

UNIT - I

SECURITY ATTACKS (INTERRUPTION, INTERCEPTION, MODIFICATION AND FABRICATION), SECURITY SERVICES (CONFIDENTIALITY, AUTHENTICATION, INTEGRITY, NON-REPUDIATION, ACCESS CONTROL AND AVAILABILITY)

AND MECHANISMS, A MODEL FOR INTERNETWORK SECURITY, INTERNET STANDARDS AND RFCs, BUFFER OVERFLOW & FORMAT STRING VULNERABILITIES, TCP SESSION HIJACKING, ARP ATTACKS, ROUTE TABLE MODIFICATION, UDP HIJACKING, AND MAN-IN-THE-MIDDLE ATTACKS.

Introduction:

This is the age of universal electronic connectivity, where the activities like hacking, viruses, electronic fraud are very common. Unless security measures are taken, a network conversation or a distributed application can be compromised easily.

Some simple examples are:

- Online purchases using a credit/debit card.
- A customer unknowingly being directed to a false website.
- A hacker sending a message to a person pretending to be someone else.

Information security has been affected by two major developments over the last several decades. First one is introduction of computers into organizations and the second one being introduction of distributed systems and the use of networks and communication facilities for carrying data between users & computers. These two developments lead to 'computer security' and 'network security', where the computer security deals with collection of tools designed to protect data and to thwart hackers. Network security measures are needed to protect data during transmission. But keep in mind that, it is the information and our ability to access that information that we are really trying to protect and not the computers and networks.

Information Security: It can be defined as "measures adopted to prevent the unauthorized use, misuse, modification or denial of use of knowledge, facts, data or capabilities". Three aspects of IS are:

- **Security Attack:**
Any action that comprises the security of information
- **Security Mechanism:**
A mechanism that is designed to detect, prevent, or recover from a security.
- **Security Service:**
It is a processing or communication service that enhances the security of the data processing systems and information transfer. The services are intended to counter

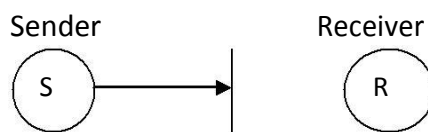
security attacks by making use of one or more security mechanisms to provide the service.

Security Attacks

Security attacks can be classified in terms of Passive attacks and Active attacks as per X.800 and RFC 2828

Different kinds of attacks are:

Interruption

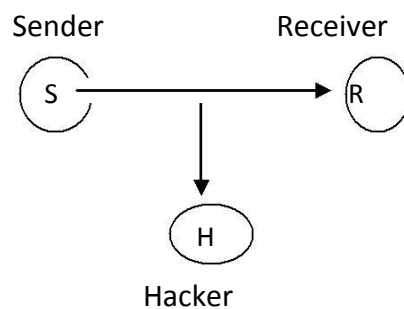


An asset of the system is destroyed or becomes unavailable or unusable. It is an attack on availability.

Examples:

- Destruction of some hardware
- Jamming wireless signals
- Disabling file management systems

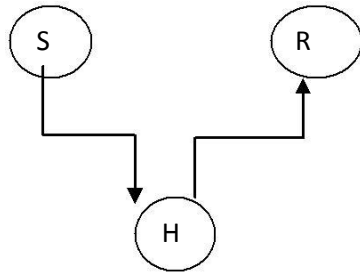
Interception



An unauthorized party gains access to an asset. Attack on confidentiality.

Examples:

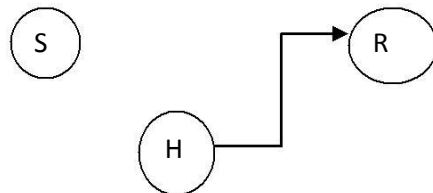
- Wire tapping to capture data in a network.
- Illicitly copying data or programs
- Eavesdropping

Modification:

When an unauthorized party gains access and tampers an asset. Attack is on Integrity.

Examples:

- Changing data file
- Altering a program and the contents of a message

Fabrication

An unauthorized party inserts a counterfeit object into the system. Attack on Authenticity. Also called impersonation

Examples:

- Hackers gaining access to a personal email and sending message
- Insertion of records in data files
- Insertion of spurious messages in a network

Passive Attacks

A Passive attack attempts to learn or make use of information from the system, but does not affect system resources.

Two types:**Release of message content**

It may be desirable to prevent the opponent from learning the contents (i.e sensitive or confidential info) of the transmission.

→ Traffic analysis

A more subtle technique where the opponent could determine the location and identity of communicating hosts and could observe the frequency & length of encrypted messages being exchanged there by guessing the nature of communication taking place.

Passive attacks are very difficult to detect because they do not involve any alternation of the data. As the communications take place in a very normal fashion, neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. So, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

Active attacks involve some modification of the data stream or creation of a false stream. An active attack attempts to alter system resources or affect their operation.

Four types:

→ **Masquerade:** Here, an entity pretends to be some other entity. It usually includes one of the other forms of active attack.

→ **Replay:** It involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

→ **Modification of messages:** It means that some portion of a legitimate message is altered, or that messages are delayed to produce an unauthorized effect.

Ex: "John's acc no is 2346" is modified as "John's acc no is 7892"

→ **Denial of service:** This attack prevents or inhibits the normal use or management of communication facilities.

