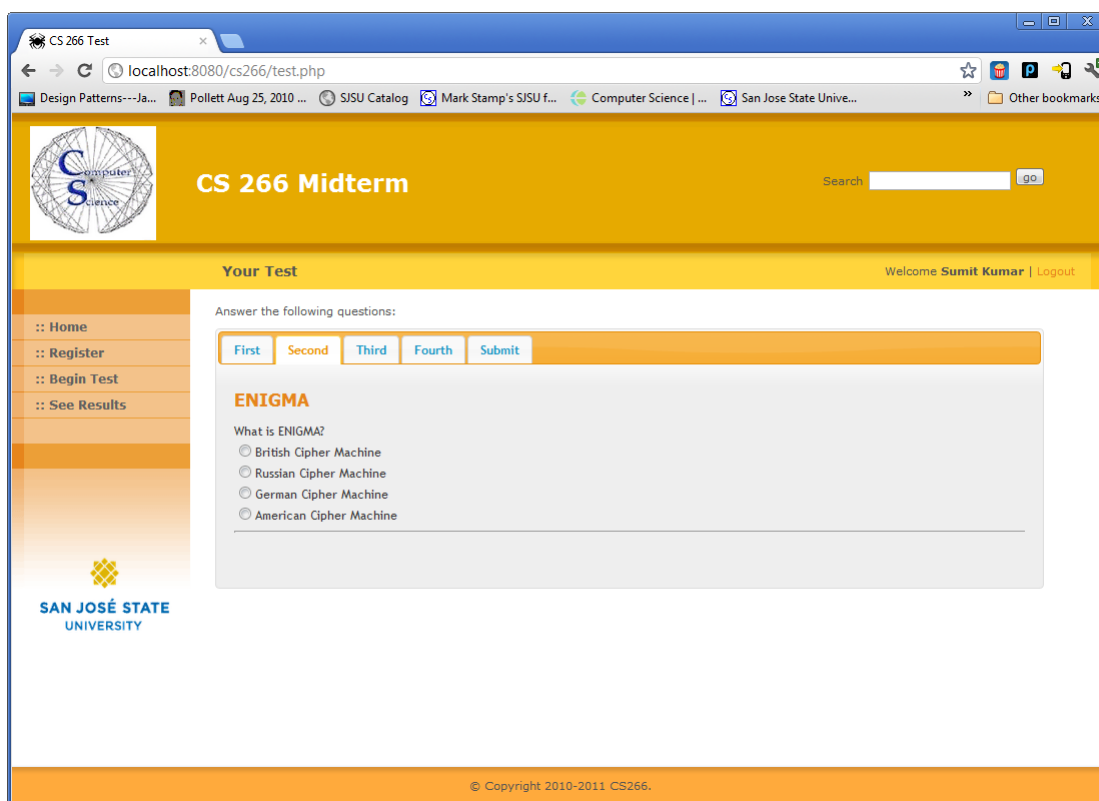


Figure 19: Home page of the online test

All web pages within the online test contain a code for an Ajax call every 5-seconds to a specific server-side PHP code that always compares the student's IP/-MAC address stored in database with their present IP/MAC address set during the session. So if the addresses do not match, then the student might have changed the network or have applied MAC spoofing, and they can be flagged.

The server also maintains a list of all the current IP/MACs that have started the test and are constantly polling the server. If the server does not receive a poll from any of the IP addresses on the list, it flags that IP as disconnected. Further checks on that disconnected IP are done to discover if that IP has been disconnected genuinely or the student has disconnected. As soon as an IP is flagged, a separate code on the Kismet server will run and will try to scan the full network to determine the MAC

address associated with this IP. The MAC address is then tracked for any suspicious activity.

3.3.2 Kismet .pcapdump Parser

To read the .pcapdump file tshark was used. With tshark the text file generated contains the details of the packet's source and destination IP/MAC. This should be done continuously until the test is over. Therefore, a Python code was written which does the following (in order) after the test has started:

1. Obtain the list of student's IP/MAC address taking the exam;
2. Retrieves the white-listed IP/MAC addresses from the database;
3. Converts a .pcapdump file to a .txt file by running a shell command;
4. Reads the text file generated line-by-line;
5. And for each line, checks that it contains student's IP/MAC taking the test
6. If found, then checks which student's IP/MAC is the IP/MAC is pointing to
7. If the student IP/MAC points to another IP/MAC that is not in the white-list, then that student's IP/MAC address is stored in a separate table within the database
8. Goto step - 3 and repeat

3.4 System Design

When Kismet runs, it makes the wireless card operate in a monitoring mode. This means that the network card can only monitor the activities within the wireless

network. It will not be able to connect and exchange data with any other computer which also means that no other computer can reach the computer running Kismet.

The approach was to have the Kismet server running on a machine and a Python script continuously parsing the log file and querying the data from the database server. In order to query the database, the machine running Kismet should be able to connect to the remote machine. However this is not possible using a wireless network card as it is in a monitoring mode. To accomplish this P2P Ethernet connection was made between these two machines: one machine running Kismet and one machine running the XAMPP server, which is also the database server as shown in Figure 20.

As shown in Figure 20, we have a web server that hosts the online test and a database server that stores student information, as well as the test details. Students taking the test, while in the examination hall, can use their laptops having any operating system on it. The students will need a web browser that can open the online test. Once online, students will connect to an access point where the web and database servers are also connected. The access point has a connection to the Internet through a firewall whereas; the Kismet server only has a connection to the database server. Once the test begins, the students will connect to the web server and the database server will log their IP/MAC addresses. The Kismet server will capture all the packets and look for all the IP/MAC addresses retrieved from the database server. It will also check if individual IP addresses from the exam hall are using any website other than the authorized white listed sites. The instructor will have a mobile device that can receive notifications from the Kismet server. If the Kismet server sees any suspicious activity, then it will generate a notification and send that as a push notification to the Android application on the instructor's Android device.

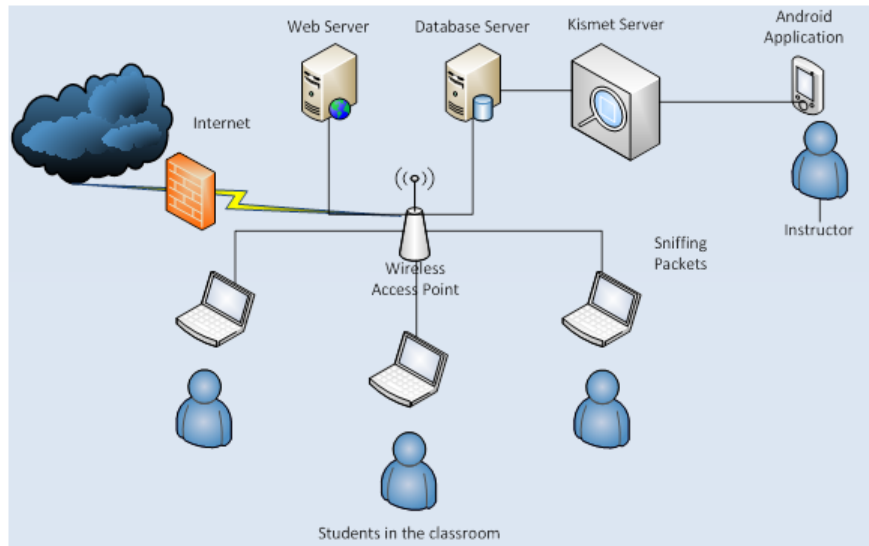


Figure 20: System Design

3.5 Database Design

The online test has to store all student's information. For this purpose a database was chosen, as the data can be stored persistently and can be easily retrieved if there are any system failures. For our project MySQL was chosen as the database since it comes bundled with the XAMPP server. To store logically separated information, different tables were used. Figure 21 illustrates the various tables within our databases and their relations.

