Network security attacks and defense

By

NAUMAN SHAH

Project Report in partial fulfillment of the requirements for the award of Bachelor of Information Technology degree

School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Rawalpindi, Pakistan
2008

National University of Sciences & Technology
School of Electrical Engineering and computer science

Network security attacks and defense

By: Nauman Shah

Project Advisor: Fauzan Mirza

Co-Advisor: Ali Khayam

National University of Sciences & Technology

School of Electrical Engineering and computer science

CERTIFICATE

Certified that the contents and the form of the project entitled "Network Security Attacks and Defense" submitted by Nauman Shah have been found satisfactory for the requirement of the degree.

Advisor: Fauzan

Dr. Fauzan Mirza

Co-Advisor: Ali Khayam

Dr. Ali Khayam

1st Committee member: [illegible]

Miss. Shamaila Kiyani

2nd Committee member: [illegible]

Mr. Qasim Rajpoot

DEDICATION

We are absolutely nothing, except as God chooses to use us in His purpose. All our meaning and value lies in being part of His work. Outside of that, we are empty, uninteresting chunks of flesh, like the cattle. If we wish to have any meaning or permanence -- to be anything other than dumb, perishing animals -- we must fill our minds with God's Word, and fill our lives with His service

ACKNOWLEDGEMENT

Praise to Allah who has created us. He is the sustainer of the world; He is the creator of the world and whatever in between. He has blessed us all and provided us the opportunity to prove ourselves His obedient creatures.

After that I would like to thank my family for providing all the support that I required to achieve all the goals of my life.

After that I would like to thank all the teachers who were always there for me when I needed guidance. I acknowledge the support of Dr. Fauzan Mirza, Dr. Ali Khayam, Sir Qasim Rajpoot, and Madam Shamaila Kiyani. Without their support this project couldn't be completed.

After that I would like to thank all faculty members who made my degree in SEECS a success. Special thanks to Sir Atif Kamal, Sir Mujhtaba Haider, Sir Umer Kalim, Sir Saad liaqat Qiyani, Sir Atif, Khawar Bhai, and Towhid-ul-Islam.

I would like to offer special thanks to all my fellow students who made my 4 years here exciting and memorable.

TABLE OF CONTENTS

Chapter 1 8
1.1 PROBLEM STATEMENT 8
1.2 MOTIVATION 8
Chapter 2 10
RELATED WORK & LITERATURE SURVEY 10
2.1 ARP POISONING 10
2.2 IP SPOOFING 13
2.3 TCP HIJACKING 14
2.4 SYN- FLOODING 15
2.5 BUFFER OVERFLOW 17
2.6 FORMAT VULNERABILITIES 19
2.7 SQL INJECTIONS 20
2.8 SHELL CODE FORMULATION 21
Chapter 3 23
ANALYSIS 23
3.1 ARCHITECTURE DIAGRAM 26
3.2 CLASS DIAGRAMS 27
3.2.1 CTF Server Class Diagram 27
3.2.2 CTF Client Class Diagram 28
3.3 FLOW DIAGRAMS 29
3.3.1 CTF Game Server Flow Diagram 29
3.3.2 CTF Game Client Flow Diagram 31
3.3.3 Sequence Diagram 32
3.3.4 Use Case Diagram 33
Chapter 4 34
IMPLEMENTATION 34
4.1 IP SPOOFING AND SYN-FLOODING 34
4.2 SQL INJECTION 34
4.3 BUFFER OVER FLOW 36
Chapter 5 37
RESULTS 37
Chapter 6 38
CONCLUSION 38
Chapter 7 39
RECOMMENDATIONS 39
Chapter 8 40
Deliverable(s) 40
8.1 SCHEDULE OF DELIVERABLES 41
References 42

LIST OF FIGURES

Figure 1 Ethernet Type II Frame format 10
Figure 2 ARP-Poisoning 10
Figure 3 ARP Poisoning 11
Figure 4 ARP-poisoning 12
Figure 5 IP Spoofing 14
Figure 6 TCP Hijacking 15
Figure 7 SYN-Flooding 17
Figure 8 Classic Buffer Overflow 18
Figure 9 Stack Buffer Overflow 19
Figure 10 Format Vulnerability 20
Figure 11 Shell code Formulation worms sending shell commands 21
Figure 12 Shell code connecting to attacker [38] 22
Figure 13 Shell Code stack pointer Alignment 22
Figure 14 Basic RMI 23
Figure 15 Reference Passing 24
Figure 16 Abstract P2P Diagram 25
Figure 17 Architecture Diagram 26
Figure 18 Server Class Diagram 27
Figure 19 Client Class Diagram 28
Figure 20 Server Flow Diagram 29
Figure 21 Client Flow Diagram 31
Figure 22 Sequence Diagram 32
Figure 23 Use case Diagram 33
Figure 24 Deliverables 41

LIST OF ABBREVIATIONS

CTF	Capture the Flag
ARP	Address Resolution Protocol
RMI	Remote Method Invocation
TCP	Transmission Control Protocol
P2P	Peer to Peer
MAC	MediaAccessControl

Chapter 1

INTRODUCTION

1.1 PROBLEM STATEMENT

This project studies in considerable detail and implements a variety of network security attacks, from each of the four layers, with the objective of designing and implementing a vulnerable network and hosts to provide a suitable case study for learning about network security attack and defense techniques. The result may be used as the basis for a study or training exercise similar to the exercise deployed in the international Capture the Flag (CTF) competition. Examples of vulnerabilities to be studied and in consideration are: ARP poisoning, IP spoofing, TCP hijacking, SYN flooding, buffer overflows, format vulnerabilities, SQL injection and shell-code formulation.

1.2 MOTIVATION

Network security is an important issue in any organization that manages electronic data with a LAN or that has access to an insecure public WAN such as the Internet. Many different types of threats to the confidentiality, integrity and availability of electronically-stored information all exists and suitable security measures are necessary to mitigate them. Attacks on network security can be conducted internally (e.g., by malicious employees of the organization) or externally (e.g., by hackers). In both cases, the method of conducting, detecting, and defending attacks differ. Additionally, an attack on the security of a network or its host may exploit vulnerabilities in any of four layers of the protocol stack: data-link, network, transport and application. The types of attacks in each of these layers uniquely exploit features in the design of the layers. As a result, there are a large number of attacks that may be conducted, although the number of actual attacks that will work will depend vastly on technical details of the specific network being attacked.

The objective of the project is to implement a vulnerable network in which some of these attacks are possible so that it serves as a case study for network attack and defense. In computer security CTF is a "computer security war game". Each team is given a machine on a small isolated network and they have to defend it. Teams are scored by either defending or by attacking the opponent system. Depending upon the nature of the game they steal the flags.[1] The project objective is to make a Capture The Flag game with software vulnerabilities which can be exploited by the players. The participants are allowed to study the client code which will be running at the participants end and they can change the code as well. The second part of the project was to look for loop holes in the client side code which can be used for attacks, and implement those attacks. These implementation will serve as case studies for the network adminitrators to understand these attacks and how to defend themselves.

Chapter 2

RELATED WORK & LITERATURE SURVEY

To create these vulnerabilities one should have idea of how these vulnerablities are created and what are they.

2.1 ARP POISONING

ARP is address resolution protocol. In this protocol if the machine A has to send some thing to machine B it must have the IP and the MAC address of the machine B in order to have a proper communication. ARP is maintained for this purpose. It consist of MAC to IP and IP to MAC mapping. The user sends an ARP request and the machine with that IP replies back the ARP request with its MAC address.

This is a normal ethernet II frame format:

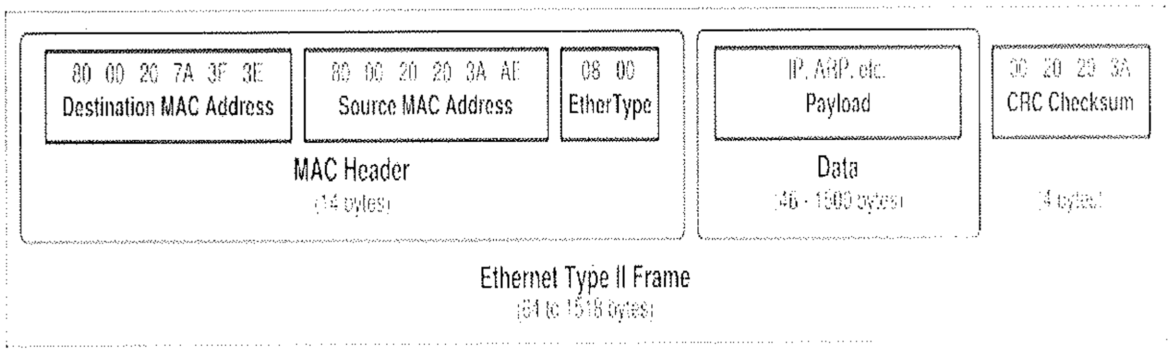


Figure 1 Ethernet_Type_II_Frame_format [27]

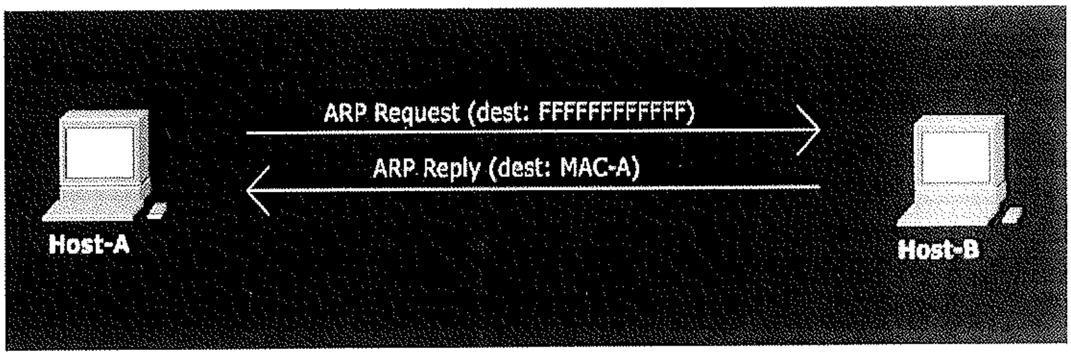


Figure 2 ARP-Poisoning [28]

A normal switch make its routing table by extracting information from the processed data and if the entry is not found it forward the data to all ports. Once the mapping is done specific ports are used. Without a mirror port the sniffer console can capture only the unicast packets to or from itself and multicast traffic.

The problem is there is no authentication procedure in the ARP. i.e. there is no way in the protocol to authenticate that the ARP reply was from the original machine because ARP is stateless protocol which donot require any authentication in other words a simple ARP packet can force the communicating parties to update their cache and that's how a hacker can easily route the traffic according to his will. If port security is not implemented in the switch the hacker can easily mask its MAC address.