Ex: a: Disruption of entire network by disabling it

b: Suppression of all messages to a particular destination by a third party.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because of the wide variety of potential physical, software and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them.

Security Services:

It is a processing or communication service that is provided by a system to give a specific kind of production to system resources. Security services implement security policies and are implemented by security mechanisms.

→ Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. It is used to prevent the disclosure of information to unauthorized individuals or systems. It has been defined as “ensuring that information is accessible only to those authorized to have access”.

The other aspect of confidentiality is the protection of traffic flow from analysis. **Ex:** A credit card number has to be secured during online transaction.

→ Authentication

This service assures that a communication is authentic. For a single message transmission, its function is to assure the recipient that the message is from intended source. For an ongoing interaction two aspects are involved. First, during connection initiation the service assures the authenticity of both parties. Second, the connection between the two hosts is not interfered allowing a third party to masquerade as one of the two parties. Two specific authentication services defines in X.800 are

- **Peer entity authentication:** Verifies the identities of the peer entities involved in communication. Provides use at time of connection establishment and during data transmission. Provides confidence against a masquerade or a replay attack
- **Data origin authentication:** Assumes the authenticity of source of data unit, but does not provide protection against duplication or modification of data units. Supports applications like electronic mail, where no prior interactions take place between communicating entities.

→ Integrity

Integrity means that data cannot be modified without authorization. Like confidentiality, it can be applied to a stream of messages, a single message or selected fields within a message. Two types of integrity services are available. They are

- **Connection-Oriented Integrity Service:** This service deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering or replays. Destruction of data is also covered here. Hence, it attends to both message stream modification and denial of service.
- **Connectionless-Oriented Integrity Service:** It deals with individual messages regardless of larger context, providing protection against message modification only.

An integrity service can be applied with or without recovery. Because it is related to active attacks, major concern will be detection rather than prevention. If a violation is detected and the service reports it, either human intervention or automated recovery machines are required to recover.

→ **Non-repudiation**

Non-repudiation prevents either sender or receiver from denying a transmitted message. This capability is crucial to e-commerce. Without it an individual or entity can deny that he, she or it is responsible for a transaction, therefore not financially liable.

→ **Access Control**

This refers to the ability to control the level of access that individuals or entities have to a network or system and how much information they can receive. It is the ability to limit and control the access to host systems and applications via communication links. For this, each entity trying to gain access must first be identified or authenticated, so that access rights can be tailored to the individuals.

→ **Availability**

It is defined to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity. The availability can significantly be affected by a variety of attacks, some amenable to automated counter measures i.e authentication and encryption and others need some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

Security Mechanisms:

According to X.800, the security mechanisms are divided into those implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. X.800 also differentiates reversible & irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted, where as irreversible encipherment include hash algorithms and message authentication codes used in digital signature and message authentication applications

→ **Specific Security Mechanisms:**

Incorporated into the appropriate protocol layer in order to provide some of the OSI security services,

- **Encipherment:** It refers to the process of applying mathematical algorithms for converting data into a form that is not intelligible. This depends on algorithm used and encryption keys.
- **Digital Signature:** The appended data or a cryptographic transformation applied to any data unit allowing to prove the source and integrity of the data unit and protect against forgery.
- **Access Control:** A variety of techniques used for enforcing access permissions to the system resources.
- **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

- **Authentication Exchange:** A mechanism intended to ensure the identity of an entity by means of information exchange.
- **Traffic Padding:** The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
- **Routing Control:** Enables selection of particular physically secure routes for certain data and allows routing changes once a breach of security is suspected.
- **Notarization:** The use of a trusted third party to assure certain properties of a data exchange

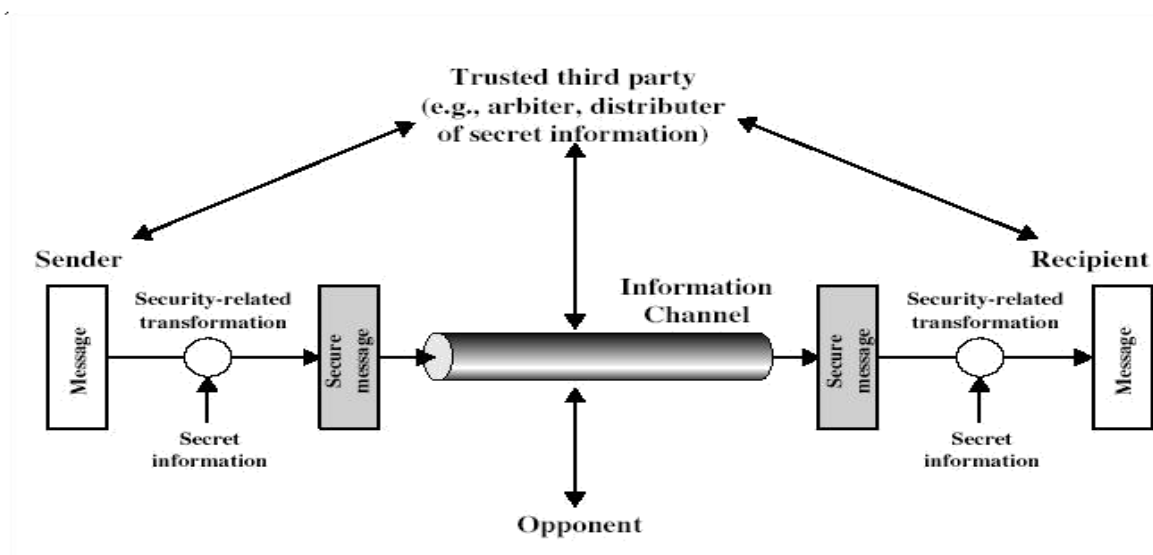


Pervasive Security Mechanisms:

These are not specific to any particular OSI security service or protocol layer.