Table Description:

student_info: This table stores student's information for those who are registered for the online test.

student_ip_mac: This table stores current student's IP/MAC addresses. So if a student has registered with a different machine and logged-in with a different machine, the new IP/MAC addresses of the different machine will be stored.

ip_accessed: This table stores all the remote IPs accessed by all student IPs

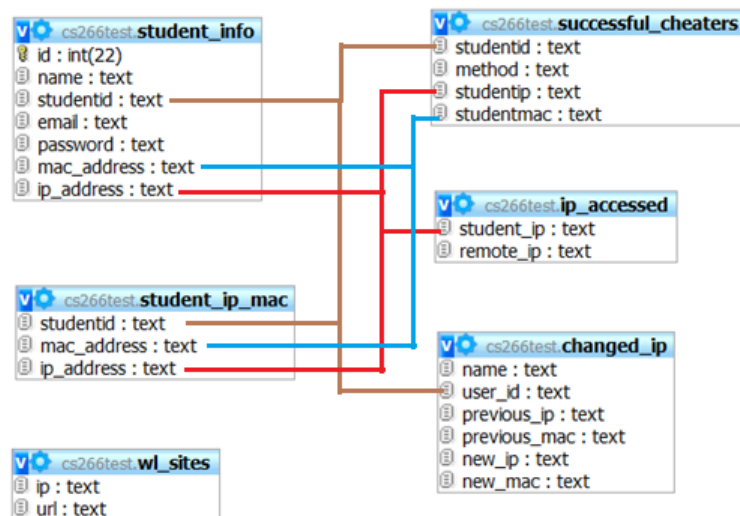


Figure 21: Database Design

within the database.

successful_cheaters: This table is created to store the method of cheating by any student if they remain undetected during the test but were successful at cheating.

changed_ip: This table is created to store the IP/MAC address of any student if they were caught while changing their IP/MAC address.

wl_sites: This table stores the list of all the white-listed sites with their IP

The above table names were chosen keeping in mind their use. The information stored within the table can be easily assessed by looking at the name of the table. The Kismet server retrieves the data from the database and can easily query these tables as it only reads data from selected tables. If more information is needed, then it can make further queries to the other tables that have a relation between themselves, thus retrieving additional information.

3.6 Mobile Application Design

To aid the instructors in obtaining quick notifications during the exam, a mobile application can be useful. The instructor can physically roam in the exam hall and still keep an eye on students taking the exam. If the system generates an alert, the instructor can immediately see it. The instructor can also remotely configure the system by adding and deleting websites from any given list. To accomplish all the above, an Android application was designed with the following main features:

- Websites can be added on the go during the examination;
- A student's access data can be viewed;
- Full system logs can also be viewed;
- And notification is received if there is any suspicious activity.

Android was chosen for developing our mobile application, as it is open source and anyone can create, test, and run the application on any operating system. Android is also one of the largest selling mobile operating systems. Following the above design goals, an Android application was developed as shown in Figures 22 through 25.

The above image shows the main screen of our application which has all the functionalities as an image button. By clicking on "Student Details" an instructor can retrieve student details, including their IP addresses and MAC address. In order to view the complete system log, the instructor clicks on "System Logs". At any time, instructor can get help by clicking on the "Get Help" button; this will help the instructor to learn and configure the application. By clicking on the "Manage WebSites" button, it brings up another view as shown in Figure 23 (i). This view

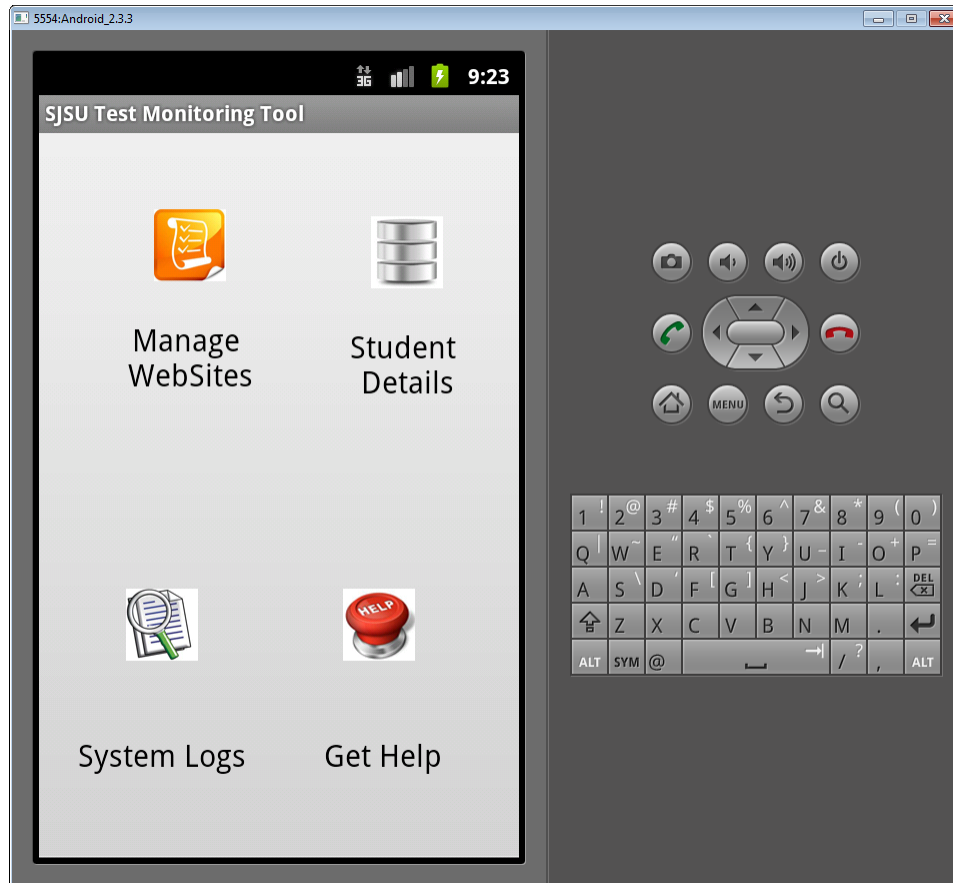
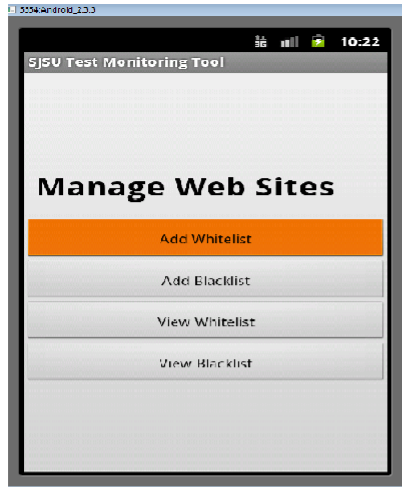


Figure 22: Android Application main screen

provides features needed in order to manage required exam such as, adding websites to the white-list and blacklist, and viewing individual lists.

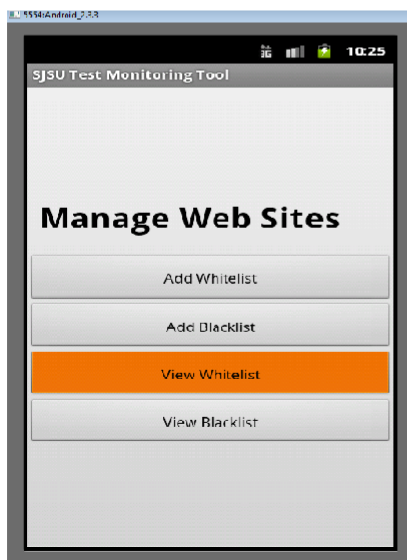
Figure 23 (ii) displays the view to enter a website name. This view has an input field where an instructor can enter the URL of the website and add it in the white-list. By clicking the submit button, entries will be saved in the local storage as well as on the Kismet server. By clicking on the “Back” button the user can go back to Figure 23 (i). If the user then clicks on “View Blacklisted Sites” they will open up the view below, as shown in Figure 24 (i).