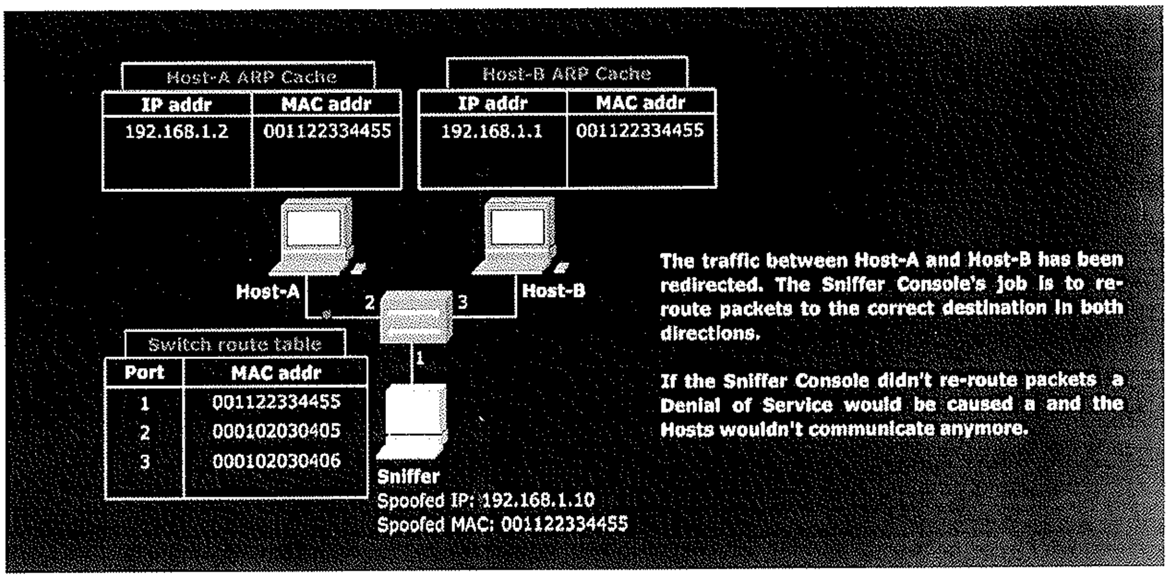
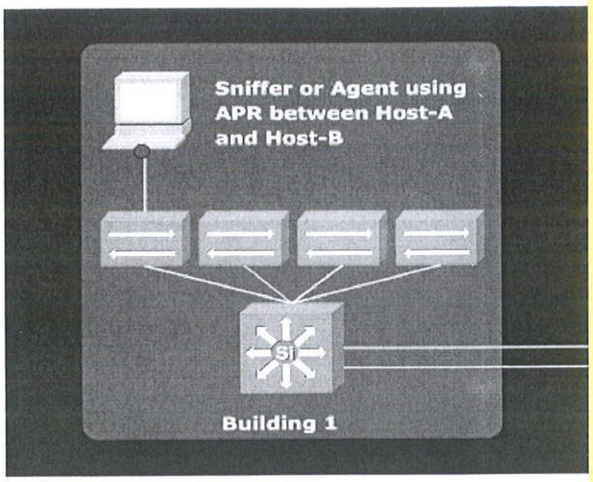

Host-A ARP Cache
IP addr
MAC addr
192.168.1.2
001122334455
Host-B ARP Cache
IP addr
MAC addr
192.168.1.1
001122334455
Host-A
Host-B
2
3
1
Switch route table
Port
MAC addr
1
001122334455
2
000102030405
3
000102030406
Sniffer
Spoofed IP: 192.168.1.10
Spoofed MAC: 001122334455
The traffic between Host-A and Host-B has been redirected. The Sniffer Console's job is to re-route packets to the correct destination in both directions.
If the Sniffer Console didn't re-route packets a Denial of Service would be caused a and the Hosts wouldn't communicate anymore.

Figure 3 ARP Poisoning [29]

An intruder can use this technique to hijack TCP connections, sniff paswords, lan discovery, and make DOS attack on any host in that subnet. The things that limits this attacks are layer three devices and VLANs which both seperates the broadcast domain.



→ should have wrote
DOS
→ this is an attack
(2nd line)
→ Punctuation mistakes.
(fail to deliver the
meaning of line)
→ Next line start with
a capital word. {when they
switch — —
→ Comma missing in the
same line
→ like :—
→ Console (spelling)

Figure 4 ARP-poisoning [30]

Dos attack is denial of service attacks in which a machine is flooded with packets so that it becomes out of service,its an attack on the availablity of the service.

Man in the middle means that all of your confidential data is going through another machine which in the middle of the parties communicating.

MAC flooding is done through overloading the switch. Overloaded switches act as hubs.when they switch to hub mode they start broad casting all the data which can easily be sniffed.

Requirement for ARP poisoning is that the hacker should be inside the network or he should have remote access to a machine in the network [2].

But there are still many things to consider for ARP poisoning like

1. ARP poisoning only works in the broadcast domain so it won't work to redirect the traffic between different host on different VLANs and on different subnets.
2. The sniffer concsole must be able to redirect the packets in the right direction otherwise the hosts won't participate in communication.
3. The sniffer console must know where to route the packets so it must have the IP and MAC of the interested hosts.
4. The ARP attack slow down the network performance as well, it will require a lot of processing on the sniffer console as well.

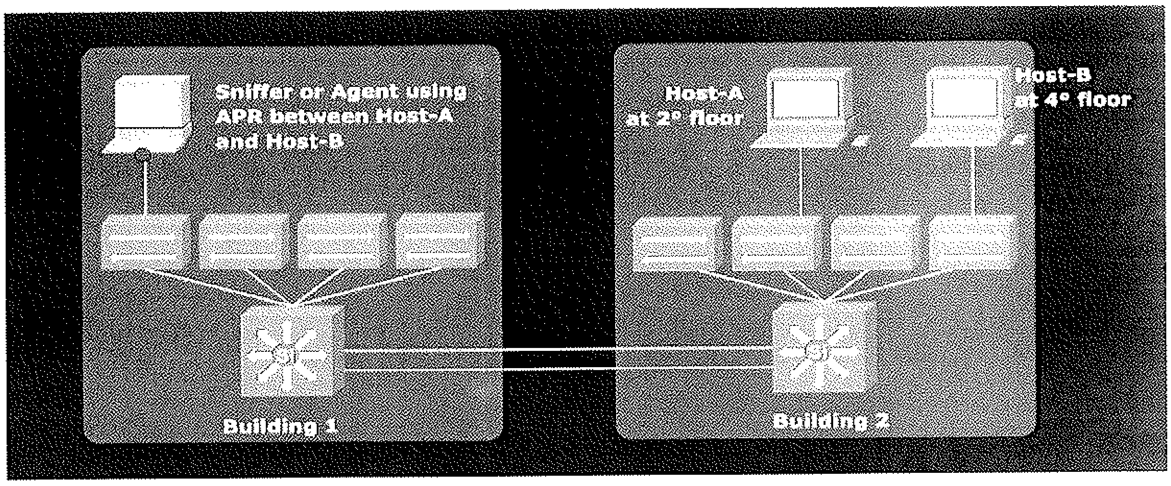


Figure 4 ARP-poisoning [30]

Dos attack is denial of service attacks in which a machine is flooded with packets so that it becomes out of service,its an attack on the availablity of the service.

Man in the middle means that all of your confidential data is going through another machine which in the middle of the parties communicating.

MAC flooding is done through overloading the switch. Overloaded switches act as hubs.when they switch to hub mode they start broad casting all the data which can easily be sniffed.

Requirement for ARP poisoning is that the hacker should be inside the network or he should have remote access to a machine in the network [2].

But there are still many things to consider for ARP poisoning like

1. ARP poisoning only works in the broadcast domain so it won't work to redirect the traffic between different host on different VLANs and on different subnets.
2. The sniffer concesole must be able to redirect the packets in the right direction otherwise the hosts won't participate in communication.
3. The sniffer console must know where to route the packets so it must have the IP and MAC of the interested hosts.
4. The ARP attack slow down the network performance as well, it will require a lot of processing on the sniffer console as well.

5. The entry of the interested hosts must be present in the ARP cache as through ARP poisoning we cannot insert a new entry in the ARP cache.

2.2 IP SPOOFING

TCP layer 4 (transport) is connection oriented in which session is maintained in three way hand shake. The intruder change the destination address of the packet so to hide its own identity from the target machine. The destination addresss can be easily masked and spoofed.

Examples of configurations that are potentially vulnerable include routers to external networks that support multiple internal interfaces routers with two interfaces that support sub-netting on the internal network proxy firewalls where the proxy applications use the source IP address for authentication To gain access, intruders create packets with spoofed source IP addresses. This exploits applications that use authentication based on IP addresses and leads to unauthorized user and possibly root access on the targeted system [3].

IP spoofing helps a malicious user to launch other attacks like TCP-hijacking, SYN-flooding, ARP poisoning, DOS attack and many more. Most of the attacks start with spoofed IP addresses. This technique can be used conceal the attackers true identity from the victim.

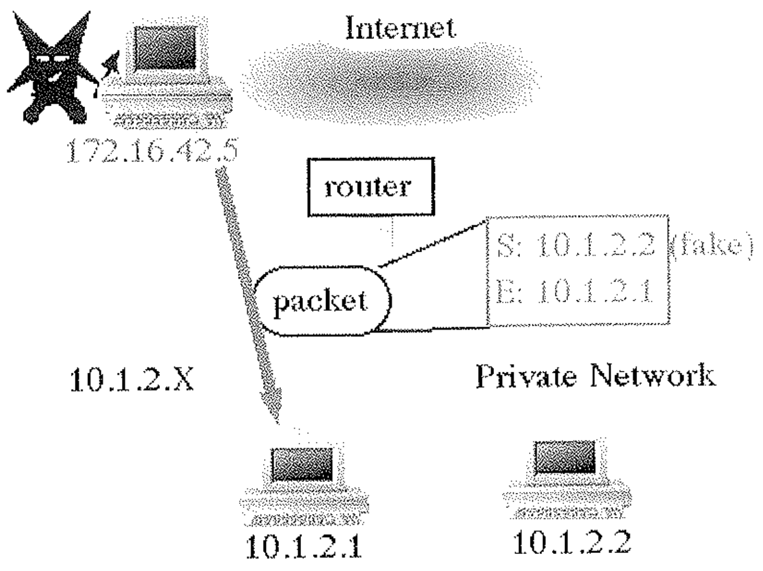


Figure 5 IP Spoofing [31]

2.3 TCP HIJACKING

When a client tries to create a TCP connection the client and server exchange a set messages. The client system begins by sending a SYN message to the server. The server then acknowledges the SYN message by sending SYN-ACK message to the client. The client then finishes establishing the connection by responding with an ACK message. The connection between the client and the server is then open, and the service-specific data can be exchanged between the client and the server.