- **Trusted Functionality:** That which is perceived to be correct with respect to some criteria
- **Security Level:** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
- **Event Detection:** It is the process of detecting all the events related to network security.
- **Security Audit Trail:** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
- **Security Recovery:** It deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

A Model Of Inter Network Security



Data is transmitted over network between two communicating parties, who must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination by use of communication protocols by the two parties. Whenever an opponent presents a threat to confidentiality,

authenticity of information, security aspects come into play. Two components are present in almost all the security providing techniques.

- A security-related transformation on the information to be sent making it unreadable by the opponent, and the addition of a code based on the contents of the message, used to verify the identity of sender.
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception

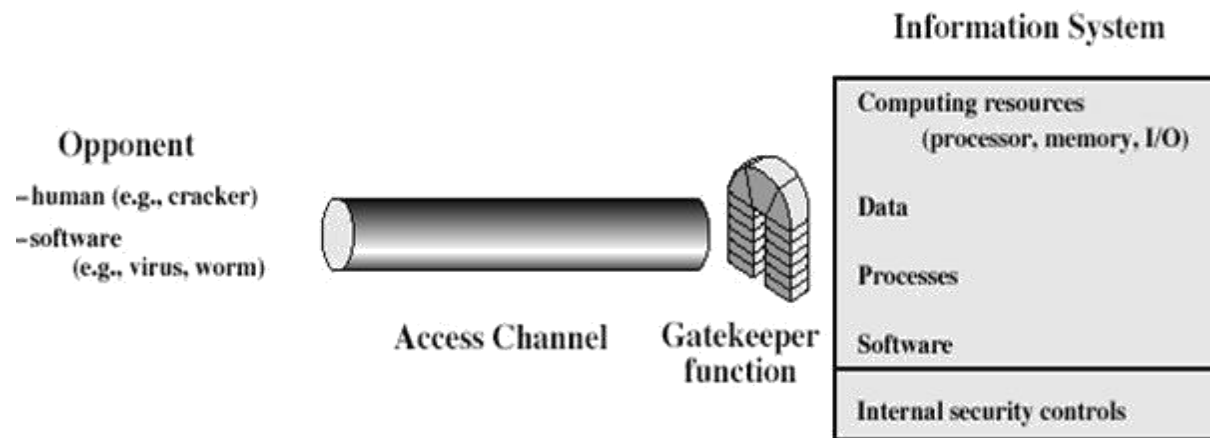
A trusted third party may be needed to achieve secure transmission. It is responsible for distributing the secret information to the two parties, while keeping it away from any opponent. It also may be needed to settle disputes between the two parties regarding authenticity of a message transmission. The general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose
2. Generate the secret information to be used with the algorithm
3. Develop methods for the distribution and sharing of the secret information
4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service

Various other threats to information system like unwanted access still exist. The existence of hackers attempting to penetrate systems accessible over a network remains a concern. Another threat is placement of some logic in computer system affecting various applications and utility programs. This inserted code presents two kinds of threats.

- **Information access threats** intercept or modify data on behalf of users who should not have access to that data
- **Service threats** exploit service flaws in computers to inhibit use by legitimate users

Viruses and worms are two examples of software attacks inserted into the system by means of a disk or also across the network. The security mechanisms needed to cope with unwanted access fall into two broad categories



- Placing a gatekeeper function, which includes a password-based login methods that provide access to only authorized users and screening logic to detect and reject worms, viruses etc
- An internal control, monitoring the internal system activities analyzes the stored information and detects the presence of unauthorized users or intruders.

Internet Standards and RFC'S

Most of the protocols related to TCP/IP protocol suite are already standardized or under the process of standardization. An organization known as internet society is responsible for development and publication of these standards. It is the actually a professional membership organization that supervises a large in internet development and standardization

An internet society refers to the organization responsible for monitoring and coordinating internet design, engineering and management. Three organizations under the internet society are responsible for actual work of standards development & publication

- 1. INTERNET ARCHITECTURE BOARD (IAB):** Responsible for defining the overall architecture of the internet, providing guidance and broad direction to IETF
- 2. INETRNET ENGINEERING TASK FORCE (IETF):** The protocol engineering and development arm of the internet
- 3. INTERNET ENGINEERING STEERING GROUP (IESG):** Responsible for technical management of IETF activities and the internet standards process

Working groups chartered by IETF carry out actual development of new standards and protocols for the internet as membership is voluntary; any party can enter into working group will make a draft version made available as an internet draft placed in IETF's "internet drafts" online directory. This will remain up to six months, where interested parties may review & comment on it. During this time, IESG may approve the draft as an RFC or else it is withdrawn from directory, and a revised edition is published.

