CHAPTER 7

MALICIOUS LOGIC

CHAPTER OUTLINE

After studying this chapter, the students will be able to

- Malicious Logic, Types of Malicious Logic: Virus, Worm, Trojan Horse, Zombies, Denial of Service Attacks,
- Intrusion, Intruders and their types, Intrusion Detection System

MALICIOUS LOGIC

The most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems. Such threats are referred to as **malicious software,** or **malware. Malicious logic** is a set of instructions that cause a site's security policy to be violated. For example, following UNIX script is named *ls* and is placed in a directory.

```
cp /bin/sh/temp/.xxsh
chmod u+s,o+x /temp/.xxsh
rm ./ls
ls $*
```

It creates a copy of the UNIX shell that is setuid to the user executing this program. This program is deleted, and then the correct *ls* command is executed. On most systems, it is against policy to trick someone into creating a shell that is setuid to them. If someone is tricked into executing this script, a violation of the (implicit) security policy occurs. This script is an example of malicious logic.

TYPES OF MALICIOUS LOGIC

The terminology in this area presents problems because of a lack of universal agreement on all of the terms and because some of the categories overlap. Table 7.1 is a useful guide.

Table 7.1: Terminologies of Malicious Logic

Name	Description
Virus	Attaches itself to a program
Worm	Propagates copies of itself to other computers
Logic bomb	"explodes" when a condition occurs
Trojan Horse	Fakes/contains additional functionality
Backdoor (Trapdoor)	Allows unauthorized access to functionality
Mobile Code	Moves unchanged to heterogeneous platforms
Auto-rooter Kit (Virus generator)	Malicious code (virus) generators
Spammer and flooder Programs	Large volume of unwanted "pkts"
Keyloggers	Capture keystrokes
Rootkit	Sophisticated hacker tools to gain root-level access
Zombie	Software on infected computers that launch attack on other (aka bot)

Malicious Logics can be divided into two categories: those that need a host program, and those that are independent. The former, referred to as parasitic, are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. Independent malware is a self-contained program that can be scheduled and run by the operating system. Worms and Zombie programs are examples.