Suppose we have two systems A and B. they wants to open up a tcp connection so that they can communicate. The process of the connection can be demonstrated by the following diagram.

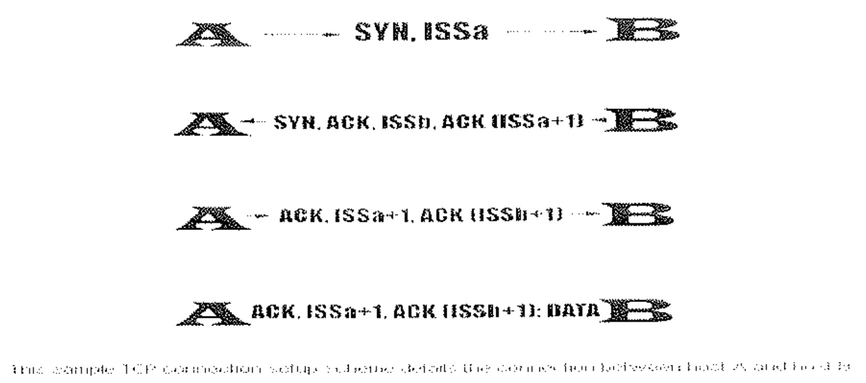

A — SYN, ISSa — B
A — SYN, ACK, ISSb, ACK (ISSa+1) — B
A — ACK, ISSa+1, ACK (ISSb+1) — B
A ACK, ISSa+1, ACK (ISSb+1): DATA B
This sample TCP connection setup scheme details the connection between host A and host B

Figure 6 TCP Hijacking [32]

The hacker tries to hijack a built connection as most of the authentication is done in the initial steps so it is easy to get access to the server. The hacker take over the TCP session between two machines. The most popular method is the source routed packets. In this the hacker can participate in the communication by making all the packets pass through him or he can use blind hijacking in which the hacker don't get any response and send a command which can be to set a password to get an access.

The other way of doing it is to guess the two sequence numbers. If some how the attacker gets the sequence numbers it can easily get over the connection.

FTP and telnet both uses TCP. Both of these protocols donot check for the source address i.e the host who send the fake packet. Both these protocols consider the fake packet to be from authentic client and thats how the attacker take over the connection. The sequence number donot matches the authentic user and the connection is droped between him and the server.

2.4 SYN- FLOODING

TCP connection is a three way hand shake protocol. Which initiates when the client sends a SYN message to the server, which in response sends a SYN with ACK messge and then client sends the final ACK which establishes the connection between server and client. SYN flooding is achieved by creating a half opened connection with the server. The server is waiting for the ACK but the IP is masked so that it never completes a connection. By doing so the hacker binds the server resources and flood it with SYN messages. It helps in making a DOS attack [5].

SYN- flooding donot change the confidential data on the target. It is not a threat to the integrity of the data. SYN-flooding mostly used to make the service unavailable for the clients who are trying to connect.

Most OSs had a table to limit the number of connection to it at a specific time. The attacker intend to fill this table with half opened connetions where the server is waiting for the final ACK. But now the OSs have efficient table scheduling which makes it difficult to attack. This attack can be described in these steps:

1. The attacker creates a fake packet with spoofed source address.
2. SYN flag is set in the packet so that after receiving the server try to open a connection with the spoofed ip packet which don't exist.
3. Victim waits for the response from the source whose IP was forged into the packet (which don't exists).
4. Creating no conntion with other clients.
5. After the attacker stops the SYN-flooding the server goes back to normal state.
6. It is difficult to crash a server with SYN-flooding.
7. It can be used to disable one side of a connection in TCP-hijacking, as well as authenticating and logging between servers.

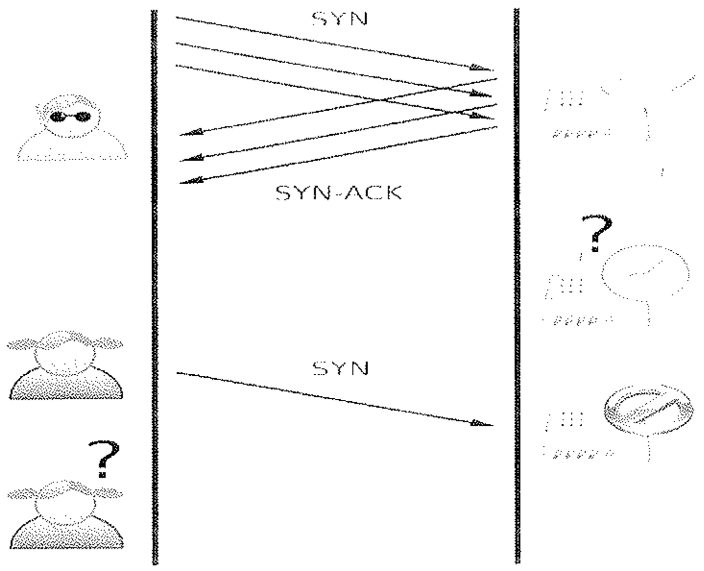


Figure 7 SYN-Flooding [33]

These attacks can be prevented by increasing the table size so that it can accommodate more connection. We can alocate more 100 bytes of space for TCP connections.

Another solution is to make the timeout for connection shorter. In this case the server will remove all those requests which have spend more than the allocated time. So in case of SYN-flooding the server will able to efficiently utilize the table and will free the stack memory in order to comply to authentic clients.

The other way of preventing is selective drop. When ever a new request for connection arrives the table remove the entry which was the oldest and allocate that to the new incoming connection.

2.5 BUFFER OVERFLOW

The buffer overflow is a programatic error that results in erratic behavior of a program. It can be a malicious user trying to run a code a to breach the system security. It is a condition when a user tries to input data that is beyond the storage of the fixed length buffer thus over writing the previous data.

The over written data might include other buffers, variables which might crash the program or produce incorrect results. Buffer overflow causes many vulnerabilities which are the basis for many malicious exploits[6][7].

The buffer overflow vulnerability attack against both legacy and newly developed application is quite the same. The problem is its quite difficult to discover a buffer overflows and exploit it. But in a wide vareity of ways buffer over flow can occur.

In a classic buffer overflow exploit, the attacker sends data to a program, which it stores in an undersized stack buffer. The result is that information on the call stack is overwritten, including the function's return pointer. The data sets the value of the return pointer so that when the function returns, it transfers control to malicious code contained in the attacker's data. [8]

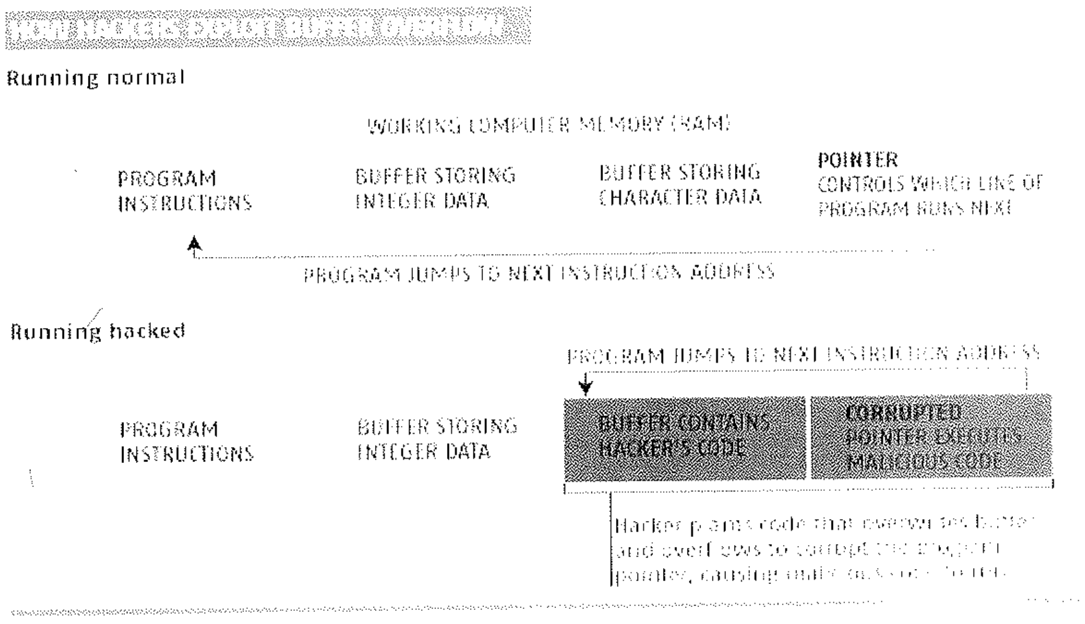

HOW HACKERS EXPLOIT BUFFER OVERFLOW
Running normal
WORKING COMPUTER MEMORY (RAM)
PROGRAM INSTRUCTIONS
BUFFER STORING INTEGER DATA
BUFFER STORING CHARACTER DATA
POINTER
CONTROLS WHICH LINE OF PROGRAM RUNS NEXT
PROGRAM JUMPS TO NEXT INSTRUCTION ADDRESS
Running hacked
PROGRAM JUMPS TO NEXT INSTRUCTION ADDRESS
PROGRAM INSTRUCTIONS
BUFFER STORING INTEGER DATA
BUFFER CONTAINS HACKER'S CODE
CORRUPTED POINTER EXECUTES MALICIOUS CODE

Figure 8 Classic Buffer Overflow [34]

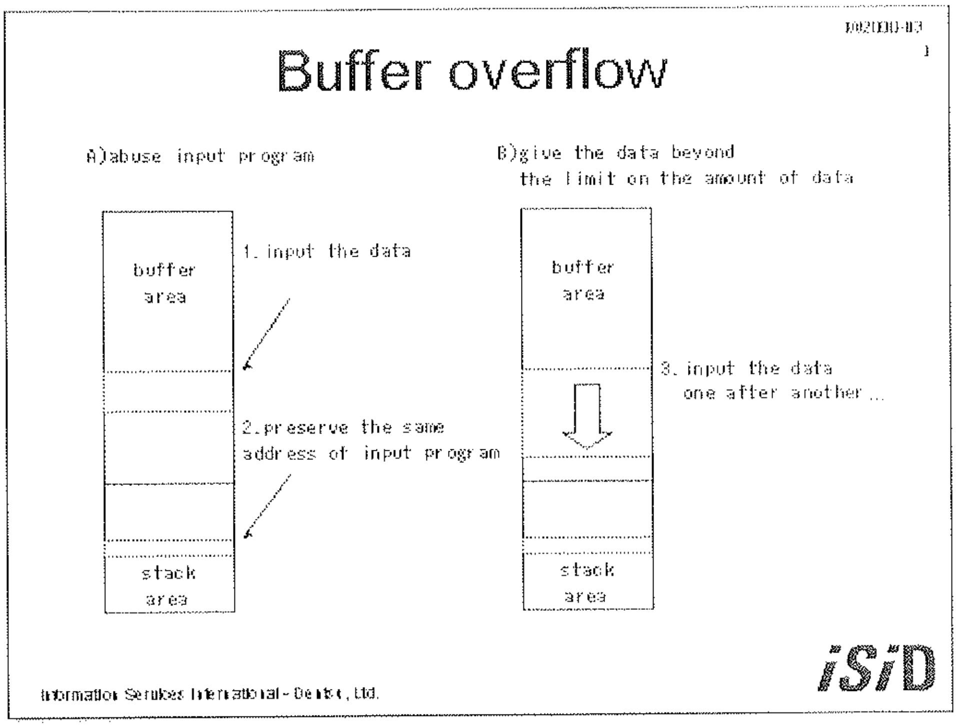

Buffer overflow
A)abuse input program
B)give the data beyond the limit on the amount of data
buffer area
1. input the data
2. preserve the same address of input program
stack area
buffer area
3. input the data one after another...
stack area
Information Services International - Dentsu, Ltd.
iSiD