The IETF is responsible for publishing the RFC'S with approval of IESG. The RFC'S are working notes of the internet research and development community. The entire activities of the IETF are categorized into eight areas each having a categorized into eight areas each having it & numerous working groups

IETF Area	Theme	Example Working Groups
General	IETF process and procedures	Policy framework Process for organization of Internet standards
Applications	Internet applications	Web-related protocols (HTTP) EDI-Internet integration LDAP
Internet	Internet infrastructure	IPv6 PPP extensions
Operations and management	Standards and definitions for network operations	SNMPv3 Remote network monitoring
Routing	Protocols and management for routing information	Multicast routing OSPF QoS routing
Security	Security protocols and technologies	Kerberos IPSec X.509 S/MIME TLS
Transport	Transport-layer protocols	Differentiated services IP telephony NFS RSVP
User services	Methods to improve the quality of information available to users of the Internet	Responsible use of the Internet User services FYI documents

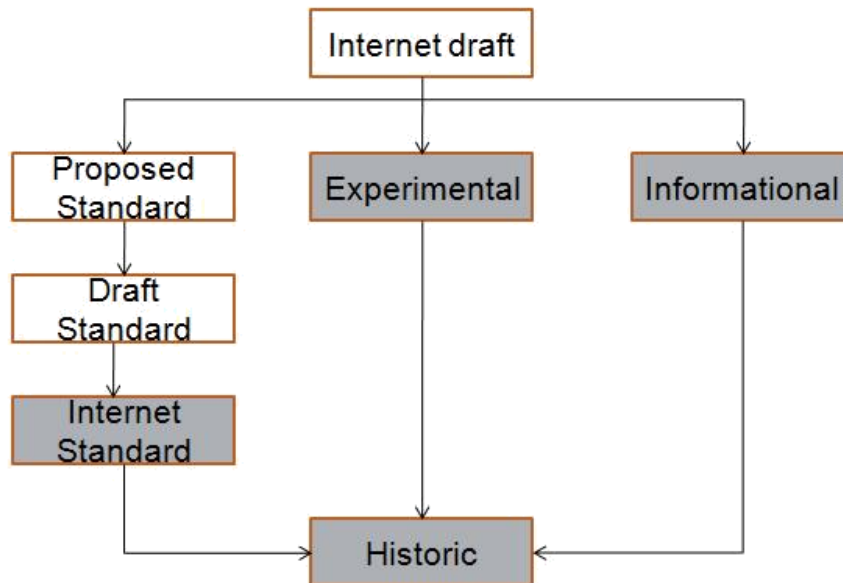
The Standardization Process:

IESG decides which RFC's become internet standard based on IETF recommendations. To become a standard, a specification must meet the following criteria.

- BE stable and easily understandable
- Be technically competent

- Have multiple, independent and interoperable implementations with substantial operations experience.
- Enjoy significant public support.
- Be recognizably useful in some or all parts of internet

The RFC publication process is shown below, in which a specification passes through a sequence of steps called standards track, in order to qualify as a standard. It involves excessive scrutinizing and testing. The actual process starts after the approval of internet draft documentation as an RFC by IESG.



For a specification to act as a draft standard it must pass through at least two non- dependent interoperable implementations for achieving proper operational experience once, necessary implementations and operational experience is achieved, it can be regarded as internet standard. Now, this specification is equipped with two numbers, an STD number and an RFC number .Finally, when a protocol becomes outdated, it is assigned to the historic state.

Internet Standard Categories

All the internet standards fall into two categories

- ➔ **TECHNICAL SPECIFICATION (TS):** TS defines a protocol, service, procedure, convention or format. Most internet standards are TS's.
- ➔ **APPLICABILITY STATEMENT (AS):** AS specifies how, and under what circumstances, one or more TS may be applied to support a particular internet capability. It identifies one or more TS's that are relevant to the capability and may specify values or ranges for particular parameters associated with a TS or functional subsets of a TS that are relevant for the capability.

Other RFC Types

Some RFC's exist that can standardize the results of community deliberations regarding the best way to perform some operations or IETF process function. These are known as best current practices (BCP) whose approval process is similar and it's a one stage process. A protocol or other specification that is not considered ready for standardization may be published as an experimental RFC and after reworking on it, submitted again and when it has resolved known design choices, is believed to be well understood, has received significant community review and has got good public community interest to be considered valuable, their RFC will be designated a proposed standard. Finally, an informational specification is published for general information of internet community.

Buffer Overflow & Format String Vulnerabilities

Vulnerability: Vulnerability is an inherent weakness in design, configuration, implementation or management of a network or system that renders it susceptible to a threat. Vulnerabilities are what make networks susceptible to information loss and downtime. Every network and system has some kind of vulnerability.

Buffer Overflow: A buffer overflow occurs when a program or process tries to store more data in a buffer than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information can overflow into adjacent buffers, corrupting or overwriting the valid data held in them. Though this may occur accidentally because of a programming error, at present it is an increasingly common type of security attack on integrity.

It happens when the attacker intentionally enters more data than a program was written to handle. The data runs over and overflows the section of valid data like part of programming instructions, user files, confidential information etc there by enabling the attacker's data to overwrite it. This allows an attacker to overwrite data that controls the program and can take over control of the program to execute the attacker's code instead of programmer's code.