j ko barema sodeni diagram ghusaune

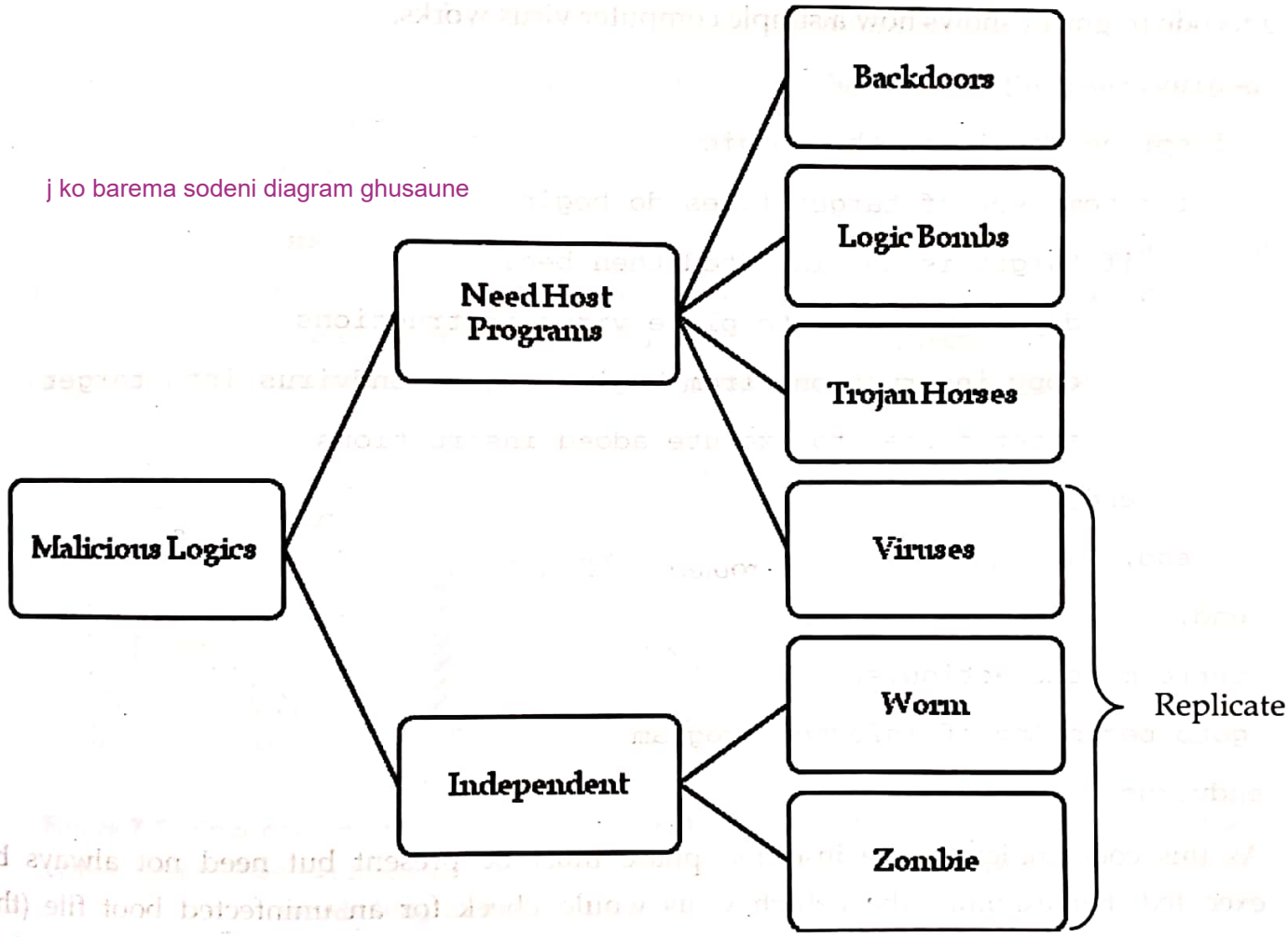

Figure 7.1: Malicious software categories according to need of host machine

We can also differentiate between those software threats that do not replicate and those that do. The former are programs or fragments of programs that are activated by a trigger. Examples are logic bombs, backdoors programs. The latter consist of either a program fragment or an independent program that, when executed, may produce one or more copies of itself to be activated later on the same system or some other system. Viruses and worms are examples.

COMPUTER VIRUS

A computer virus is a hidden, self-replicating section of computer software, usually malicious logic, that propagates by infecting - i.e., inserting a copy of itself into and becoming part of - another program. A virus cannot run by itself; it requires that its host program be run to make the virus active. In brief, a **computer virus** is a program that inserts itself into one or more files and then performs some (possibly null) action.

A virus can do anything that other programs do. The difference is that a virus attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs that is allowed by the privileges of the current user.

The first phase, in which the virus inserts itself into a file, is called the **insertion phase.** The second phase, in which it performs some action, is called the **execution phase.** The following pseudocode fragment shows how a simple computer virus works.

```
beginvirus:    dormant                    begin vnerw tala ko kaam talako kaam gar vneko raexa
 if spread-condition then begin    propagation
    for some set of target files do begin
       if target is not infected then begin
          determine where to place virus instructions
          copy instructions from beginvirus to endvirus into target
          alter target to execute added instructions    trigger
       end;
    end;
 end;
 perform some action(s)    execution
 goto beginning of infected program
endvirus:
```

suruma if spread vnerw condition xa vne balla gara vaniraxa ani target files haru hunxani tw ani teha ko file ma rkahne if vyenw vne duiwota steps grne thyakkae place determine grne ani copy grdihlne ani teha euta alert instruction dekhaune to execute

As this code indicates, the insertion phase must be present but need not always be executed. For example, the Lehigh virus would check for an uninfected boot file (the spread-condition mentioned in the pseudocode) and, if one was found, would infect that file (the set of target files). Then it would increment a counter and test to see if the counter was at 4. If so, it would erase the disk. These operations were the action(s).