Figure 9 Stack Buffer Overflow [35]

2.6 FORMAT VULNERABILITIES

File format vulnerabilities are taking the center stage in information system security threats faced by many enterprizes. Hackers exploiting this vulnerability create well crafted malicious files which can trigger flaws in the operating system. Like a file format vulnerability in adobe acrobate might allow a malicious user to make a malicious PDF file that compromises linux, windows, or machintosh systems. This vulnerability usually triggers flaws like buffer overflow in many applications, denial of service attack , or execute arbitrary code. Most of these files are crafted in order to attack the OS.

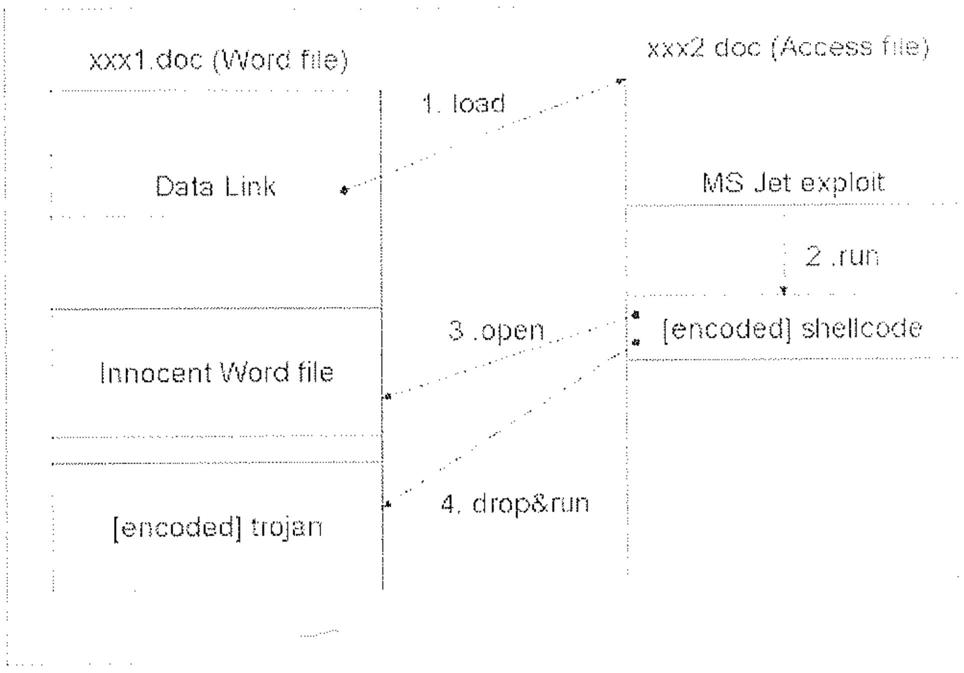


Figure 10 Format Vulnerability[36]

2.7 SQL INJECTIONS

SQL injection is one of type of web hacking that require nothing but port 80 and it might just work even if the admin is patch-happy. It attacks on the web application (like ASP, JSP, PHP, CGI, etc) itself rather than on the web server or services running in the OS. It is a trick to inject SQL query/command as an input possibly via web pages. Many web pages take parameters from web user, and make SQL query to the database. Take for instance when a user login, web page that user name and password and make SQL query to the database to check if a user has valid name and password. With SQL Injection, it is possible for us to send crafted user name and/or password field that will change the SQL query and thus grant us something else [9].

SQL injection is an exploit of vulnerability in the database of an application. The vulnerability exists if the user input is incorrectly checked for escape sequences or the user input is directly fed into the queries. This result in manipulation of actions performed on the database by the end user. For example we have a query which checks for the user name and password for authentication in the capture the flag game.

Statement: select * from login where user name ='user' and user pass ='pass'; Now if we put the password as: (a' or't' ='t) and user name as Nauman the query becomes Statement: select * from login where user name ='Nauman' and user pass ='a' or 't' ='t'; Now pass='a' returns false but it is or with 't' ='t' which returns always true.

The last segment returns true which is finally and with user name so this query will always return a record and thus the user is authenticated without a valid password.

2.8 SHELL CODE FORMULATION

In the simplest variant, the return address on the stack is overwritten with an address of the attacker's own code. Making the stack memory pages non-executable makes placement of the attacker's code on the stack harmless. This technique typically requires hardware support to be efficient. Executable code can still be injected if other data areas are executable [10].

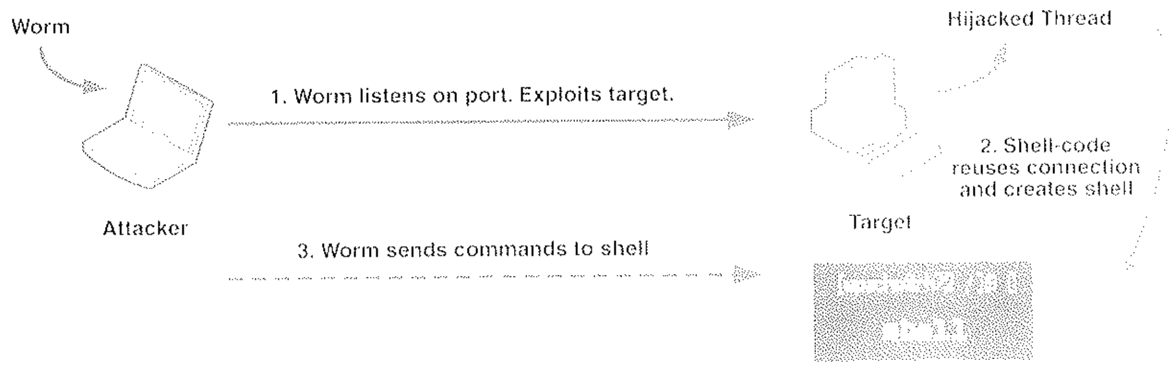

Worm
1. Worm listens on port. Exploits target.
Hijacked Thread
2. Shell-code reuses connection and creates shell
Attacker
Target
3. Worm sends commands to shell

Figure 11 Shell code Formulation worms sending shell commands [37]

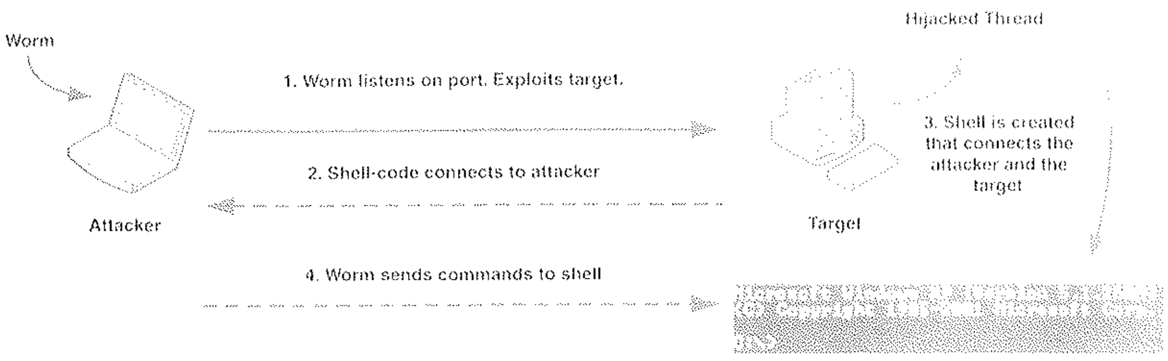

Worm
Hijacked Thread
1. Worm listens on port. Exploits target.
2. Shell-code connects to attacker
3. Shell is created that connects the attacker and the target
Attacker
Target
4. Worm sends commands to shell

Figure 12 Shell code connecting to attacker [38]

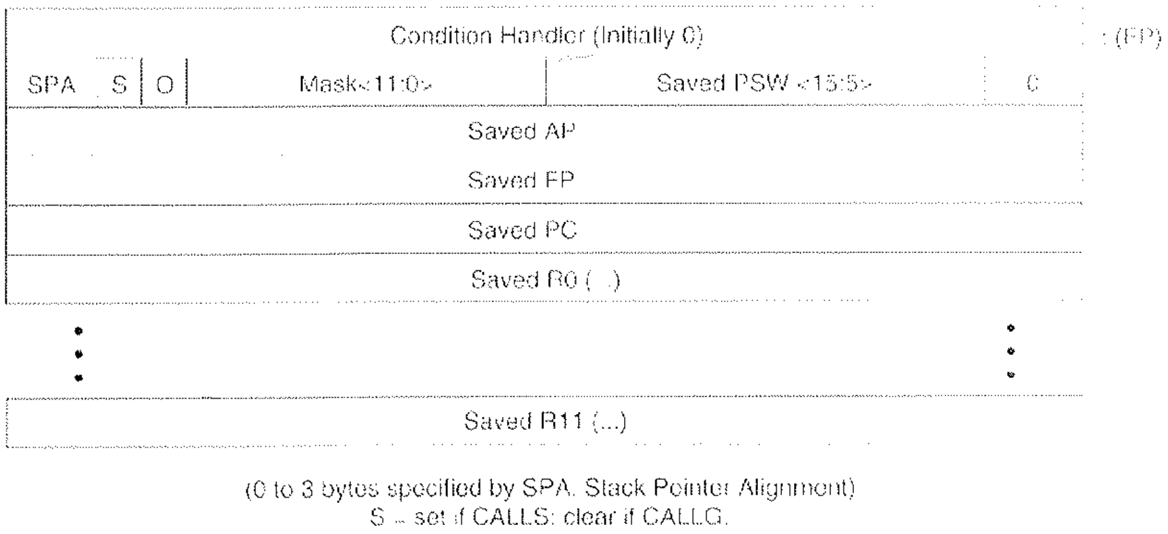

Condition Handler (Initially 0)
:(FP)
SPA
S
0
Mask<11:0>
Saved PSW <15:5>
0
Saved AP
Saved FP
Saved PC
Saved R0 (...)
Saved R11 (...)
(0 to 3 bytes specified by SPA, Stack Pointer Alignment)
S = set if CALLS; clear if CALLG.