(i)



(ii)



(iii)



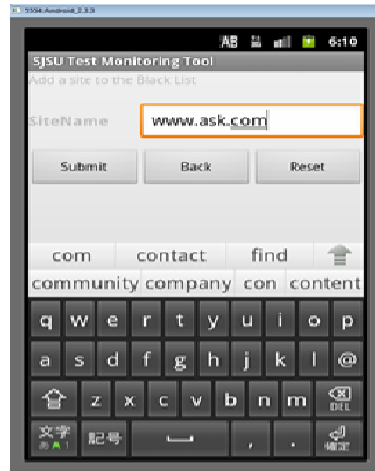
(iv)

Figure 23: Adding a website to the white-list

Figure 24 (iv), shows the blacklisted sites that have been added in the Kismet server database. These are sites, which if accessed by any student, will be flagged immediately. Going back to the view in Figure 22 and clicking “Student Details” the user gets the view as shown in Figure 25 (i) below. By clicking on any student name we can see their details as shown in Figure 25 (ii) below.



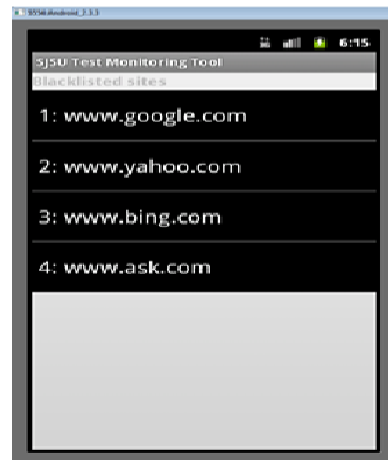
(i)



(ii)



(iii)



(iv)

Figure 24: Adding a website to the black-list

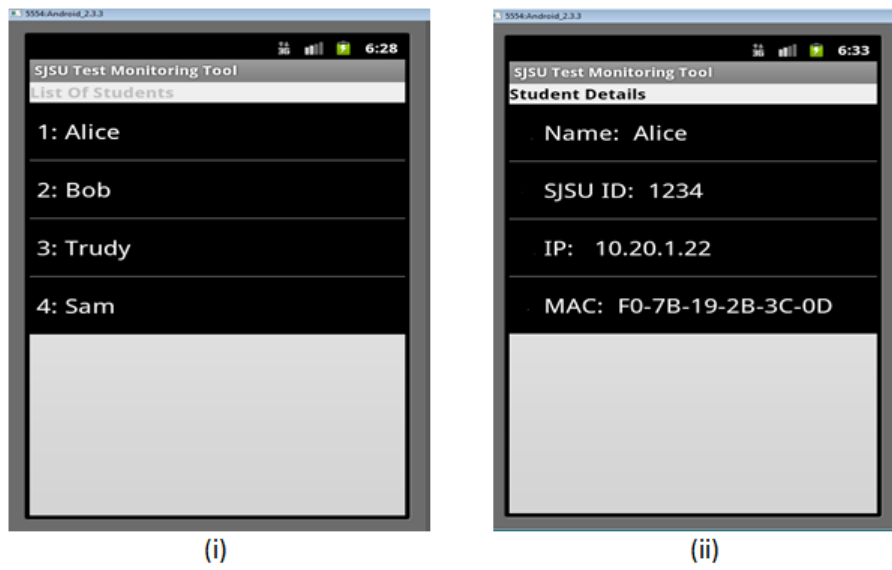


Figure 25: Student Details

CHAPTER 4

Implementation and Results

This section discusses the implementation of the application developed and experiments completed to test it. Various experiments were performed at different locations in order to test the system.

4.1 Experiment 1

As shown in Figure 26, BackTrack Linux 5 was installed on one laptop. Kismet, installed with BackTrack Linux, was configured for wlan0 and was started. A second laptop was configured to run XAMPP and an online test was hosted. The Kismet server was also connected to XAMPP server via an Ethernet cable in order to access the database. Once the servers were set-up on the laptops, they both were connected to a wireless router which was open and had no password or encryption. Since the router was unsecured, anyone can connect to it and the packets exchanged via this router can be read by anyone. After the three devices were set-up they were also connected to the router. Once these devices were ready, the test was started through the browser. The first page of the test required the user to register; therefore an unique user was registered. As soon as each user was registered on the device, their IP and MAC addresses were stored within the database. The test was started and the Kismet server was already capturing packets. The Python script was then started in order to convert and parse the log from Kismet. This Python script retrieved the IPs for all three devices from the database and started monitoring them by identifying their activities from the log. A special page “cheaters.php” was designed that displays the name, IP address, and the non-white-list IP accessed by each device. This page

refreshes every 5 seconds and shows if it has identified anyone cheating.

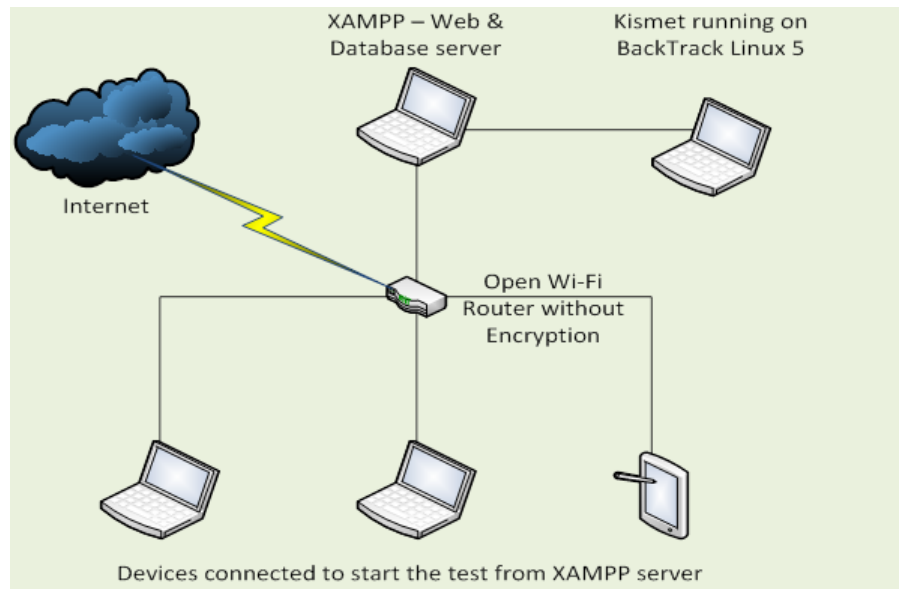


Figure 26: Experiment 1 System Architecture

After the devices were set-up and connected, and students were given access to the test, they were not allowed to open websites other than those on the white-list.

The following are the activities that were performed during the experiment:

- **Activity - 1:** The test was started and no other site was opened from the computer

Expected Result: The cheaters.php, which shows the list of students who are caught cheating, should remain empty

Actual Results: The cheaters.php was empty

- **Activity - 2:** Only the websites listed in the white-list were opened on the devices.

Expected Result: The cheaters.php page should not show any username or IP.

Actual Results: No name or IP was displayed on the page.

- **Activity - 3:** Google was opened from one of the connected PC

Expected Result: The name of the user accessing Google should display in the cheaters.php

Actual Results: The name of the user was instantly displayed

- **Activity - 4:** One PC was disconnected from the test network and was connected to some other Wi-Fi

Expected Result: The cheaters.php should display the user name along with the MAC addresses of the wireless networks

Actual Results: The MAC address of the two networks was displayed against the name of the user

Results and explanation: By looking at our results, we can determine that there are three major ways a student can use to cheat and each was successfully detected by the application.

4.2 Experiment 2

A second experiment was performed in the college with students enrolled in CS266 course (Information Security). The overall architecture of the system was the same as shown in Figure 27. The wireless router used was a special router and was not the same as the college access point. The steps for configuring the servers were again the same and all CS266 students were allowed to join the assigned network. Before the students went online, Kismet had already been started. Once students registered and started the test, Kismet was already capturing student packets. During the test,

students were only allowed to visit 2 sites www.sjsu.edu and www.cs.sjsu.edu. After the test was started they were instructed to cheat. The Python code was run to parse and read the .pcapdump file and extract information regarding student activities.