Several types of computer virus have been identified.

- **Boot Sector Infectors:** The boot sector is the part of a disk used to bootstrap the system or mount a disk. Code in that sector is executed when the system "sees" the disk for the first time. When the system boots or the disk is mounted, any virus in that sector is executed. A **boot sector infector** is a virus that inserts itself into the boot sector of a disk.

  Example: The Brain virus for the IBM PC is a boot sector infector. When the system boots from an infected disk, the virus is in the boot sector and is loaded. It moves the disk interrupt vector (location 13H or 19) to an alternative interrupt vector (location 6DH or 109) and sets the disk interrupt vector location to invoke the Brain virus now in memory. It then loads the original boot sector and continues the boot.

Whenever the user reads a floppy, the interrupt at location 13H is invoked. The Brain virus checks for the signature 1234H in the word at location 4. If the signature is present, control is transferred to the interrupt vector at location 6DH so that a normal read can proceed. Otherwise, the virus infects the disk.

To do this, it first allocates to itself three contiguous clusters (of two contiguous sectors each). The virus then copies the original boot sector to the first of the six contiguous sectors and puts copies of itself into the boot sector and the remaining five sectors.

If there are no unused clusters, the virus will not infect the disk. If it finds only one unused cluster, it will simply overwrite the next two. This accounts for the sometimes destructive nature of the Brain virus.

- **Executable infectors:** An **executable infector** is a virus that infects executable programs. The PC variety of executable infectors are called COM or EXE virus because they infect programs with those extensions. Figure 7.2 illustrates how infection can occur. The virus can pretend itself to the executable (as shown in the figure) or append itself.

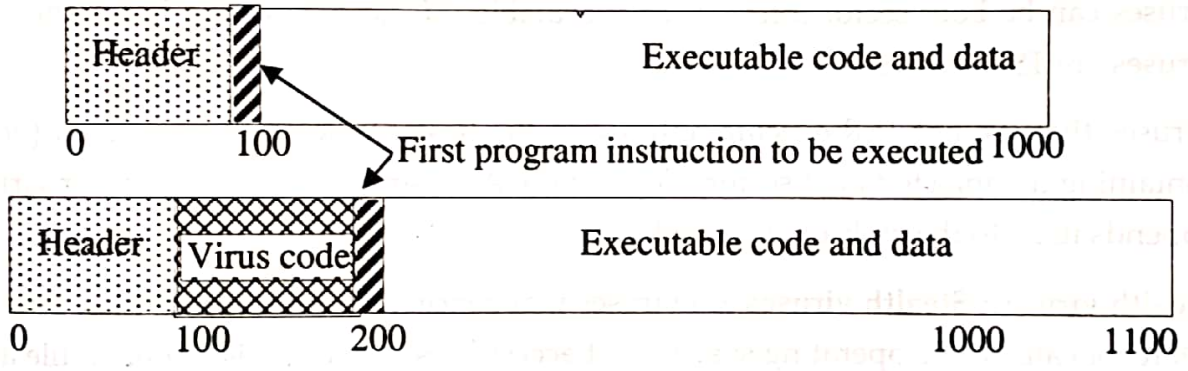

Figure 7.2: How and executable infector works. It inserts itself into the program so that the virus code will be executed before the application code. In this example, the virus is 100 words long and pretends itself to the executable code.

Example: The Jerusalem virus (also called the Israeli virus) is triggered when an infected program is executed. The virus first puts the value 0E0H into register ax and invokes the DOS service interrupt (21H). If on return the high eight bits of register *ax* contain 03H, the virus is already resident on the system and the executing version quits, invoking the original program. Otherwise, the virus sets itself up to respond to traps to the DOS service interrupt vector.

The Jerusalem virus then checks the date. If the year is 1987, it does nothing. Otherwise, if it is not a Friday and not the 13th (of any month), it sets itself up to respond to clock interrupts (but it will not infect on clock calls). It then loads and executes the file originally executed. When that files finishes, the virus puts itself in memory. It then responds to calls to the DOS service interrupt. If it is a Friday and the 13th (of any month), and the year is not 1987, the virus sets a flag in memory to be destructive. This flag means that the virus will delete files instead of infecting them.