Figure 13 Shell Code stack pointer Alignment [39]

Chapter 3

ANALYSIS

The objective of this project is to make a CTF game that has all the vulnerabilities as discussed above. The game is like we have some dictators who wants to take over the world. To do that he should have a maximum number of soldiers. To start with, every dictator have a fix number of soldiers, to increase his army he buys the other dictator's army. In the end the winner would be the dictator with maximum number of soldiers. For this we have a client-server architecture. The server hold the information about the soldiers and the client is the dictator. The game is the main challenge because the rest of the project is to identify the loop holes and exploit those vulnerabilities, the main objective is to create a sort of network game with vulnerabilities. The players would use any of the vulnerabilities mentioned above to win the game and all those vulnerabilities would serve as a case study for network security. The language for this project would be java. The server will publish the services and the clients will access them and it would be like an RMI architecture.

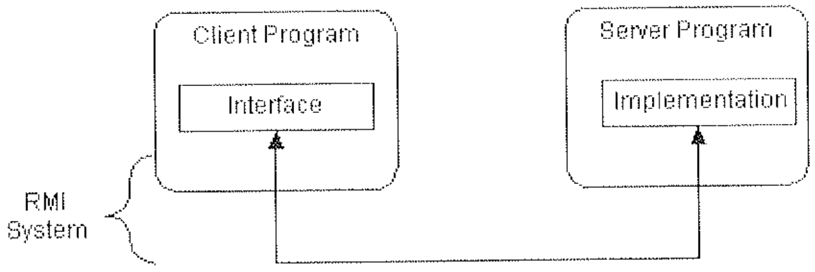


Figure 14 Basic RMI

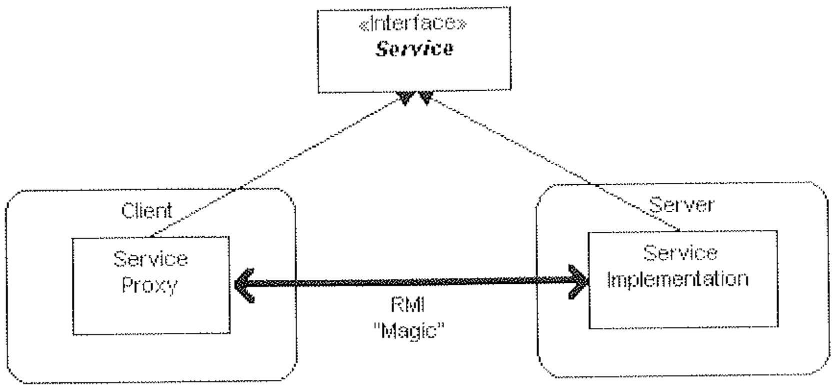


Figure 15 Reference Passing

The first design of application was a pure p2p. In which the clients do transaction with each other. Each client was keeping information of every client (player) resource. The basic problem that arouse of that was the synchronization of the database. Let a transaction is made and during the update one of the participants goes offline, its database won't get updated. Solution to this problem was either each participant keeps his information or during login it synchronizes it with the rest of the participants. Both the solution might slow down the run time of the CTF game. So an alternate solution of hybrid p2p was chosen in order to cater for this problem. The hybrid P2P solution was like this at the abstract level.

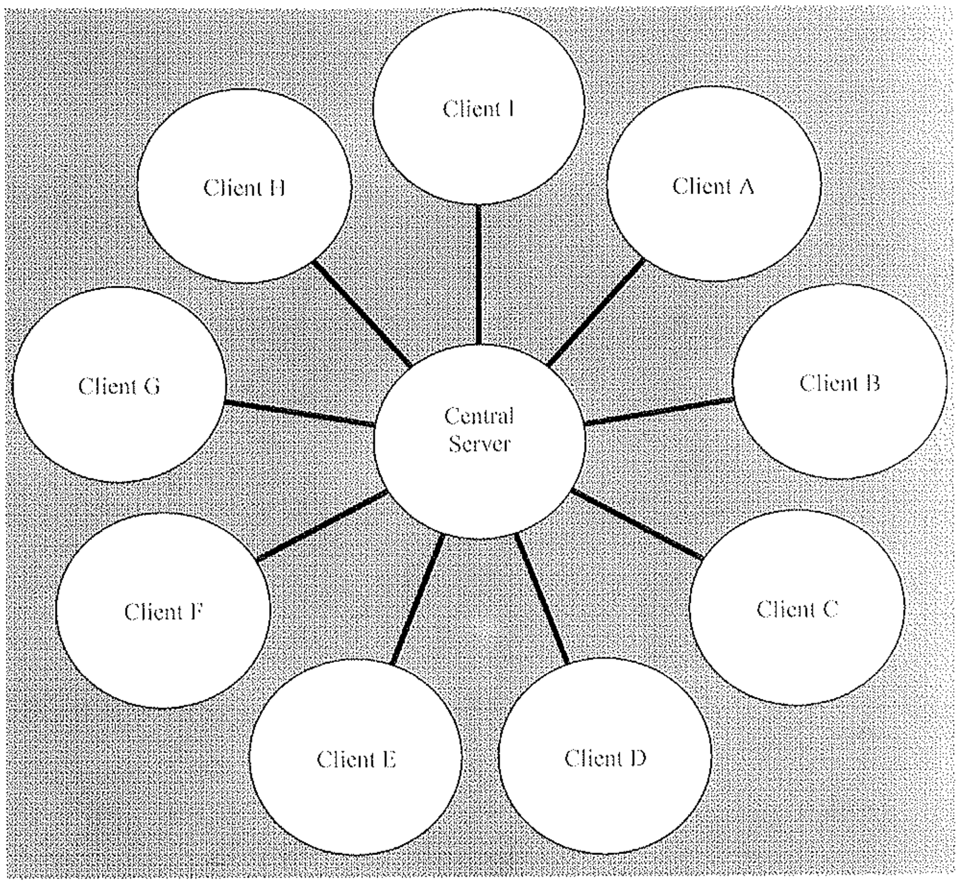


Figure 16 Abstract P2P Diagram

The database was kept at a central server. All the transactions were made on the distributed database. So there were no problems of synchronization. Following is the architecture diagram of the CTF game.

3.1 ARCHITECTURE DIAGRAM

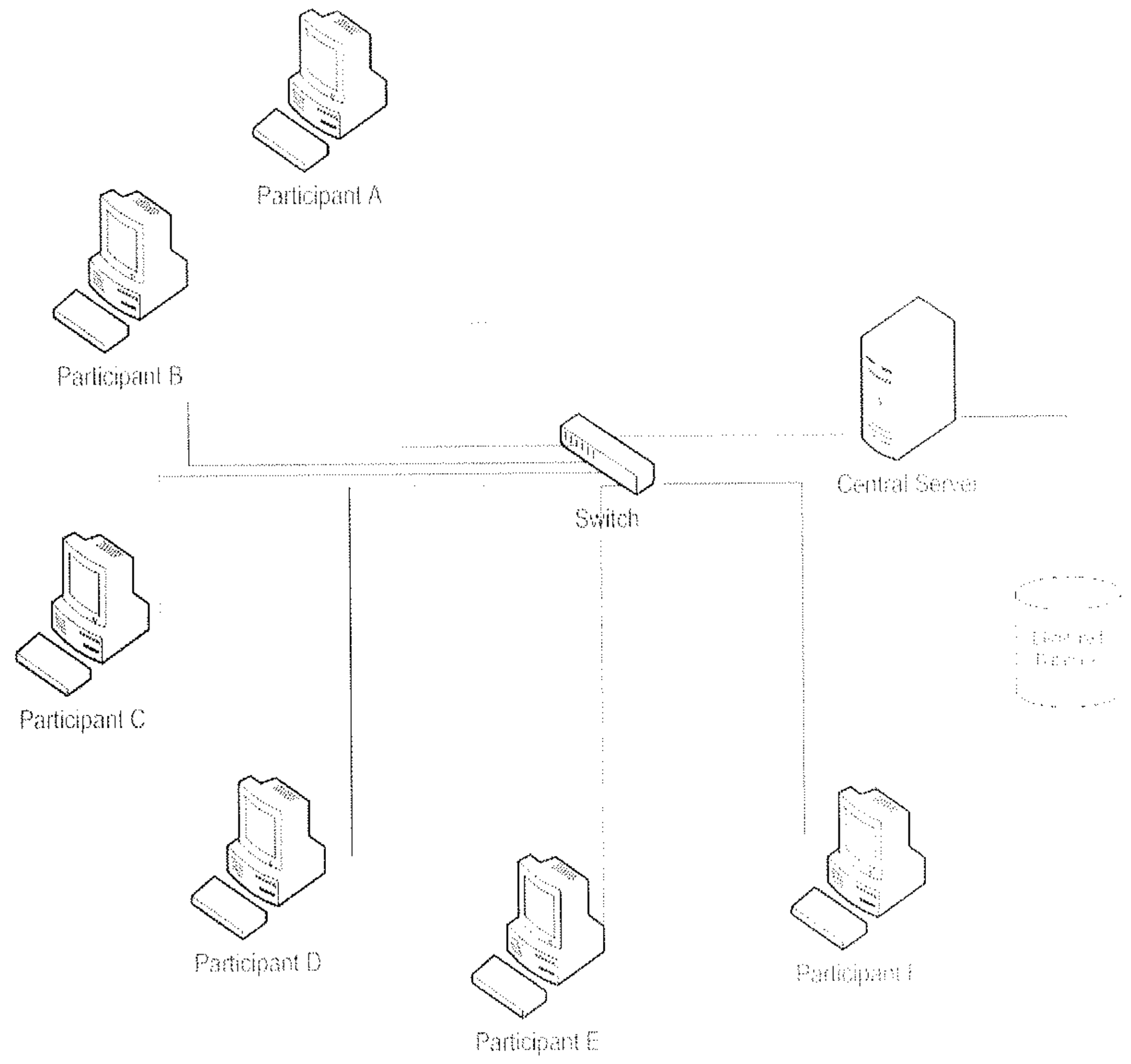


Figure 17 Architecture Diagram

According to this architecture the central server has to take care of the database. The central server maintains information about all the available clients and keeps them up to date in case of any change. All the transactions take place at the server secondly the server is keeping track of the clients participating. When a client does transaction it asks for a list of participants from the server that's how the clients know who are the other participants.