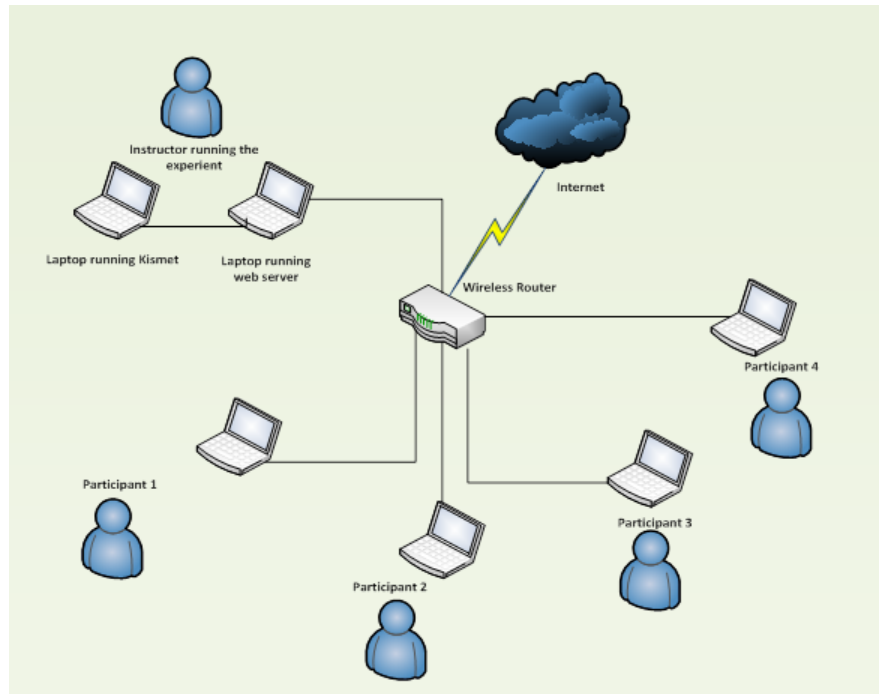


Figure 27: Experiment 2 System Architecture

The following are the activities that were performed during the experiment:

- **Activity - 1:** The test was started and students were told to cheat without getting caught

Expected Result: The `cheaters.php`, which shows the list of students who are caught cheating, should show only those students who tried to access any site other than the white listed sites

Actual Results: The `cheaters.php` showed the list of all the students opening multiple sites

- **Activity - 2:** The test was restarted and the students were asked to re-register. This time they all were told not to cheat and only one student should open other sites

Expected Result: Only that one student's name should be displayed on the cheaters.php

Actual Results: The name of many students showed on the cheaters.php including the one student opening other sites

Results and explanation: There were many false alarms showing that a student was cheating, but were actually not. The main reason for the false alarm was due to the software installed on student laptops. The software might be sending requests for an update to their server of which students were unaware. There were also many plugins embedded within web browsers, such as Chrome and Firefox. These plugins may be sending packets to a remote server and since the white list does not have those IPs, students were being blacklisted.

4.3 Experiment 3

By reviewing our results in the above two experiments, we can clearly see that our project was able to handle large amount of traffic within the college. It was also able to retrieve all students connected on a test server that was hosting the online test. One major problem, that of scalability of our system, was tested and our system proved to be fast and better as compared to our previous attempt [32].

The main problem that remains with our system is the amount of false positives. Due to various software and operating systems sending anonymous packets without user notification, it becomes important for our system to immediately filter false

positives. To accomplish this, our Python script was modified to look for traffic on port number 80 that is the Hypertext Transfer Protocol (HTTP) in the text file generated from the .pcapdump file. The assumption was that software updates, OS updates, and/or anti-virus or browser plugin updates will not get to port 80 of the server. For example, the Chrome plug-in updates will not go to `www.google.com:80` but will go to some other port on the Google server. Similarly the OS updates will not go to `www.microsoft.com:80` but may be routed to some other port on the Microsoft server. Any student who is trying to cheat using the Internet will surely visit the web server from their browser. They will go to port 80 of the server so we can safely neglect any traffic going on other ports. There may be cases that the webserver is not configured on the default port but on port 8080. So our Python script was modified further to check for these non-standard web server ports. All other traffic was still logged for each student and can be analyzed later by the instructor.

Experiments conducted after these enhancements showed better results as many of the false positives did not occur this time.

In our third experiment we conducted a test in CS165/265 class (Cryptography and Computer Security) with 25 students. We further enhanced our system architecture as well. A web server (`cs17.cs.sjsu.edu`) was configured inside our college campus to host the online test. This web server was not accessible outside the college network. This eliminated the need of having two laptops in the classroom for our monitoring tool. The students connect to the college wireless network and take the test by opening the link `cs17.cs.sjsu.edu` in their web browser. The laptop running Kismet was configured with two wireless adapters, one for sniffing packets and the other to connect to the college network to access the database on the server `cs17.cs.sjsu.edu` as shown in Figure 28.

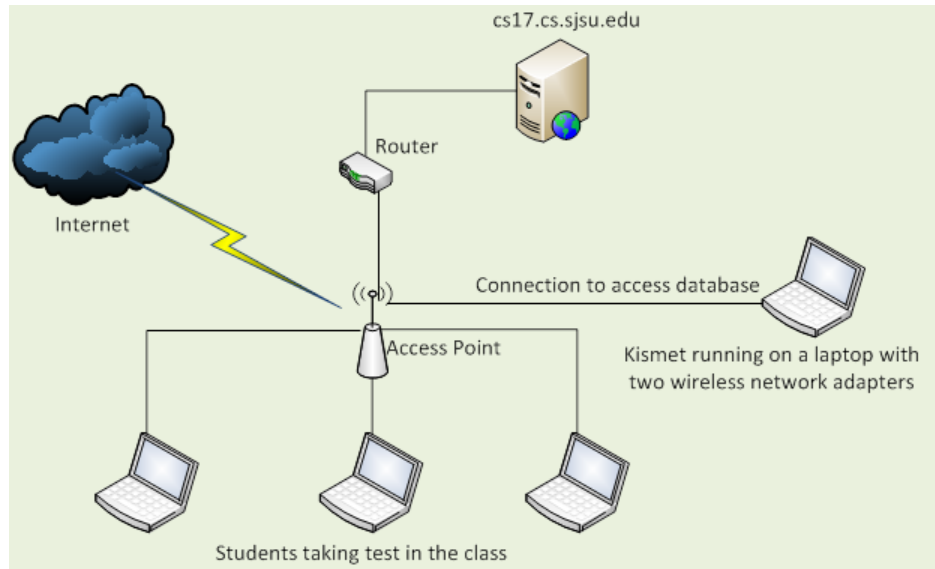


Figure 28: Experiment 3 System Architecture

Results and explanation: Once the test was complete, the log from Kismet was analyzed along with the database entries for all the students. It was observed that all the students had the same IP address in the database as shown in Figure 29. Due to this the packets captured by Kismet cannot be classified. Since the packets cannot be classified, we cannot determine the activities of students in the exam.

The reason for getting the same IP address was due to the fact that all the access points in the building are connected to a router and this router is natting the student's IP address to the test server cs17.cs.sjsu.edu. So our test server will always get the IP address of the router as the student's IP. So this experiment was not completely successful as it was not able to tell us about the student activities although we had the packet details.

id	userName	ip_address
3	user1	130.65.11.72
5	user2	130.65.11.72
6	user3	130.65.11.72
7	user4	130.65.11.72
9	user5	130.65.11.72
10	user6	130.65.11.72
11	user7	130.65.11.72
12	user8	130.65.11.72
13	user9	130.65.11.72
14	user10	130.65.11.72
15	user11	130.65.11.72
16	user12	130.65.11.72
17	user13	130.65.11.72
18	user14	130.65.11.72
19	user15	130.65.11.72
20	user16	130.65.11.72
21	user17	130.65.11.72
22	user18	130.65.11.72
23	user19	130.65.11.72
24	user20	130.65.11.72
25	user21	130.65.11.72
26	user22	130.65.11.72
27	user23	130.65.11.72
28	user24	130.65.11.72
29	user25	130.65.11.72

Figure 29: Database entries for all the students

4.4 Experiment 4

With the same setup as shown in Figure 28, another experiment was conducted with the students enrolled in the course CS46B (Introduction to Data Structures). There were 83 students who were taking the test and the sniffer was capturing the packets in the class room. This time the students were also asked to submit their IP address at the end of the test. Their IP addresses were then manually entered in the database and the Python script was run to convert the Kismet packet dump to parsable text file. A PHP web page was created to extract meaningful data from the text file by searching student IP address.