Exploiting the overflowable buffer involves the following tasks

- Finding a way of injecting into the buffer
- Specify a return address where malicious code resides for the program to execute the code
- Determining the payload/code to be executed

Buffer Injection Techniques

For creating an exploit, it is important to determine a way of getting a large buffer into the overflowable buffer. A simple process of filling a buffer over the network



Injection vector: It refers to the customized operational code needed to monitor and control an instruction pointer on the remote system. It depends on host and targeted machine and is used to execute the payload.



Payload: Something like a virus that can run at anytime, anywhere irrespective of its injection into a remote machine.

Determining the location of payload

Both injection vector and payload are commonly located in the stack, but the problem with this approach is that one has to keep track of the payload size and how the payload interacts with injection vector. For example, collision occurs when payload starts before injection vector and a jump instruction is included to overcome this which makes the payload jump over injection code. But, if these problems become too complex, the payload has to be placed somewhere else.

Any location in the program, where you can store a buffer becomes a candidate for storing a payload. The main step is to get the processor to start executing that buffer. Some common places to store payloads include

- Files on disk, which are then loaded into memory
- Environment variables controlled by a local user
- Environment variables passed within a web request
- User-controlled fields within a network protocol

Once the payload is injected, the task is simply to get the instruction pointer to load the address of payload. This technique of storing the payload somewhere other than stack has made tight and difficult to exploit buffer overflows very much possible. A single off-by-one error can still be used to take control of a computer.

Methods to execute payload

There are several techniques that are used to execute payload. These are the ways to decide what to put into the saved EIP on the stack to make it finally point to our code.



Direct Jump (Guessing offsets)

Here, an overflow code is instructed to jump directly to a specific location in memory. No effort to determine the true location of the stack in memory is made. Though it is simple to use, it has two major drawbacks.

- If the address of stack contains a null character, the entire payload has to be placed before the injection i.e. reducing the available space for payload.
- As the address of a payload is not always constant, it requires initial guessing of the address to be jumped.



Blind Return

The ESP register points to the current stack location. Any 'ret' instruction will cause the EIP register to be loaded with whatever is pointed to by ESP. this is called 'popping'. Any ret instruction leads to popping of the EIP with top most value on a stack allowing the EIP to point for a new address. If the attacker is able to inject an initial EIP value that points to a ret instruction, the value stored at ESP will be loaded into the ESI.

Nothing can be injected into the instruction pointer that will cause a register to be used for execution. The instruction pointer is made point to a real instruction.



Pop Return

If the value on the top of the stack does not point to an address within the attacker's buffer, the injected EIP can be set to point to a series of pop instructions followed by a 'ret'. This causes the stack to be popped a number of times, before a value is used for EIP register.

This technique is useful when there is an address near the top of stack that points to within the attacker's buffer and the attacker just pops down the stack until the useful address is reached.



Call Register

If a register is already loaded with an address that points to the payload, the attacker simply needs to load the EIP to an instruction that performs a "call EDX" or "call EDI" or equivalent.

Many useful pairs are found by a search of process memory, and can be used from almost any normal process. As, these are part of kernel interface DLL, they will normally be at fixed address which can be hand coded. These vary for different versions of windows depending on the type of service pack applied.



Push Return

It slightly varies from call register method and it also makes use of the value stored in a register. If the register is loaded, but the attacker cannot find a call instruction, another option is to find a "push" followed by a "return".

Stack Frame:

The term 'stack frame' refers to the collection of the entire information related to a stack of any function. The information includes the arguments that are passed to any function, the stored EIP along with any other stored registers and local variables. It can be effectively explained by the 'call' and 'ret' instructions.

**Call Instruction**

This instruction is used to change the processor control in such a way that the control now points to a different piece of code somewhere inside a program, there by notifying the point where to return after executing the function call. The operations are

- The immediate next instruction after a call is pushed onto the stack to be executed after returning from function.
- Jump to the address available at the top of a stack.



Ret Instruction: The return instruction takes the control back to the location immediately after a call function in the caller. The operations are

- The return address at the top of the stack is popped off
- The address popped off the stack is then jumped

Hence, a combination of 'push' and 'return' statements allow jumping to specific portion of code and returning from it after executing it. As the location of the stored EIP is available onto a stack, writing a popped value at that location is possible.

Computer programs are organized into sub-routines. The program's main-routine calls each subroutine which performs its particular function and then returns control to the main routine. Each subroutine in turn has to save various pieces of information in order to perform its work. Subroutines use an area of memory called the stack for storing this information. One of these pieces of information is the memory address to which the subroutine should return control, when it is finished with its work.

Subroutines also store temporary data on stack. Each time a subroutine is run, the required memory is allocated on the stack in unit called stack frame. The stack frame includes space for any buffers the subroutine requires, as well as the calling routines return address. When the subroutine completes its work, it returns control to the calling routine by jumping the address stored in stack frame, and the stack frame is deleted.

When a user sends 1000 characters to a 100 characters stack buffer, the extra 900 characters overwrite adjacent memory in the stack frame, overwriting other buffers and the stack frame's return address. Now, when the subroutine attempts to return control to the main program, it jumps to the address that is stored in the return address portion of the stack frame. Unfortunately, this address has been overwritten by the overflowed buffer and the address is corrupted. When the program tries to jump to a non-existing address, the program crashes

If the attacker sends 1000 characters that are carefully chosen, he or she can control the return address. Rather than jumping to a non-existing address, the attacker can instruct the program to jump to the address of malicious exploit code (payload).