3.2 CLASS DIAGRAMS

3.2.1 CTF Server Class Diagram

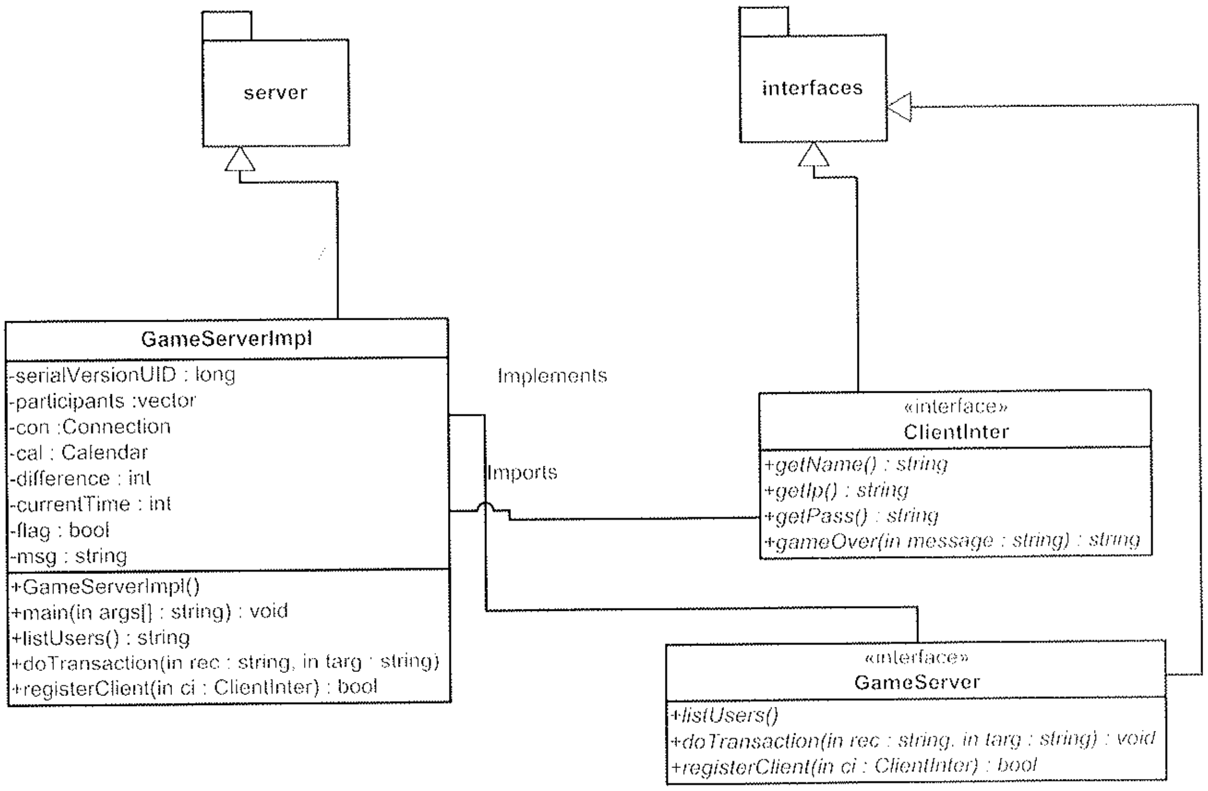


Figure 18 Server Class Diagram

3.2.2 CTF Client Class Diagram

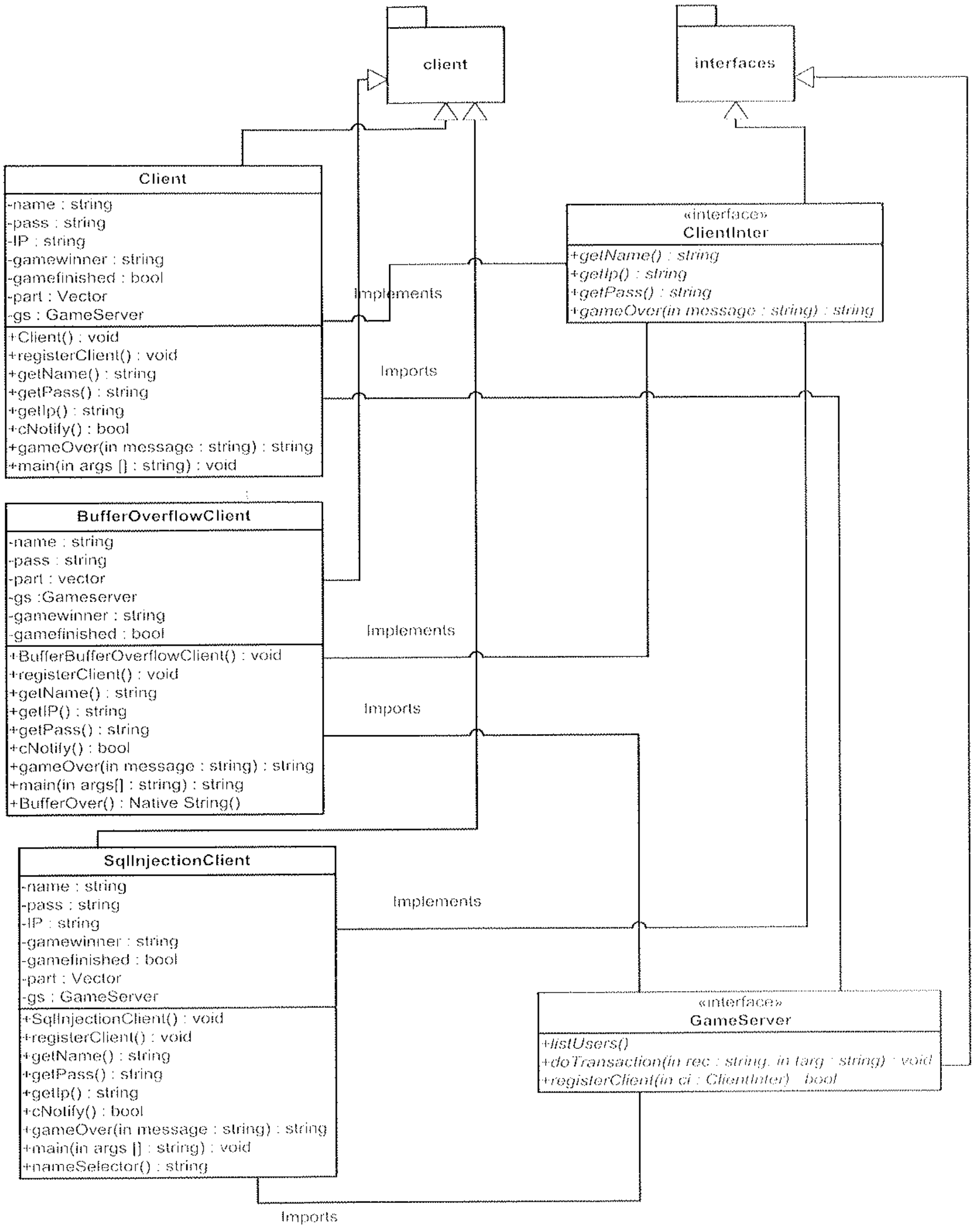


Figure 19 Client Class Diagram

3.3 FLOW DIAGRAMS

3.3.1 CTF Game Server Flow Diagram

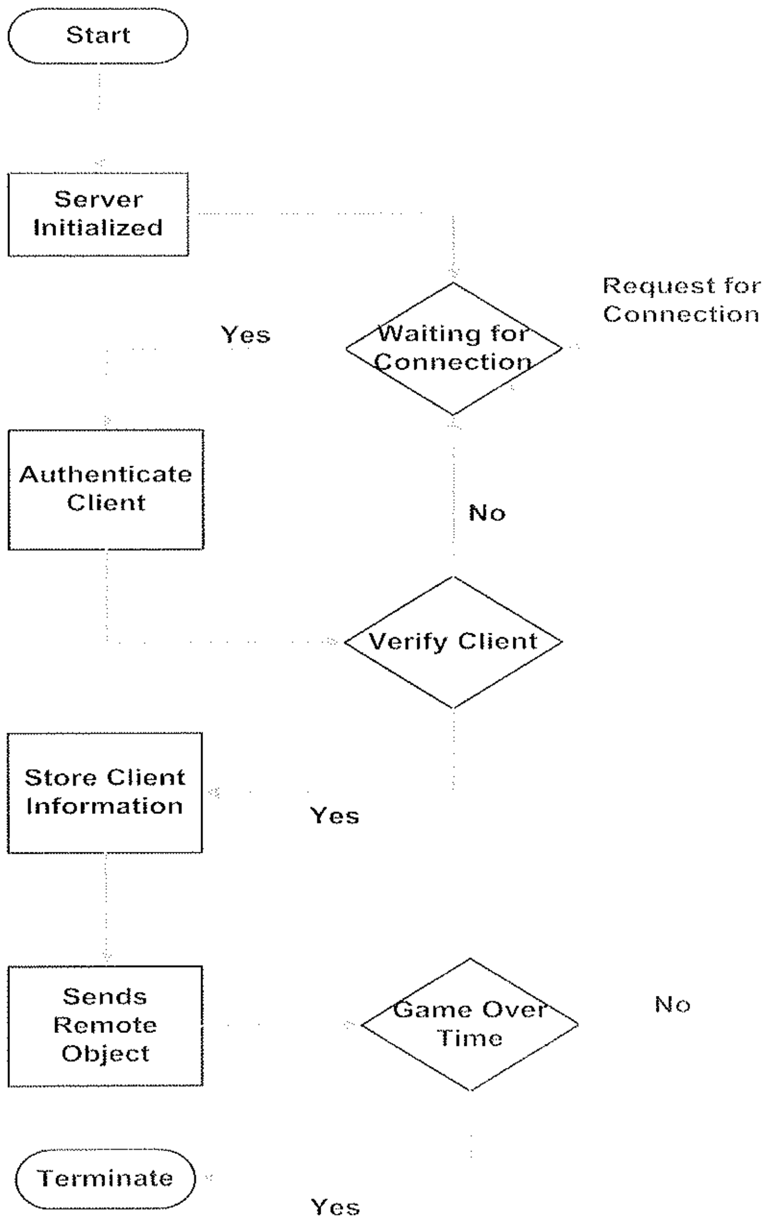


Figure 20 Server Flow Diagram

The server first initializes its self when it starts like hosting the service on a particular port. Then the server goes into the state of listening where the server waits for clients. If a participant tries to connect to the server it ask for a valid user name and a password. If the user has provided correct password and user name, the client gets registered with the server and all the connected clients are notified of the new participant. After the client is registered the server sends a remote object to the client so that it can start its transactions from the other clients.

The client first initializes its ports, and make a stub which serve as the client reference with the server as well as with the other participants. The client first searches for the server. If a server object is found in the remote registry it sends its imports the object and tries to register by sending its own stub to the server the server takes the username and password and sends it for verification if the client is verified it is registered at the server and information is passed to other clients as well. The client participates in the game until it receives an interrupt from the server which is send after a particular period. The working of the client can be understood from the following figure.