Once in memory, the virus checks all calls to the DOS service interrupt, looking for those asking that files be executed (function 4B00H). When this happens, the virus checks the name of the file. If it is COMND.COM, the virus does nothing. If the memory flag is set to be destructive, the file is deleted. Otherwise, the virus checks the last five bytes of the file. If they are the string "MsDos," the file is infected. If they are not, the virus checks the last character of the file name. If it is "M," the virus assumes that a .COM file is being executed and infects it; if it is "E," the virus assumes that a .EXE file is being executed and infects it. The file's attributes, especially the date and time of modification, are left unchanged.

- **Multipartite viruses:** A **multipartite virus** is one that can infect either boot sectors or applications. Such a virus typically has two parts, one for each type. When it infects an executable, it acts as an executable infector; when it infects a boot sector, it works as a boot sector infector.

- **TSR viruses:** A **terminate and stay resident** (TSR) virus is one that stays active (resident) in memory after the application (or bootstrapping, or disk mounting) has terminated. TSR viruses can be boot sector infectors or executable infectors. Both the Brain and Jerusalem viruses are TSR viruses.

  Viruses that are not TSR execute only when the host application is executed (or the disk containing the infected boot sector is mounted). An example is the Encroacher virus, which appends itself to the ends of executables.

- **Stealth viruses: Stealth viruses** are viruses that conceal the infection of files. These viruses intercept calls to the operating system that access files. If the call is to obtain file attributes, the original attributes of the file are returned. If the call is to read the file, the file is disinfected as its data is returned. But if the call is to execute the file, the infected file is executed.

  Example: The Stealth virus (also called the IDF virus or the 4096 virus) is an executable infector. It modifies the DOS service interrupt handler (rather than the interrupt vector; this way, checking the values in the interrupt vector will not reveal the presence of the virus). If the request is for the length of the file, the length of the uninfected file is returned. If the request is to open the file, the file is temporarily disinfected; it is reinfected on closing. The Stealth virus also changes the time of last modification of the file in the file allocation table to indicate that the file is infected.

- **Encrypted viruses:** An **encrypted virus** is one that enciphers all of the virus code except for a small decryption routing. Computer virus detectors often look for known sequences of code to identify computer viruses. To conceal these sequences, some viruses encipher most of the virus code, leaving only a small decryption routine and a random cryptographic key in the clear. Figure 7.3 summarizes this technique.

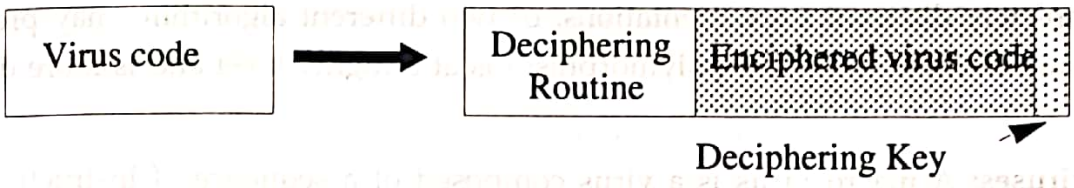

Figure 7.3: An encrypted virus. The ordinary virus code is at the left. The encrypted virus, plus encapsulating decryption information, is at the right.

Example: Ferbrache4 cites the following as the decryption code in the 1260 virus. It uses two keys, stored in *k1* and *k2*. The virus code itself begins at the location *sov* and ends at the location *eov*. The pseudocode is as follows.

```
(* initialize the registers with the keys *)
rA ← k1;
rB ← k2;
(* initialize rC with the message *)
rC ← sov;
(* the encipherment loop *)
while (rC != eov) do begin
        (* encipher the byte of the message *)
        (*rC) ← (*rC) xor rA xor rB;
        (* advance all the counters *)
        rC ← rC + 1;
        rA ← rA + 1;
end
```

The dual keys and the shifting of the first key prevent a simple *xor*'ing from uncovering the deciphered virus.