Results and explanation: The PHP web page showed the name of all the students who took the test and the IP addresses accessed by their laptop during the exam. After analyzing the logs we can see that the number of false positives due to the software and Operating System updates went down drastically. But there were still quite a bit of false positives. The reason was unclear but it might be because

of some software updating over HTTP. We could also see a lot of GET request with safebrowsing in it. This might be due to the Firefox plugin sending updates over the web to check for malicious websites. The IP addresses accessed by the students were also resolved to find their hostname to get a more details about the access. Thus it was clear from the results that most of the activities where the student tried to cheat were detected successfully as shown in Figures 30 through 36.

The top activities that were flagged are as follows:

- **Student 12, 34, 47** accessing Piazza (Ask Answer Explore Whenever) website. This website is a place where student can come together to ask, answer and explore questions.

Student No.12 - 10.185.195.212 -> 184.73.248.186

Student No.34 - 10.185.193.89 -> 184.73.248.186

Student No.47 - 10.185.205.119 -> 184.73.248.186

- **Student 18, 39** accessing Java API docs from an IP whose domain name couldn't be looked up.

Student No.18 - 10.185.209.79 -> 198.189.255.201 as shown in Figure 32

Student No.39 - 10.185.200.69 -> 198.189.255.224 as shown in Figure 34

Student No.50 - 10.185.210.47 -> 198.189.255.201 as shown in Figure 36

- **Student 26, 33** using googletalk plugin to chat in the exam.

Student No.26 - 10.185.201.236 -> 173.194.25.57 as shown in Figure 32

Student No.26 - 10.185.201.236 -> 173.194.25.70 as shown in Figure 32

Student No.33 - 10.185.203.245 -> 74.125.224.78 as shown in Figure 33

- **Student 12** accessing www.facebook.com

Student No.12 - 10.185.195.212 -> 69.171.234.37 as shown in Figure 31

- **Student 8** accessing www.oxytube.com

Student No.8 - 10.185.206.17 -> 50.116.55.164 as shown in Figure 30

This access may not be an attempt to cheat as there are browser plug-ins for Firefox which might be sending auto-update packets to the server. Just like we have lots of access to 74.125.224.72 (www.google.com) for safebrowsing as shown in Figure 30 and 34

Some activities were not detected by our tool as we were only sniffing the HTTP traffic. There were couple of students who submitted identical files as their answers and our tool was not able to detect it. This shows that they might have communicated through some channel other than HTTP. This means that our tool needs to be modified for tracking the traffic other than the HTTP as well.

```

Student No.8 - 10.185.206.17 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/Sum.java - HTTP/1.1
Student No.8 - 10.185.206.17 -> 74.125.224.98 - HTTP - GET -
/safebrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAEY6ZYFIJCXBSoidksBAP___wcyBmlLAQD_Hw -
HTTP/1.1
Student No.8 - 10.185.206.17 -> 74.125.224.98 - HTTP - GET -
/safebrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAAy8egEIMDpBCoNfzQBAP_____AzIGcTQBA
P8_ - HTTP/1.1
Student No.8 - 10.185.206.17 -> 74.125.224.98 - HTTP - GET -
/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhcABGNGBiCgjQYqBpSGAQD_HzINUYYBAP_____B
w - HTTP/1.1
Student No.8 - 10.185.206.17 -> 74.125.224.98 - HTTP - GET -
/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhcAAGiv8DCCU_AwqBQ4-AwB_MgULPgMABw -
HTTP/1.1
Student No.8 - 10.185.206.17 -> 50.116.55.164 - HTTP - GET -
/ext/youtube_downloader_update.xml?x=id%3Dbaghcaokjpifgfdiobkomaaklphhg%26v%3D11.0%26uc
- HTTP/1.1

```

Figure 30: Results of Experiment 4

Student No.12 - 10.185.195.212 -> 69.171.234.37 - HTTP - GET -
 /ajax/hovercard/user.php?id=100000689465501&__a=1&endpoint=%2Fajax%2Fhovercard%2Fuser.php%3Fid%3D100000689465501&__user=100001116249104 - HTTP/1.1
Student No.12 - 10.185.195.212 -> 198.189.255.216 - HTTP - GET - /hprofile-ak-
 snc4/161396_100000689465501_93803835_n.jpg - HTTP/1.1
Student No.12 - 10.185.195.212 -> 69.171.234.37 - HTTP - GET -
 /ajax/pagelet/generic.php/PhotoViewerInitPagelet?__a=1&ajaxpipe=1&ajaxpipe_token=AXI111_EKz7ApSqV&data=%7B%22fbid%22%3A%22240501232717414%22%2C%22set%22%3A%22a.212223438878527.33964.212208148880056%22%2C%22type%22%3A%221%22%2C%22size%22%3A%22500%2C500%22%2C%22theater%22%3A%22null%7D&__user=100001116249104&__adt=6 - HTTP/1.1
Student No.12 - 10.185.195.212 -> 69.171.227.72 - HTTP - GET -
 /pull?channel=p_100001116249104&seq=13&partition=1&clientid=3d1c1e8e&cb=2uxj&idle=32&state=active - HTTP/1.1
Student No.12 - 10.185.195.212 -> 184.73.248.186 - HTTP - GET -
 /logic/push?id=h2dmdx8x8hw9&t1=1337368507680&t2=1337368507680&o1=gxp7d9jxo6x1y&o2=h2ch6v0rx51ge&t=1337368507680 - HTTP/1.1
Student No.12 - 10.185.195.212 -> 69.171.227.72 - HTTP - GET -
 /pull?channel=p_100001116249104&seq=19&partition=1&clientid=3d1c1e8e&cb=dj3u&idle=93 - HTTP/1.1
Student No.12 - 10.185.195.212 -> 69.171.227.72 - HTTP - GET -
 /pull?channel=p_100001116249104&seq=14&partition=1&clientid=226e1a4&cb=m12l&idle=21&state=active - HTTP/1.1
Student No.12 - 10.185.195.212 -> 208.46.17.24 - HTTP - GET - / - HTTP/1.1
Student No.12 - 10.185.195.212 -> 72.246.53.65 - HTTP - GET - / - HTTP/1.1
Student No.12 - 10.185.195.212 -> 69.171.234.21 - HTTP - GET -
 /ajax/pagelet/generic.php/MoreStoriesPagelet?__a=1&ajaxpipe=1&ajaxpipe_token=AXI111_EKz7ApSqV&data=%7B%22filter%22%3A%22h_nor%22%2C%22oldest%22%3A1337376330%2C%22oldestMR%22%3A1337300300%2C%22last_seen_time%22%3A1337376484%2C%22scroll_count%22%3A1%2C%22scroll_position%22%3A11%2C%22last_viewstate_id%22%3A%22-7837595259991462488%22%2C%22delay_load_count%22%3A30%7D&__user=100001116249104&__adt=3 - HTTP/1.1
Student No.12 - 10.185.195.212 -> 69.171.227.72 - HTTP - GET -
 /pull?channel=p_100001116249104&seq=20&partition=1&clientid=65b8edcc&cb=3n5l&idle=10&state=active - HTTP/1.1

Figure 31: Results of Experiment 4

Student No.18 - 10.185.209.79 -> 74.125.224.78 - HTTP - GET -
 /safebrowsing/rd/ChFnb29nLXB0aXNoLXNoYXZhcAAAGIHSdCCaK0qwwILPgMA_____PzIGAT4DAP8D - HTTP/1.1
Student No.18 - 10.185.209.79 -> 198.189.255.201 - HTTP - GET - /javasc/7/docs/api/java/lang/Object.html - HTTP/1.1
Student No.18 - 10.185.209.79 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/85c74e3d/PhoneNumberReader.java - HTTP/1.1
Student No.24 - 10.185.202.4 -> 143.127.102.125 - HTTP - GET - /brdefzud=http:%2F%2Fcs17.cs.sjsu.edu%2F&&v=2.5 - HTTP/1.1
Student No.26 - 10.185.201.236 -> 173.194.25.57 - HTTP - GET - /edgedl/googletalk/googletalkplugin/2.9.10.7526/googletalkpluginaccel.msi?ms=nvh&mt=1337367615&cms_redirect=yes - HTTP/1.1
Student No.26 - 10.185.201.236 -> 173.194.25.70 - HTTP - GET - /edgedl/googletalk/googletalkplugin/2.9.10.7526/googletalkpluginaccel.msi?ms=nvh&mt=1337367615&cms_redirect=yes - HTTP/1.1
Student No.26 - 10.185.201.236 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.26 - 10.185.201.236 -> 208.80.152.211 - HTTP - GET - /wikipedia/en/math/2/6/8/26887d99dbc1e32bbfa34256341735d9.png - HTTP/1.1
Student No.26 - 10.185.201.236 -> 208.80.152.211 - HTTP - GET - /wikipedia/en/math/f/3/9/f39d56274a6581e102d27a7ceb5cb2c2.png - HTTP/1.1