Format String Vulnerability

In the second half of the year 2000, a whole new class of vulnerabilities has been disclosed and caused a wave of exploitable bugs being discovered in all kinds of programs, ranging from small utilities to big server applications. These are known as '*format string vulnerabilities*'. A format string vulnerability occurs when programmers pass externally supplied data to a *printf* function as or as part of the format string argument.

Format string attacks can be used to crash a program or to execute harmful code. The problem stems from the use of unfiltered user input as the format string parameter in certain C functions that perform formatting, such as `printf()`. These are some of the most commonly seen programming mistakes resulting in exploitable format string vulnerabilities. The first is where a *printf* function is called with no separate format string argument, simply a single string argument. A malicious user may use the `%s` and `%x` format tokens, among others, to print data from the stack or possibly other locations in memory. One may also write arbitrary data to arbitrary locations using the `%n` format token, which commands `printf()` and similar functions to write the number of bytes formatted to an address stored on the stack. A typical exploit uses a combination of these techniques to force a program to overwrite the address of a library function or the return address on the stack with a pointer to some malicious shell code.

Format string bugs most commonly appear when a programmer wishes to print a string containing user supplied data. The programmer may mistakenly write `printf(buffer)` instead of `printf("%s", buffer)`. The first version interprets `buffer` as a format string, and parses any formatting instructions it may contain. The second version simply prints a string to the screen, as the programmer intended.

Format string vulnerability attacks fall into three categories: denial of service, reading and writing.

- Format string vulnerability denial of service attacks are characterized by utilizing multiple instances of the `%s` format specifier to read data off of the stack until the program attempts to read data from an illegal address, which will cause the program to crash.
- Format string vulnerability reading attacks typically utilize the `%x` format specifier to print sections of memory that we do not normally have access to. This is a serious problem and can lead to disclosure of sensitive information. For example, if a program accepts authentication information from clients and does not clear it immediately after use, these vulnerabilities can be used to read it.

- Format string vulnerability writing attacks utilize the %d, %u or %x format specifiers to overwrite the Instruction Pointer and force execution of user-supplied shell code. This is exploited using single write method or multiple writes method.

Session Hijacking:

Session Hijacking is a common-cum valiant security threat to which most systems are prone to. It refers to the exploitation of a valid computer session to gain unauthorized access to information or services in a computer system. Sensitive user information is constantly transported between sessions after authentication and hackers put their best efforts to steal them. Session hijack is a process whereby the attacker inserts themselves into an existing communication session between two computers. The three main protocols that manage the data flow on which session hijacking occurs are TCP, UDP, and HTTP.

Session hijacking can be done at two levels: Network Level and Application Level. Network level hijacking involves TCP and UDP sessions, whereas Application level session hijack occurs with HTTP sessions. The network level refers to the interception and tampering of packets transmitted between client and server during a TCP or UDP session. The application level refers to obtaining session IDs to gain control of the HTTP user session as defined by the web application. In the application level, the session hijacker not only tries to hijack existing sessions, but also tries to create new sessions using stolen data.

TCP Session Hijacking

TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. In order to guarantee that packets are delivered in the right order, TCP uses acknowledgement (ACK) packets and sequence numbers to create a “full duplex reliable stream connection between two end points,” with the end points referring to the communicating hosts. The connection between the client and the server begins with a three-way handshake.



Fig: The three way handshake method for session establishment and sending Data over TCP

- Client sends a synchronization (SYN) packet to the server with initial sequence number X.
- Server responds by sending a SYN/ACK packet that contains the server's own sequence number p and an ACK number for the client's original SYN packet. This ACK number indicates the next sequence number the server expects from the client
- Client acknowledges receipt of the SYN/ACK packet by sending back to the server an ACK packet with the next sequence number it expects from the server, which in this case is P+1.

After the handshake, it's just a matter of sending packets and incrementing the sequence number to verify that the packets are getting sent and received.

The goal of the TCP session hijacker is to create a state where the client and server are unable to exchange data, so that he can forge acceptable packets for both ends, which mimic the real packets. Thus, attacker is able to gain control of the session. At this point, the reason why the client and server will drop packets sent between them is because the server's sequence number no longer matches the client's ACK number and likewise, the client's sequence number no longer matches the server's ACK number. To hijack the session in the TCP network the hijacker should employ following techniques:

- ➔ **IP Spoofing:** IP spoofing is "a technique used to gain unauthorized access to computers, whereby the intruder sends messages to a computer with an IP address indicating that the message is coming from a trusted host." Once the hijacker has successfully spoofed an IP address, he determines the next sequence number that the server expects and uses it to inject the forged packet into the TCP session before the client can respond. By doing so, he creates the "desynchronized state."
- ➔ **Blind Hijacking:** If source routing is disabled, the session hijacker can also employ blind hijacking where he injects his malicious data into intercepted communications in the TCP session. It is called "blind" because the hijacker can send the data or commands, but cannot see the response. The hijacker is basically guessing the responses of the client and server.