3.3.2 CTF Game Client Flow Diagram

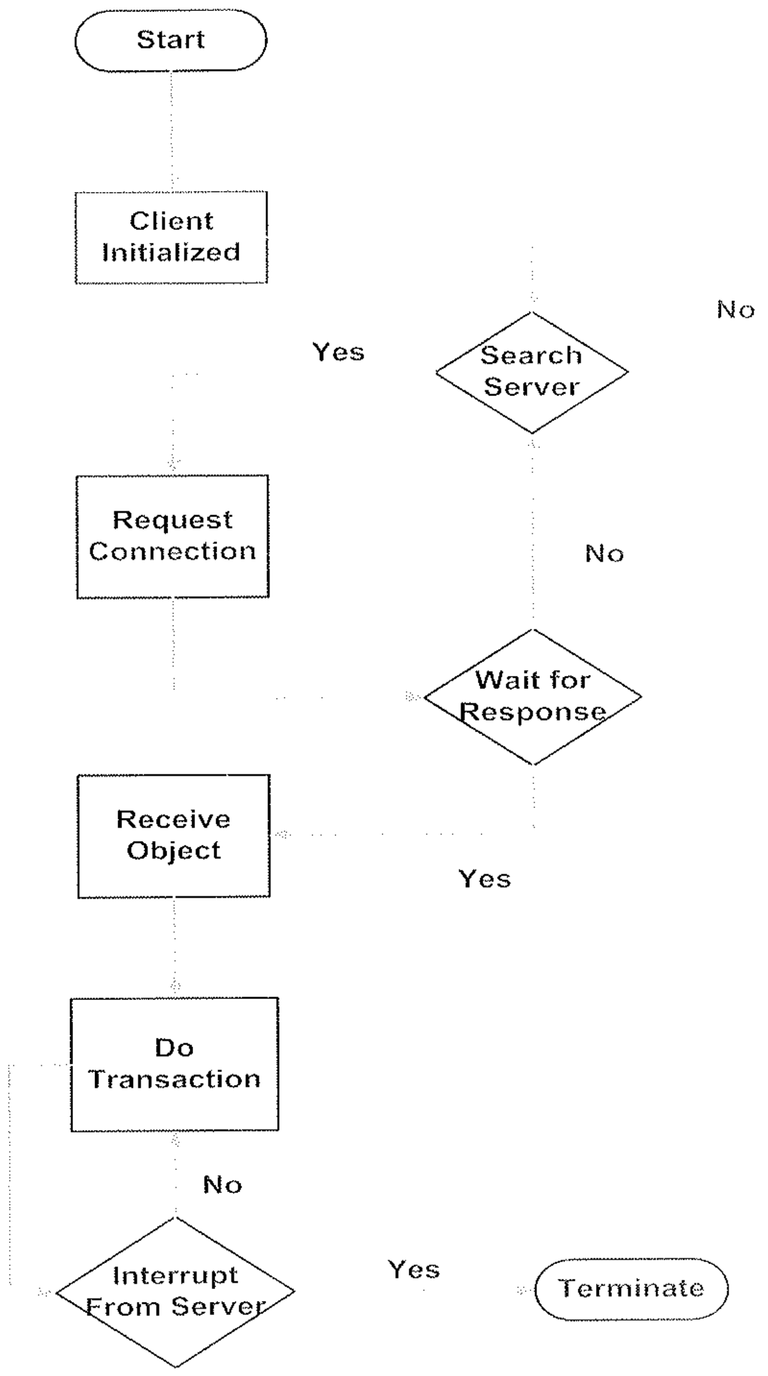


Figure 21 Client Flow Diagram

3.3.3 Sequence Diagram

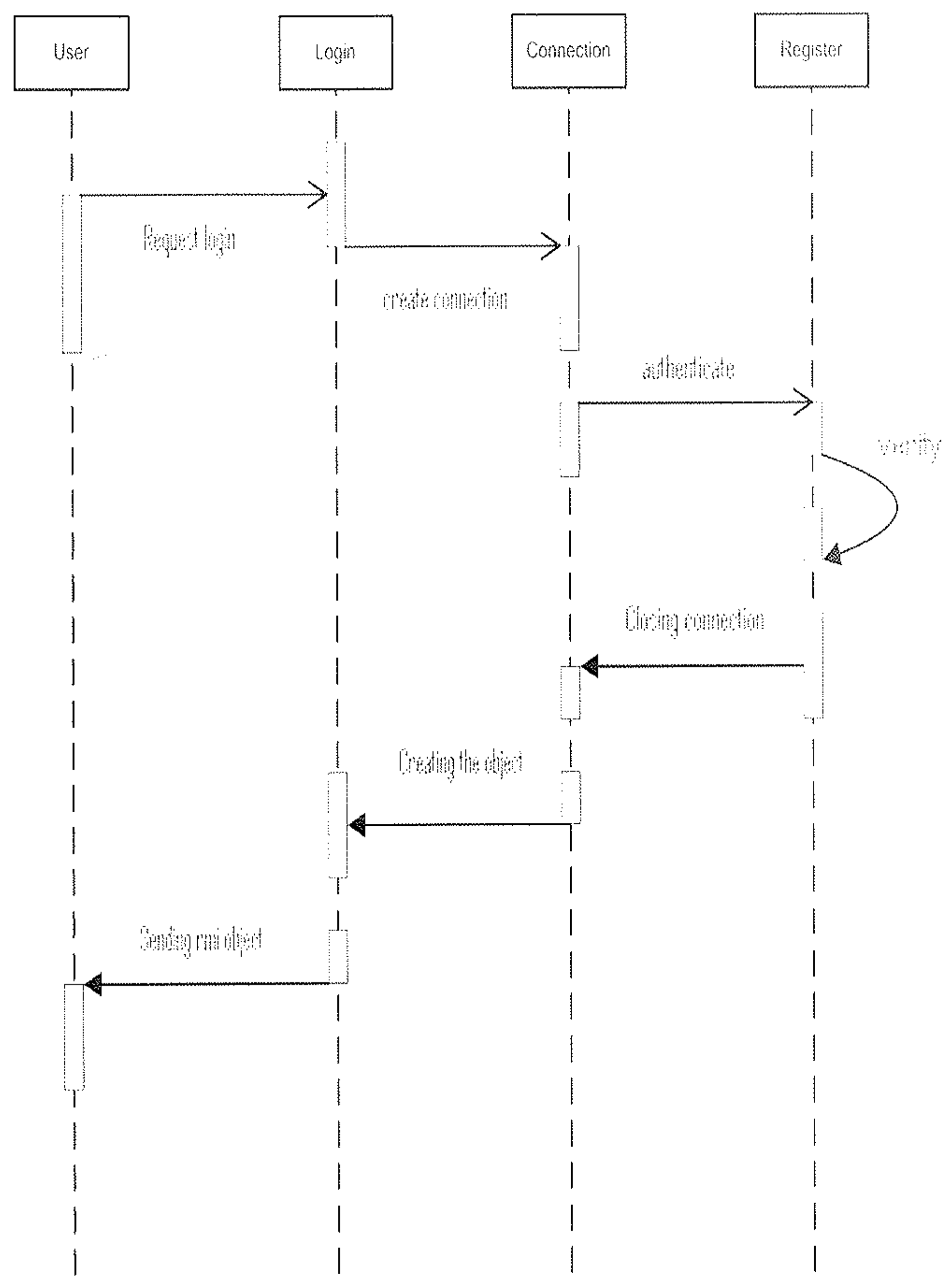


Figure 22 Sequence Diagram

3.3.4 Use Case Diagram

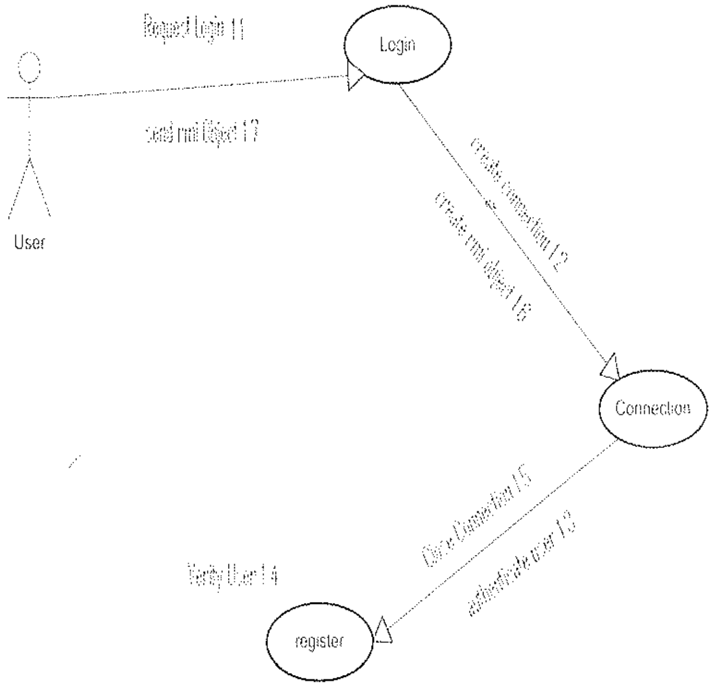


Figure 23 Use case Diagram

Chapter 4

IMPLEMENTATION

4.1 IP SPOOFING AND SYN-FLOODING

As stated earlier java don't allow accessing bellow application layer and it do not allow forging source or destination IP addresses in the packet so in order to demonstrate IP spoofing and SYN-Flooding a separate example was implemented using RAW sockets. As RAW sockets are disabled in WINXP SP2 this code can only be executed on a Linux platform. Before sending the packet to the destination the IP of the source is forged and then the packet is send with the SYN flag set so the server sends the SYN-ACK to the forged IP address. In order to accomplish SYN-Flooding IP spoofing is not necessary but is useful in order to hide the attacker's identity

We can prevent IP spoofing by using filter routers which will not allow packets which has the IP of your internal network through external interface and outgoing packet with source IP different from your internal network.

1. One way is to check if we have a lot of SYN-received state
2. Using UPC (usage parameter control) mechanism or policing
3. By controlling the traffic, preventing congestion control the arrival rate of packets according to the status of backlog.
4. By increasing the size of the connection table
5. By making the timcout for connection shorter

4.2 SQL INJECTION

But as the game is a Fight for resources so it would be better to change the transaction query in order gain benefit. The transaction is composed of two query one which adds 10 resources to the client and deduct 10 resources from the target client. The query which deducts resources is as following.