Figure 32: Results of Experiment 4

```

Student No.28 - 10.185.194.214 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/BinarySearchTree.java - HTTP/1.1

Student No.29 - 10.185.210.177 -> 184.178.143.164 - HTTP - GET - /GetYLChromeBoot2.ashx - HTTP/1.1
Student No.29 - 10.185.210.177 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.29 - 10.185.210.177 -> 184.178.143.164 - HTTP - GET -
/GetClientData.ashx?key=null&id=f43e8302-1e39-40bb-b31e-
35456cba3762&loc=http%3A//horstmann.com/sjsu/spring2012/cs46b/85c74e3d/ - HTTP/1.1
Student No.29 - 10.185.210.177 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.29 - 10.185.210.177 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.29 - 10.185.210.177 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.29 - 10.185.210.177 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.29 - 10.185.210.177 -> 74.125.224.112 - HTTP - GET -
/complete/search?sugexp=chrome,mod=0&client=chrome&hl=en-
US&q=horstmann.com%2Fsjsu%2Fspring2012%2Fcs46b%2Fs - HTTP/1.1
Student No.29 - 10.185.210.177 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/ - HTTP/1.1

Student No.30 - 10.185.197.34 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/85c74e3d/ -
HTTP/1.1

Student No.32 - 10.185.209.224 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/BinarySearchTree.java - HTTP/1.1
Student No.32 - 10.185.209.224 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.32 - 10.185.209.224 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1

Student No.33 - 10.185.203.245 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/Sum.java - HTTP/1.1
Student No.33 - 10.185.203.245 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/BinarySearchTree.java - HTTP/1.1
Student No.33 - 10.185.203.245 -> 74.125.224.78 - HTTP - GET -
/edgedl/googletalk/googletalkplugin/2.9.10.7526/googletalkpluginaccel.msi - HTTP/1.1

```

Figure 33: Results of Experiment 4

```

Student No.34 - 10.185.193.89 -> 184.73.248.186 - HTTP - GET -
/logic/push?id=h2djwfwrrbu0&t1=1337369667915&t2=1337369667915&o1=gxp7d9jxo6x1yi&o2=h2de1dkikbp137&t=1337369
663447 - HTTP/1.1
Student No.34 - 10.185.193.89 -> 74.125.224.72 - HTTP - GET -
/safebrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAey6ZYfJcXBSoidksBAP_____wcyBmlLAQD_Hw - HTTP/1.1
Student No.34 - 10.185.193.89 -> 74.125.224.72 - HTTP - GET -
/safebrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAAY8egEIMdpBCoNfzQBAP_____AzIGtCQBAP8_ - HTTP/1.1
Student No.34 - 10.185.193.89 -> 74.125.224.72 - HTTP - GET -
/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhcABGNGMBICgjQYqBpSGAQD_HzINUYYBAP_____Bw - HTTP/1.1
Student No.34 - 10.185.193.89 -> 74.125.224.72 - HTTP - GET -
/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhcAAGIn8DCCk_AwyBQk-AwAD - HTTP/1.1
Student No.34 - 10.185.193.89 -> 74.125.224.72 - HTTP - GET -
/safebrowsing/rd/ChFnb29nLXBoaXNoLXNoYXZhcAAGIv8DCCU_AwqBg0-AwD_ADIFCz4DAAM - HTTP/1.1
Student No.34 - 10.185.193.89 -> 184.73.248.186 - HTTP - GET -
/logic/push?id=h2djwfwrrbu0&t1=1337371303359&t2=1337371303359&o1=gxp7d9jxo6x1yi&o2=h2de1dkikbp137&t=1337371
298430 - HTTP/1.1
Student No.34 - 10.185.193.89 -> 184.73.248.186 - HTTP - GET -
/logic/push?id=h2djwfwrrbu0&t1=1337374627320&t2=1337374627320&o1=gxp7d9jxo6x1yi&o2=h2de1dkikbp137&t=1337374
620381 - HTTP/1.1

Student No.39 - 10.185.200.69 -> 74.125.224.102 - HTTP - GET -
/safebrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAAY8egEIMdpBCoNfzQBAP_____AzIGtCQBAP8_ - HTTP/1.1
Student No.39 - 10.185.200.69 -> 199.47.219.150 - HTTP - GET -
/subscribe?host_int=145819350&ns_map=96851500_249181214463532,87619277_205582992209613&ts=1337370782 -
HTTP/1.1
Student No.39 - 10.185.200.69 -> 198.189.255.224 - HTTP - GET -
/javase/6/docs/api/java/lang/System.html - HTTP/1.1

Student No.41 - 10.185.208.127 -> 199.7.50.72 - HTTP - GET -
/MEFwTzBNMEswSTAJBgUrDgMCGGUABBTsQZMG5M8TA9rdzkbCnNwuMad5VgQUz5mp6nsm9EvJjo%2FX8AUm7%2BSP50CEFBII
1IKTxu7fpKH1QnrsX1%3D - HTTP/1.1

```

Figure 34: Results of Experiment 4

```

Student No.42 - 10.185.198.195 -> 204.9.163.247 - HTTP - GET -
/ui/0/5.8.0.158./en/getlatestversion?ver=5.8.0.158&uhash=11bae699d39adf3367593b9b3d352968b&google-
chrome:notoffered;disabled - HTTP/1.1
Student No.42 - 10.185.198.195 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/LinkedList.java - HTTP/1.1
Student No.42 - 10.185.198.195 -> 62.75.183.27 - HTTP - GET - /485/claymore.html - HTTP/1.1
Student No.42 - 10.185.198.195 -> 176.31.248.182 - HTTP - GET - /Manga_Viewer/Legend_of_Tyr - HTTP/1.1
Student No.42 - 10.185.198.195 -> 176.31.248.182 - HTTP - GET - /Manga_Viewer/Magician - HTTP/1.1
Student No.42 - 10.185.198.195 -> 176.31.248.182 - HTTP - GET - /Manga_Viewer/Noblesse_100- - HTTP/1.1
Student No.42 - 10.185.198.195 -> 176.31.248.182 - HTTP - GET - /Manga_Viewer/Superior_Day - HTTP/1.1
Student No.42 - 10.185.198.195 -> 176.31.248.182 - HTTP - GET - /Manga_Viewer/TAL - HTTP/1.1
Student No.42 - 10.185.198.195 -> 176.31.248.182 - HTTP - GET - /Manga_Viewer/The_God_of_High_School -
HTTP/1.1
Student No.42 - 10.185.198.195 -> 62.75.183.27 - HTTP - GET - /408/gamaran.html - HTTP/1.1
Student No.42 - 10.185.198.195 -> 62.75.183.27 - HTTP - GET - /97/gantz.html - HTTP/1.1
Student No.42 - 10.185.198.195 -> 199.19.78.170 - HTTP - GET - /read/_/9757/cradle-of-monsters_ch1_by_death-
toll-scanlations - HTTP/1.1
Student No.42 - 10.185.198.195 -> 199.19.78.170 - HTTP - GET - /read/_/9943/ouroboros_ch1_by_death-toll-
scanlations - HTTP/1.1
Student No.42 - 10.185.198.195 -> 199.19.78.170 - HTTP - GET - /read/_/97877/queen-
emeraldus_v1_ch1_by_happyscans - HTTP/1.1


---


Student No.43 - 10.185.193.167 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/85c74e3d/ - HTTP/1.1
Student No.43 - 10.185.193.167 -> 65.55.53.190 - HTTP - GET -
/StageOne/bluej_exe/3_0_5_0/4e1fd833/ntdll_dll/6_1_7601_17725/4ec49b8f/c000005/00033ab3.htm?LCID=1033&OS=6.1.7601.2.
00010300.1.0.3.17514&SM=Hewlett-
Packard&SPN=HP%20Pavilion%20dm4%20Notebook%20PC&BV=F.26&MRK=103C_HP_CNB_Pavilion%20dm4%20Notebook%20PC_Y53
35KV_0U_QCNU035123M_EFU1006752BIS_4A_I146A_SHP_V58.1F_F.11_T100708_WU3-
0_L409_M3894_J500_7Intel_8655_92.40_%23100830_N10EC8168_(XH124UA%23ABA)_XMOBILE_CN10_Z&MID=EC404F0A-3E87-
4BD9-98A3-DC570E6901EA&HCU=427 - HTTP/1.1
Student No.43 - 10.185.193.167 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/85c74e3d/ - HTTP/1.1
Student No.43 - 10.185.193.167 -> 198.189.255.215 - HTTP - GET - /minitd.flg - HTTP/1.1