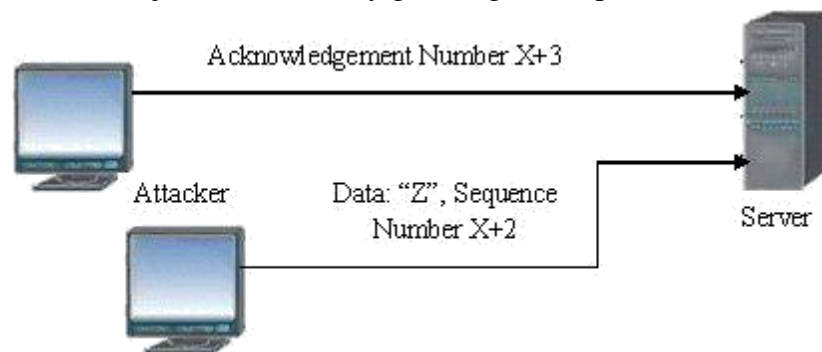


Fig: Blind Injection technique

- ➔ **Man in the Middle attack** (packet sniffing): This technique involves using a packet sniffer that intercepts the communication between the client and server. With all the data

between the hosts flowing through the hijacker's sniffer, he is free to modify the content of the packets. The trick to this technique is to get the packets to be routed through the hijacker's host.

UDP Session Hijacking

UDP which stands for User Datagram Protocol is defined as "a connectionless protocol that, like TCP, runs on top of IP networks. Unlike TCP/IP, UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagram's over an IP network." Therefore, the delivery, integrity, non-duplication and ordering are not guaranteed i.e. it does not use packet sequencing and synchronizing. UDP doesn't use sequence numbers like TCP. It is mainly used for broadcasting messages across the network or for doing DNS queries.



Fig: Session Hijacking over UDP

Hijacking a session over User Datagram Protocol (UDP) is exactly the same as over TCP, except that UDP attackers do not have to worry about the overhead of managing sequence number and other TCP mechanisms. Since UDP is connectionless, injecting data into session without being detected is extremely easy. If the "man in the middle" situation exists, this can be very easy for the attacker, since he can also stop the server's reply from getting to the client in the first place

To defend a network against these attacks, a defender has to implement both security measures at Application level and Network level. Network level hijacks can be prevented by **ciphering the packets** so that the hijacker cannot decipher the packet headers, to obtain any information which will aid in spoofing. This encryption can be provided by using protocols such as **IPSEC, SSL, SSH** etc. To prevent your Application session to be hijacked it is recommended to use **Strong Session ID's** so that they cannot be hijacked or deciphered at any cost.

Route Table Modification:

An attacker would be able to put himself in such a position to block packets by modifying routing tables, so that packets flow through a system he has control of (Layer 3 redirection), by changing bridge tables by playing games with spanning-tree frames (Layer 2 redirection), or by rerouting physical cables so that the frames must flow through the attacker's system (Layer 1 redirection). Most of the time, an attacker will try to change route tables remotely. There has been some research in the area of changing route tables on a mass scale by playing games with the Border Gateway Protocol (BGP) that most Internet service providers (ISPs) use to exchange routes with each other.

A more locally workable attack might be to spoof Internet Control Message Protocol (ICMP) and redirect packets to fool some hosts into thinking that there is a better route via the attacker's IP address. Many OS's accept ICMP redirects in their default configuration. Unless, the connection is to be broken entirely (or proxy it in some way), the packets have to be forwarded back to the real router, so they can reach their ultimate destination. When that happens, the real router is likely to send ICMP redirect packets to the original host, too, informing it that there is a better route. To attempt that sort of attack, it is necessary to keep up the flow of ICMP redirect messages.

If the attacker has managed to change route tables to get packets to flow through his system, some of the intermediate routers will be aware of the route change, either because of route tables changing or possibly because of an Address Resolution Protocol (ARP) table change. The end nodes would not normally be knowledgeable to this information, if there are at least a few routers between the two nodes. Possibly the nodes could discover the change via a traceroute-style utility, unless the attacker has planned for that and programmed his "router" to account for it (by not sending the ICMP unreachable and not decrementing the Time-to-Live [TTL] counter on the IP packets).

ARP Attacks

Another way to make sure that your attacking machine gets all the packets going through it is to modify the ARP tables on the victim machine(s). An ARP table controls the Media Access Control (MAC)-address-to-IP-address mapping on each machine. ARP is designed to be a dynamic protocol, so as new machines are added to a network or existing machines get new MAC addresses for whatever reason, the rest update automatically in a relatively short period of time. There is absolutely no authentication in this protocol.

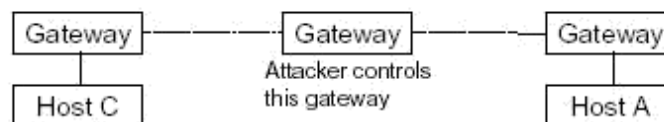
Address Resolution Protocol (ARP) spoofing, also known as **ARP poisoning** or **ARP Poison Routing (APR)**, is a technique used to attack an Ethernet wired or wireless network. ARP

Spoofing allows an attacker to sniff data frames on a local area network (LAN), modify the traffic, or stop the traffic altogether. The attack can only be used on networks that actually make use of ARP and not another method of address resolution.