- **Polymorphic viruses:** A **polymorphic virus** is a virus that changes its form each time it inserts itself into another program. Consider an encrypted virus. The body of the virus varies depending on the key chosen, so detecting known sequences of instructions will not detect the virus. However, the decryption algorithm can be detected. Polymorphic viruses were designed to prevent this. They change the instructions in the virus to something equivalent but different. In particular, the deciphering code is the segment of the virus that is changed. In some sense, they are successors to the encrypting viruses and are often used in conjunction with them.

  Consider polymorphism at the instruction level. All of the instructions have exactly the same effect, but they are represented as different bit patterns on most architecture. A polymorphic virus would insert these instructions into the deciphering segment of code.

```
add 0 to operand
or 1 with operand
no operation
subtract 0 from operand
```

Polymorphism can exist at many levels. For example, a deciphering algorithm may have two completely different implementations, or two different algorithms may produce the same result. In these cases, the polymorphism is at a higher level and is more difficult to detect.

- **Macro viruses:** A **macro virus** is a virus composed of a sequence of instructions that is interpreted, rather than executed directly. Conceptually, macro viruses are no different from ordinary computer viruses. Like Duff's *sh* computer virus, they can execute on any system that can interpret the instructions. For example, a spreadsheet virus executes when the spreadsheet interprets these instructions. If the macro language allows the macro to access files or other systems, the virus can access them, too.

  A macro virus can infect either executables or data files (the latter leads to the name data virus). If it infects executable files, it must arrange to be interpreted at some point. Duff's experiments did this by wrapping the executables with shell scripts. The resulting executables invoked the Bourne shell, which interpreted the virus code before invoking the usual executable.

  Example: The Melissa virus infected Word 97 and 98 documents on Windows and Macintosh systems. It is invoked when the program opens an infected file. It installs itself as the "open" macro and copies itself into the Normal template (so any files that are opened are infected). It then invokes a mail program and sends copies of itself to people in the user's address book associated with the program.

COMPUTER WORM

A computer virus infects other programs. A variant of the virus is a program that spreads from computer to computer, spawning copies of itself on each one. A **computer worm** is a program that copies itself from one computer to another with the goal of overtaking the entire network of computers. Most worms are designed to infiltrate systems by exploiting their security failures, while very few also try to change the system settings. Even if they don't, they are still very dangerous as they take up a lot of bandwidth and other valuable resources. If a worm is indeed malicious and not just used to breach the system security, the code designed to carry out the attack is referred to as the payload. Payloads are usually created to change or delete files on a target network, extract personal data from them, or encrypt them and seek a ransom from the victim.

Despite the fact that many people use the two terms interchangeably, computer worms are not the same as computer viruses. For one, computer viruses by definition target individual computers, whereas worms target networks of computers to create botnets. Furthermore, while viruses are usually bundled with legitimate files or programs, computer worms are standalone and don't require a host file.

Research into computer worms began in the mid-1970s. Schoch and Hupp developed distributed programs to do computer animations, broadcast messages, and perform other computations. These programs probed workstations. If the workstation was idle, the worm

copied a segment onto the system. The segment was given data to process and communicated with the worm's controller. When any activity other than the segments began on the workstation, the segment shut down.

Example: On November 2, 1988, a program targeting Berkeley and Sun UNIX based computers entered the Internet; within hours, it had rendered several thousand computers. Among other techniques, this program used a virus-like attack to spread: it inserted some instructions into a running process on the target machine and arranged for those instructions to be executed. To recover, these machines had to be disconnected from the network and rebooted, and several critical programs had to be changed and recompiled to prevent reinfection. Worse, the only way to determine if the program had suffered other malicious side effects (such as deletion of files) was to disassemble it. Fortunately, the only purpose of this virus turned out to be self-propagation. Infected sites were extremely lucky that the worm5 did not infect a system program with a virus designed to delete files and did not attempt to damage attacked systems.

Since then, there have been several incidents involving worms. The **Father Christmas** worm was interesting because it was a form of macro worm.