```

Figure 35: Results of Experiment 4

```

Student No.45 - 10.185.196.36 -> 74.125.224.98 - HTTP - GET -
/safebrowsing/rd/ChNnb29nLW1hbHdcmUtc2hdmFyEAEY6ZYFDjCXBSoIdUsBAP___w8yBmlLAQD_Dw - HTTP/1.1


---


Student No.47 - 10.185.205.119 -> 205.234.175.175 - HTTP - GET - /images/piazza/dashboard/feed-icons3.png -
HTTP/1.1
Student No.47 - 10.185.205.119 -> 205.234.175.175 - HTTP - GET -
/images/dashboard/common/default_anonymous.png - HTTP/1.1
Student No.47 - 10.185.205.119 -> 205.234.175.175 - HTTP - GET -
/images/dashboard/common/default_user.png - HTTP/1.1
Student No.47 - 10.185.205.119 -> 207.171.163.159 - HTTP - GET - /photos/gq1qze1rVLB/1327851828.png -
HTTP/1.1
Student No.47 - 10.185.205.119 -> 184.73.248.186 - HTTP - GET - /images/dashboard/common/spinner_blue.gif -
HTTP/1.1
Student No.47 - 10.185.205.119 -> 198.189.255.225 - HTTP - GET - /pki/crl/products/microsoftrootcert.crl -
HTTP/1.1


---


Student No.50 - 10.185.210.47 -> 198.189.255.201 - HTTP - GET - /javase/6/docs/api/java/util/Iterator.html -
HTTP/1.1
Student No.50 - 10.185.210.47 -> 199.47.218.147 - HTTP - GET -
/subscribe?host_int=232057978&ns_map=128126920_81732505544&ts=1337376075 - HTTP/1.1


---


Student No.51 - 10.185.193.87 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/85c74e3d/Sum.java -
HTTP/1.1
Student No.51 - 10.185.193.87 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.51 - 10.185.193.87 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.51 - 10.185.193.87 -> 198.189.255.225 - HTTP - GET - /minitd.flg - HTTP/1.1
Student No.51 - 10.185.193.87 -> 198.189.255.225 - HTTP - GET -
/streaming/norton$202009$20streaming$20virus$20definitions_1.0_symalllanguages_livetri.zip - HTTP/1.1
Student No.51 - 10.185.193.87 -> 198.189.255.225 - HTTP - GET -
/streaming/norton$202009$20streaming$20virus$20definitions_1.0_symalllanguages_livetri.zip - HTTP/1.1
Student No.51 - 10.185.193.87 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1

```

Figure 36: Results of Experiment 4

```

Student No.52 - 10.185.197.133 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.52 - 10.185.197.133 -> 199.47.219.148 - HTTP - GET -
/subsctibe?host_int=164694389&ns_map=47438349_52875389819405&ts=1337370411 - HTTP/1.1
Student No.52 - 10.185.197.133 -> 199.47.219.148 - HTTP - GET -
/subsctibe?host_int=164694389&ns_map=47438349_52875389819405&ts=1337371909 - HTTP/1.1
Student No.52 - 10.185.197.133 -> 199.47.219.148 - HTTP - GET -
/subsctibe?host_int=164694389&ns_map=47438349_52875389819405&ts=1337372077 - HTTP/1.1
Student No.52 - 10.185.197.133 -> 199.47.219.148 - HTTP - GET -
/subsctibe?host_int=164694389&ns_map=47438349_52875389819405&ts=1337372742 - HTTP/1.1
Student No.52 - 10.185.197.133 -> 199.47.219.148 - HTTP - GET -
/subsctibe?host_int=164694389&ns_map=47438349_52875389819405&ts=1337376136 - HTTP/1.1

```

```

Student No.54 - 10.185.199.162 -> 4.28.136.36 - HTTP - GET - /bases/av/kdb/i386/kdb-i386-0607g.xml.dif -
HTTP/1.0
Student No.54 - 10.185.199.162 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1

```

```

Student No.61 - 10.185.192.201 -> 74.125.224.102 - HTTP - GET -
/safefrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAEY6ZYFJCXBSoIdksBAP____wcyBmlLAQD_Hw - HTTP/1.1
Student No.61 - 10.185.192.201 -> 199.47.217.144 - HTTP - GET -
/subsctibe?host_int=199855852&ns_map=113183872_7069629353088,113188999_51652796551,124832565_416736660277,1191
39741_9414687452573&ts=1337371136 - HTTP/1.1
Student No.61 - 10.185.192.201 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/BinarySearchTree.java - HTTP/1.1
Student No.61 - 10.185.192.201 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.61 - 10.185.192.201 -> 67.210.118.65 - HTTP - GET - /sjsu/spring2012/cs46b/85c74e3d/ - HTTP/1.1
Student No.61 - 10.185.192.201 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1

```

```

Student No.62 - 10.185.201.68 -> 65.54.87.100 - HTTP - GET - /pki/mscorp/crl/mswww(5).crl - HTTP/1.1

```

Figure 37: Results of Experiment 4

```

Student No.70 - 10.185.208.159 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.122 - HTTP - GET - /simgad/17235009215521256804 - HTTP/1.1
Student No.70 - 10.185.208.159 -> 50.17.214.94 - HTTP - GET -
/ct/0_0_0_6281744_0_7396/us/0/1/0/0/0/15/242/784/0/pixel.gif?v=714&ttid=2&d=ad.doubleclick.net&m=5&r=68443 -
HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.122 - HTTP - GET - /simgad/1530130418844188652 - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.122 - HTTP - GET - /pagead/images/ad_choices_en.png -
HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.122 - HTTP - GET - /pagead/images/ad_choices_1.png - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.122 - HTTP - GET - /simgad/17279640298767510112 - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.79 - HTTP - GET - /v6exp3/redirect.html - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.79 - HTTP - GET - /v6exp3/iframe.html - HTTP/1.1
Student No.70 - 10.185.208.159 -> 212.96.161.250 - HTTP - GET - /softw/12free-vip1/update/avg12infowin.ctf -
HTTP/1.1
Student No.70 - 10.185.208.159 -> 67.210.118.65 - HTTP - GET - /favicon.ico - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.224.97 - HTTP - GET -
/edgedl/chrome/install/1084.46_1025.168/chrome_updater.exe - HTTP/1.1
Student No.70 - 10.185.208.159 -> 74.125.170.240 - HTTP - GET -
/edgedl/chrome/install/1084.46_1025.168/chrome_updater.exe?ms=nhv&mt=1337374813&cms_redirect=yes - HTTP/1.1

```

```

Student No.72 - 10.185.198.174 -> 67.210.118.65 - HTTP - GET -
/sjsu/spring2012/cs46b/85c74e3d/PhoneNumberReader.java - HTTP/1.1

```

```

Student No.79 - 10.185.209.65 -> 74.125.224.70 - HTTP - GET -
/safefrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAEY6ZYFJCXBSoIdksBAP____wcyBmlLAQD_Hw - HTTP/1.1
Student No.79 - 10.185.209.65 -> 74.125.224.70 - HTTP - GET -
/safefrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyEAAy8egEIMdpBCoNfzQBAP_____AzIGcTQBAP8_ - HTTP/1.1
Student No.79 - 10.185.209.65 -> 74.125.224.70 - HTTP - GET -
/safefrowsing/rd/ChFnb29nLXB0aXNoLXNoYXZhcABGNGBICgJYqBpSGAQD_HzINUYBAP_____Bw - HTTP/1.1
Student No.79 - 10.185.209.65 -> 74.125.224.70 - HTTP - GET -
/safefrowsing/rd/ChFnb29nLXB0aXNoLXNoYXZhcAAAGIV8DCCU_AwqBQ4-AwB_MgULPgMABw - HTTP/1.1

```

Figure 38: Results of Experiment 4

CHAPTER 5

Conclusion

In this project we designed an application that was able to track students' activities taking an online test. We used a well-known wireless sniffer Kismet [21] which made most of our tasks automated and simple. After conducting our experiments, our results demonstrate many improvements over previous work completed within our college [32].