The principle of ARP spoofing is to send fake, or "spoofed", ARP messages to an Ethernet LAN. Generally, the aim is to associate the attacker's MAC address with the IP address of another node (such as the default gateway). Any traffic meant for that IP address would be mistakenly sent to the attacker instead. The attacker could then choose to forward the traffic to the actual default gateway (passive sniffing) or modify the data before forwarding it (man-in-the-middle attack). The attacker could also launch a denial-of-service attack against a victim by associating a nonexistent MAC address to the IP address of the victim's default gateway. ARP spoofing attacks can be run from a compromised host or from an attacker's machine that is connected directly to the target Ethernet segment. Also spoofed ARP replies are sent at an extremely rapid rate to the switch making its MAC table to overflow and sometimes resulting in switches being reverted to broadcast mode, allowing the sniffing to be done. The best defense against ARP attacks are having a static ARP, DHCP Snooping (access control based on IP, MAC, and port) and detection. Some detection techniques are ARPWatch (Free UNIX Program), Reverse ARP (RARP- used to detect MAC cloning) and Promiscuous Mode Sniffing.

Man in the Middle Attacks

In cryptography, the **man-in-the-middle attack** (often abbreviated **MITM**), is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all messages going between the two victims and inject new ones, which is straightforward in many circumstances (ex: unencrypted Wi-Fi access point).

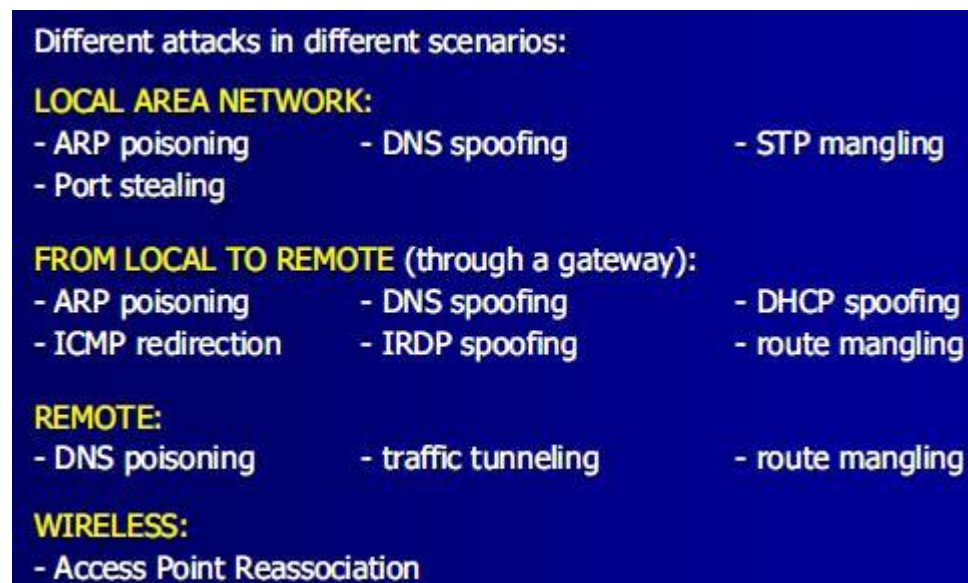


This is not easy in the Internet because of hop-by-hop routing, unless you control one of the backbone hosts or source routing is used. This can also be done combined with IP source routing option. IP source routing is used to specify the route in the delivery of a packet, which is independent of the normal delivery mechanisms. If the traffic can be forced through specific routes (=specific hosts), and if the reverse route is used to reply traffic, a host on the route can

easily impersonate another host. Once in the middle, the attacker can perform injection, key manipulation, downgrade attack and filtering.

Injection implies possibility of adding packets to an already established connection or modifying sequence numbers, maintaining connection synchronization while injecting packets. Key manipulation is possible in protocols like SSHv1(modification of public key exchanged by client and server), IPSEC, HTTPS (issuing fake certificates to clients relying on browser misconfiguration). Downgrade attacks involve forcing a client to initialize a SSH1 connection rather than SSH2 or sometimes blocking the key material exchanged in IPSEC. In filtering attacks, the attacker can modify the payload of packets by recalculating the checksum or can create filters in the path and in some cases like full-duplex can change the length of payload.

Various kinds of MITM attacks in different scenarios are given below:



Different attacks in different scenarios:

LOCAL AREA NETWORK:

- ARP poisoning
- DNS spoofing
- STP mangling
- Port stealing

FROM LOCAL TO REMOTE (through a gateway):

- ARP poisoning
- DNS spoofing
- DHCP spoofing
- ICMP redirection
- IRDP spoofing
- route mangling

REMOTE:

- DNS poisoning
- traffic tunneling
- route mangling

WIRELESS:

- Access Point Reassociation

Assignment Questions

- 1) a) Define a Security attack. Explain in detail about the various types of attacks an Internetwork is vulnerable to.
b) Write about Man-in-the-middle attacks.
- 2) a) "Gaining control over the Routing tables at layer 3 is one of the attacks" – explain how Route tables modification is crucial.
b) Explain how Buffer overflow is created for any known platforms (eg., WIN- DOWS NT / LINUX).
- 3) a) "Internetwork security is both fascinating and complex" - Justify the statement with valid reasoning.
b) Explain the terms related to Buffer overflow:
 - i. Stack dumping
 - ii. Execute Payload.
- 4) a) Explain about how the Internet standards and RFCs.
b) Explain how Address Resolution Protocol table becomes a victim for attacks.
- 5) a) Describe the various Security Services.
b) Compare TCP session hijacking and UDP hijacking.