Example: Slightly before the Internet worm, an electronic "Christmas card" passed around several IBM-based networks. This card was an electronic letter instructing the recipient to save the message and run it as a program. The program drew a Christmas tree (complete with blinking lights) and printed "Merry Christmas!" It then checked the recipient's list of previously received mail and the recipient's address book to create a new list of e-mail addresses. It then sent copies of itself to all these addresses. The worm quickly overwhelmed the IBM networks and forced the networks and systems to be shut down.

This worm had the characteristics of a macro worm. It was written in a high-level job control language, which the IBM systems interpreted. Like the Melissa virus, which was written in the Visual Basic programming language, the Father Christmas worm was never directly executed—but its effects (spreading from system to system) were just as serious.

Some Examples of a Computer Worm

Jerusalem, the first known computer worm, was discovered in 1987. Since then, other computer worms have made the news, either because of their devastating effects or due to the sheer scale of the attack. Some of the most notorious examples of computer worms include the following:

- **The Morris Worm** was launched in 1988 by Robert Morris, an American student who wanted to discover how big the internet really was. To do this, he launched a few dozen lines of code, but he didn't know that the code was riddled with bugs that would cause a variety of problems on affected hosts. The result was thousands of overloaded computers running on UNIX and a financial damage ranging between $10 million and $100 million.

- **The Storm Worm** is an email worm launched in 2007. Victims would receive emails with a fake news report about an unprecedented storm wave that had already killed hundreds of people across Europe. More than 1.2 billion of these emails were sent over the course of ten years in order to create a botnet that would target popular websites. Experts believe that there are still at least a million infected computers whose owners don't know that they are part of a botnet.
- **SQL Slammer** was unique in that it didn't utilize any of the traditional distribution methods. Instead, it generated a number of random IP addresses and sent itself out to them in hopes that they weren't protected by antivirus software. Soon after it hit in 2003, the result was more than 75,000 infected computers unknowingly involved in DDoS attacks on several major websites.

Types of Computer Worms

There is no universal classification of computer worms, but they can be organized into types based on how they are distributed between computers. The five common types are as follows:

- **Internet Worms:** Like they do with computer networks, computer worms also target popular websites with insufficient security. When they manage to infect the site, internet worms can replicate themselves onto any computer being used to access the website in question. From there, internet worms are distributed to other connected computers through the internet and local area network connections.
- **Email Worms:** Email worms are most often distributed via compromised email attachments. They usually have double extensions (for example, .mp4.exe or .avi.exe) so that the recipient would think that they are media files and not malicious computer programs. When the victims click on the attachment, copies of the same infected file will automatically be sent to addresses from their contacts list.

  An email message doesn't have to contain a downloadable attachment to distribute a computer worm. Instead, the body of the message might contain a link that's shortened so that the recipient can't tell what it's about without clicking on it. When they click on the link, they will be taken to an infected website that will automatically start downloading malicious software to their computer.
- **Instant Messaging Worms:** Instant messaging worms are exactly the same as email worms, the only difference being their method of distribution. Once again, they are masked as attachments or clickable links to websites. They are often accompanied by short messages like "LOL" or "You have to see this!" to trick the victim into thinking that their friend is sending them a funny video to look at.

  When the user clicks on the link or the attachment – be it in Messenger, WhatsApp, Skype, or any other popular messaging app – the exact same message will then be sent to their contacts. Unless the worm has replicated itself onto their computer, users can solve this problem by changing their password.

- **File-Sharing Worms:** Although illegal, file-sharing and peer-to-peer file transfers are still used by millions of people around the world. Doing so, they are unknowingly exposing their computers to the threat of file-sharing worms. Like email and instant messaging worms, these programs are disguised as media files with dual extensions.

  When the victim opens the downloaded file to view it or listen to it, they will download the worm to their computer. Even if it seems that users have downloaded an actual playable media file, an executable malicious file could be hidden in the folder and discreetly installed when the media file is first opened.