Statement: UPDATE items SET items.numberof item (items.numberof item-10) "WHERE items.user id in (select login.user id from login where user name like '"+target+"') ;");

Now the query is supposed to take on target client to deduct the resources. What happens is the sql injection client sum up all the clients currently playing the game. The input is provided in such a way that instead of deducting resources from one client it subtracts resources from multiple clients. Suppose 5 clients including Nauman are playing the game Nauman is the one running the sql injection client. The input to the server would be like:

Shumaila' or user name like 'Fauzan' or user name like 'Ali ' or user name like 'Qasim: After that input the query becomes:

Statement: UPDATE items SET items.numberof item (items.numberof item-10) "WHERE items.user id in (select login.user id from login where user name like ' Shumaila' or user_name like 'Fauzan'or user_name like 'Ali ') ;");

The inner select return multiple records which are then updated by the update Query. We can also run multiple queries in a single command on some databases. Like mysql, oracle etc. like "select * from login; drop table users;" But due to limitations in java such kind of query won't execute which returns multiple result sets or number for updated rows.

The best way to prevent SQL injection is to check the query for escape sequences or the use of parameterized queries. In parameterized queries the user input is not directly fed into the query. Enforcing parameterized queries means to avoid the chances that the user input has embedded code for SQL and they can be rejected at run time. The second method is to check the code manually like using code review in order to cater this vulnerability.

4.3 BUFFER OVER FLOW

As we have mutable arrays in java and java make sure that user is not allowed to access the memory directly so there are few chances to make a buffer overflow.

Due to this limitation of java language the only way to demonstrate a buffer overflow attack was to use JNI. Through the utility of JNI we can call any native language code in java. So I have made a C++ method which takes a String which an IP address of the client checks it by coping it into an unsigned char array. The array size is to accommodate a normal ipv4 address. A separate buffer overflow client is made which sends an arbitrary data to the client, which exceeds the limit of that buffer. Now when the code tries to copy the huge data in the buffer it overflows. As the exception is thrown outside the JVM so it cannot handle the problem. In order to know what problem has occurred it auto generates a log file which logs all the information related to the cause of buffer overflow.

There are some counter measures to avoid the buffer overflow with several tradeoffs. The efficient way to avoid buffer overflow problem is to use automatic protection at the programming level. One way is to use safe libraries which have checks for bound exceptions. Like strlcpy () make sure that there is no stack overflow rather than using strcpy (). Other way is to use stack smashing protection which checks the stack that if a function returns with the stack altered the program terminates with a segmentation fault. There are many other ways to check for buffer overflows like executable space protection, address space layout randomization, and deep packet inspection etc.

Chapter 5

RESULTS

Over all this project was a great learning experience for me. I can now say that i have quite an experience working in java and C/C++. While making this project i encountered many challenges which were so vague in the begining that i thought they were impossible but checking things on many forums and informative sites helped me understand those problems. Most of the sites don't give any information about working at link layer or below TCP layer the reason is the doubt that the user might use this information for malicious activities. I have come up with four of the vulnerabilities from the attacks that were in consideration. These implementation will allow any individual to understand how these attacks are made and how they exploits different services. I have provided solutions to detect or to prevent these attacks. Wheih can help the network administrators to make their autonomous network secure from these attacks. Most of these attacks are not applicable nowadays like SYN-flooding due to the efficient scheduling of the stack in modern OSs but they of much more importance as they cover all the basics to make a new attack.

Chapter 6

CONCLUSION

There is a large potential for the introduction of network security in many fields of networking. All the organizations wants their data to be secure and the integrity of their data should be mantained under all circumstances.

The project provided a hybrid P2P solution for the CTF game on much more secure language java, in which most of the attacks are already taken care off. But the project provides sufficient information about those and how they can exploit an application in java language. Some of the attacks are still implemented in C/C++ which have a lot of vulnerable libraries with no bound checking measures.

The project provided a CTF game developed using RMI (remote method invocation), rmi is supposed to be much more secure. The project provide some very highly significant case studies on how loop holes can be exploited in Java RMI.

Chapter 7

RECOMMENDATIONS

As I have chosen Java to be the implementation language so there were a lot of limitation in implementing those attacks what I will suggest that in future this project can be made in C/C++ where these and many other attacks can be implemented. Like TCP-hijacking, SYN-flooding etc. which were not implemented due to lack of time.

These attacks will provide suitable case studies for the network administrators to have a strong grip over their network.

Chapter 8

Deliverable(s)

Multi-player e-gamming environment on a java platform with a vulnerable network which includes the following

1. Game Logical Design
2. Game Physical Design
    a) Client-Side GUI
    b) Server services
    c) Identifying and creating loop holes in the network
3. Implementation
    a) Client implementation
    b) Server implementation
4. Test beds for testing network vulnerability in relation with some of the following aspects:
    a) ARP Poisoning
    b) IP Spoofing
    c) SQL Injections
    d) Shell Code
    e) File Format
    f) TCP Hijacking
    g) Buffer overflow
    h) SYN Flooding
5. Analysis and comparison of the test results.
6. Documentation

8.1 SCHEDULE OF DELIVERABLES

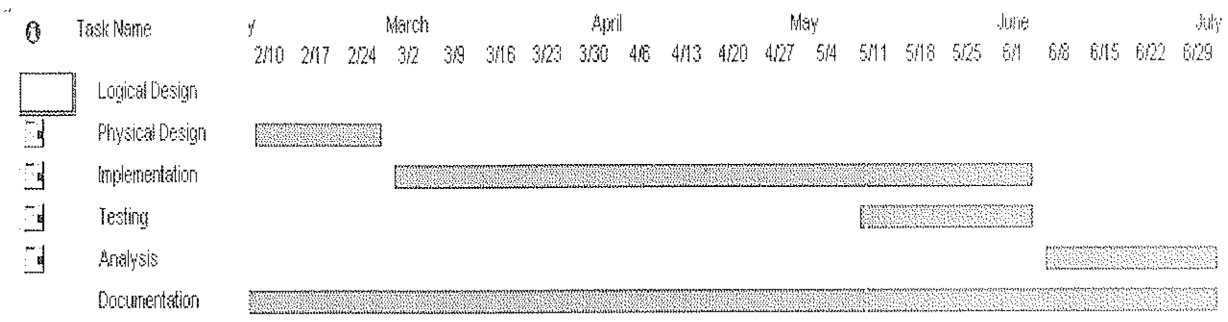


Figure 24 Deliverables

References

[1] WikiPedia : Capture the Flag Game
http://en.wikipedia.org/wiki/Capture_the_flag

[2] Watch Guard:ARP-Poisoning
http://www.watchguard.com/infocenter/editorial/135324.asp

[3] CERTAdvisory:IP-Spoofing
http://www.cert.org/advisories/CA-1995-01.html

[4] CERTAdvisory:SYN-Flooding
http://www.cert.org/advisories/CA-1996-21.html

[5] Internet Security Systems:SYN-Flooding
http://www.iss.net/security_center/advice/Exploits/TCP/SYN_flood

[6] WikiPedia: BufferOverflow
http://en.wikipedia.org/wiki/Buffer_overflow

[7] Search Security: BufferOverflow
http://searchsecurity.techtarget.com/

[8] OWASP: BufferOverflow
http://www.owasp.org/index.php/Buffer_Overflow

[9] Security Team; SQL Injection
http://www.securiteam.com/securityreviews/5DP0N1P76E.html

[10] Watch guard: Anatomy of ARP poisoning attack
http://www.watchguard.com/infocenter/editorial/135324.asp

[11] Wikipedia: Capture The Flag
http://en.wikipedia.org/wiki/Capture_the_flag

[12] watch guard: Session hijacking
http://www.watchguard.com/glossary/s.asp#session_hijacking

[13] Search Security: File Format Vulnerabilities
http://searchsecurity.techtarget.com/tip

[14] Information Security and Security Architecture: Code Injection
http://209.85.129.104/search?q=cache:aYwbZ6i5_IEJ:www.hig.no/index.php

[15] Justin ma, john dunagan, j Wang : finding diversities of remote code
http://www.slideshare.net/amiable_indian/finding-diversity-inremote-code-*injection-exploits/*

[16] Matt Conover, Oded Horvitz : Reliable Windows Heap Exploits
http://www.slideshare.net/amiable_indian/reliable-windows-heap-exploits/

[17] Doug Seven Microsoft MVP : Hackers Paradise SQL Injection
http://www.slideshare.net/amiable_indian/hackers-paradise-sql-injection-attacks/

[18] Tech Republic: useful information on different topics
http://images.google.com.pk/imgres?imgurl=http://articles.techrepublic.com.com

[19] Nice Scripts. Net: SQL Injections and many other information
http://nicescript.net/ebooks-hacker-nulled-script/hacking-the-art-of-exploitation-*2nd-edition.html*

[20] Unix Local Attacks and how Buffer/Stack Overflow is done in Unix
http://www.nmrc.org/pub/faq/hackfaq/hackfaq-29.html

[21] A complete Buffer overflow example for a Unix platform
http://insecure.org/stf/mudge_buffer_overflow_tutorial.html

[22] windows security related articles
http://www.windowsecurity.com/

[23] Search Security: Network Security Tactics
http://searchsecurity.techtarget.com/tip/0,289483,sid14_gci1243042,00.html

[24] Session Hijacking Complete step wise example
http://staff.washington.edu/dittrich/talks/qsm-sec/script.html

[25] HTML hacking Scripts
http://www.cpan.org/authors/id/TOMC/scripts/html-hacking.html

[26] Maximum Security by sams.net: A hackers guide for protecting internet
http://docs.rinet.ru/LomamVse/ch28/ch28.htm

[27] ARP poisoning Image
http://docs.rinet.ru/LomamVse/ch28/ch28.htm

[28] ARP poisoning image
http://www.oxid.it/downloads/apr-intro.swf

[29] ARP poising image
http://www.oxid.it/downloads/apr-intro.swf

[30] ARP poisoning Image
http://www.oxid.it/downloads/apr-intro.swf

[31] IP Spoofing Image
http://www.jaist.ac.jp/~nakagawa/jaist/sub/fig10.gif

[32] TCP Hijacking Image
http://articles.techrepublic.com.com/5100-10878_11-5033594.html

[33] SYN Flooding Image
http://en.wikipedia.org/wiki/SYN_flood

[34] Buffer Overflow Image
http://www.newscientist.com/data/images/ns/cms/dn4696/dn4696-1_547.jpg

[35] Buffer Overflow Image
http://www.soi.wide.ad.jp/soi-asia/pkg1/07/img/36.png

[36] Format Vulnerability Image
http://vil.nai.com/images/avert_blog_exploit_flow.gif

[37] Shell Code Formulation Image
http://vx.netlux.org/lib/img/aps00/Figure09_12.gif

[38] Shell Code Formulation Image
http://vx.netlux.org/lib/img/aps00/Figure09_11.gif

[39] Shell Code Formulation
http://vx.netlux.org/lib/img/aps00/Figure10_13.gif