The proposed method of wireless sniffing, during an online exam, is fast and can easily handle quite a number of students and their traffic. This solution can easily be ported to any exam hall for conducting online exam. The proposed solution of online testing also removes the dependency on paper tests and manual work required for checking student answers. Since this solution does not require any software installation on a student's computer, its set-up is quick and does not require additional attention except for the part where a student needs to connect to the test server. Since students can easily go online and start the test, it makes an instructor's job easier. Test monitoring requires less attention, as all activities performed by the student are being logged by Kismet. The real time update on the web page, displaying the name and details of a student involved in cheating, makes this system very friendly and useful for instructors.

CHAPTER 6

Future Work

The project can be of immense use if there could be a way to read the packet contents flowing through the network. This project can be modified to capture packets and scan for key words from the test. Students might have found or set-up their own proxy server on a non-standard port and might be using it to search the answers. Therefore, if a keyword is found from the packet belonging to the student IP taking exam and that packet is going to a non-white-listed website, that student can be flagged. Keywords taken from exam questions can also be placed in the database for the sniffer to capture all the packets from that data. This can determine if the student is using help outside the classroom in an attempt to cheat.

Due to false alarms, in a real class full of students, the algorithm for flagging students as cheaters has to be modified in order to capture data going on port 80 and some other non-standard ports like port 8080. The HTTP data is more important; however all other data should be logged as it might be used for post-test analysis. Due to these false positives the white list of each IP should also be modified to include broadcast addresses. We should also extend our tool to track the non-HTTP traffic as well. This will give us more data if a student tries to cheat using some other channel.

LIST OF REFERENCES

- [1] A. Etter, (2002). A Guide to Wardriving and Detecting Wardrivers. SANS Institute, 8-9
- [2] Analyzing network traffic with tcpdump <http://tournasdimitrios1.wordpress.com/2011/02/19/analyzing-network-traffic-with-tcpdump-part-1/>
- [3] Android Developers Portal, Android Notification, May 20, 2012 <http://developer.android.com/guide/topics/ui/notifiers/notifications.html>
- [4] A. Nigavekar, & W. Harris, (2010). Examinations and the Role of Technology: Emerging Directions. New Delhi: EDGE-ValueNotes
- [5] B. Haines, F. Thornton, & M. Schearer. (2008). Kismet Hacking. Massachusetts: Syngress
- [6] Change MAC address, May 20, 2012 <http://lantoolbox.com/articles/find-and-change-spoof-your-mac-address-in-windows-xp/>
- [7] C. Hurley, (2007). Penetration Tester's Open Source Toolkit. Burlington: Syngress Publishing, Inc
- [8] C. Pan, K. Yang, & T. Lee. (2004). Secure Online Examination Architecture Based on Distributed Firewall. IEEE International Conference on e-Technology, e-Commerce and e-Service, 533-536
- [9] D. Dalibor Dvorski, March 2007. Installing, configuring and developing with XAMPP
- [10] F. Fuentes, & Kar, D. (2005). Ethereal vs. tcpdump: a comparative study on packet sniffing tools for educational purpose. Texas: Texas A&M University-Corpus Christi.
- [11] G. Cluskey, C. Ehlen, & M. Raiborn, (2011). Thwarting online exam cheating without proctor supervision. Journal of Academic and Business Ethics, 1-6.
- [12] G. Costagliola, V. Fuccella, M. Giordano, & G. Polese, (2009). Monitoring Online Tests through Data Visualization. Knowledge and Data Engineering IEEE Transactions on, 21(6), 773,778,782
- [13] G. Costagliola, & V. Fuccella, Online testing, current issues and future trends. Journal of e-Learning and Knowledge Society, pp 81-83

- [14] G. Villegas, (2008). Analysis of Tool for conducting Wireless Penetration Testing. Texas: Texas A&M University-Corpus Christi.
- [15] I. Jung, & Yeom, H. (2009). Enhanced Security for Online Exams Using Group Cryptography. IEEE transactions on education, 52, 341-343
- [16] IP Sniffer, May 20, 2012 http://www.scanwith.com/download/IP_Sniffer.htm
- [17] Java RMI example, May 20, 2012 <http://www.javacoffeebreak.com/articles/javarmi/javarmi.html>
- [18] J. Castella-Roca, J. Herrera-Joancomarti, & A. Dorca-Josa, (2006). A secure e-exam management system. First International Conference on Availability Reliability and Security ARES06, 5-8
- [19] J. Murray. (2009). An Inexpensive wireless IDS using Kismet and OpenWRT. SANS Institute, 8-9
- [20] J. Sung (2009). U-Learning Model Design Based on Ubiquitous Environment. International Journal of Advanced Science and Technology, pp. 77-86
- [21] Kismet, a wireless packet sniffer, May 15, 2012, <https://www.kismetwireless.net/>
- [22] MAC Sniffer, May 20, 2012 http://www.filebuzz.com/fileinfo/40152/Wake_on_LAN_Packet_Sniffer.html
- [23] M. Frandsen, (2010). Detection of cheating when students use their own computers during examinations. Denmark: Technical University of Denmark (DTU).
- [24] M. Kershaw, (2009). Kismet Documentation. Retrieved July 11, 2011, from <http://www.kismetwireless.net>
- [25] N. Rowe, (2004). Cheating in Online Student Assessment: Beyond Plagiarism. Online Journal Of Distance Learning Administration, 7(II), 4.
- [26] Oracles Java RMI, May 20, 2012 <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html>
- [27] Peter Kemper, Android Notifications, May 20,2012 <http://www.cs.wm.edu/~kemper/cs301/slides/a11.pdf>
- [28] Public Proxy Server, May 20, 2012 <http://www.publicproxyservers.com/>
- [29] Python, programming language, May 20, 2012 www.python.org
- [30] R. McRee, (2006). Security Analysis with Wireshark. ISSA Journal, 39-45.

- [31] R. Spangler, (2003). Packet Sniffer Detection with AntiSniff. Wisconsin: University of Wisconsin - Whitewater
- [32] S. Anandan, Online Application Monitoring Tool, Master's report, Department of Computer Science, San Jose State University, 2010,
- [33] Sniffing, methodology and definition, May 15, 2012 http://en.wikipedia.org/wiki/Packet_sniffing
- [34] Spyware, definition, May 20, 2012 <http://en.wikipedia.org/wiki/Spyware>
- [35] Spyware, overview and their types, May 15, 2012, http://www.us-cert.gov/reading_room/spywarehome_0905.pdf
- [36] Technitium MAC Address Changer, configuration and installation, May 24, 2012 <http://www.technitium.com/tmac/index.html>
- [37] T. King, (2002). Packet Sniffing In a Switched Environment. Information Security, 1-6
- [38] Wireshark and Tshark, a wireless sniffer, May 15, 2012, <http://www.wireshark.org/docs/man-pages/tshark.html>
- [39] Wireshark screenshot, May 15, 2012 http://en.wikipedia.org/wiki/File:Wireshark_screenshot.png
- [40] XAMPP server, configuration and installation, May 15, 2012 <http://www.apachefriends.org/en/xampp.html>