- **IRC Worms:** Internet Relay Chat (IRC) is a messaging app that is mostly outdated nowadays but was all the rage at the turn of the century. Same as with today's instant messaging platforms, computer worms were distributed via messages containing links and attachments. The latter was less effective due to an extra layer of protection that prompted users to accept incoming files before any transfer could take place.

TROJAN HORSE

A **Trojan horse** is a program with an overt (documented or known) effect and a covert (undocumented or unexpected) effect.

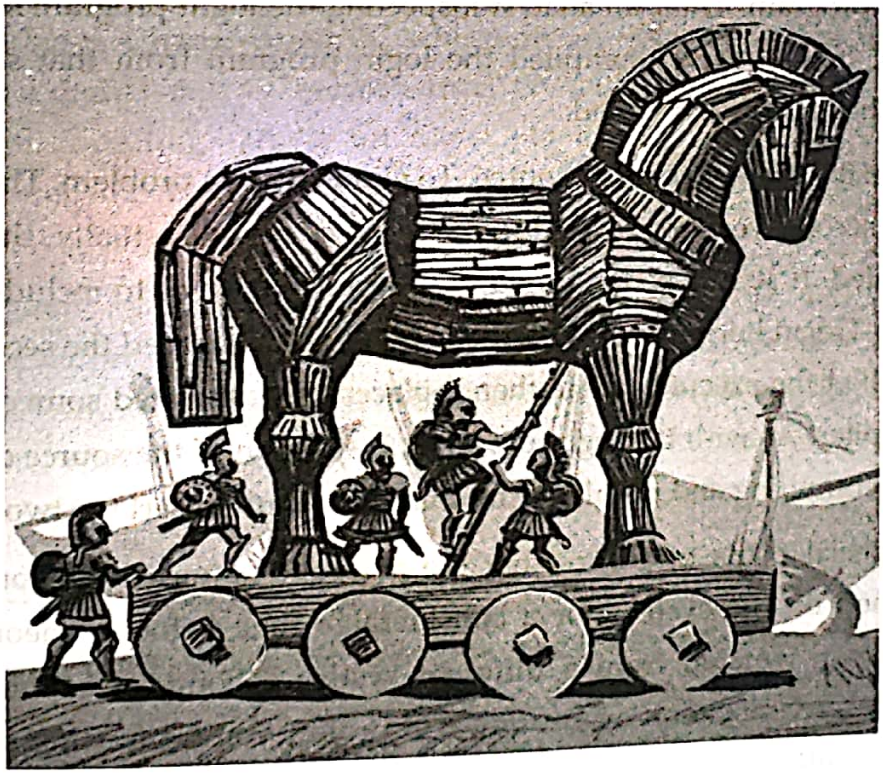

Figure 7.4: Trojan horse (Wooden horse)

A critical observation is the notion of "tricked." Suppose the user root executed this script unintentionally (for example, by typing "ls" in the directory containing this file). This would be a violation of the security policy. However, if root deliberately typed the security policy would not be violated. This illustrates a crucial component of the problems with malicious logic. The system cannot determine whether the instructions being executed by a process are known to the user or are a set of instructions that the user does not intend to execute. The next definition makes this distinction explicit.

```
cp /bin/sh /tmp/.xxsh
chmod o+s,w+x /tmp/.xxsh
```

Example: In the preceding example, the overt purpose is to list the files in a directory. The covert purpose is to create a shell that is setuid to the user executing the script. Hence, this program is a Trojan horse.

A **propagating Trojan horse** (also called a replicating Trojan horse) is a Trojan horse that creates a copy of itself. Karger and Schell, and later Thompson, examined detection of Trojan horses. They constructed a Trojan horse that propagated itself slowly and in a manner that was difficult to detect. The central idea is that the Trojan horse modifies the compiler to insert itself into specific programs, including future versions of the compiler itself.

Example: Thompson added a Trojan horse to the login program. When a user logged in, the Trojan horse would accept a fixed password as well as the user's normal password. However, anyone reading the source code for the login program would instantly detect this Trojan horse. To obscure it, Thompson had the compiler check the program being compiled. If that program was login, the compiler added the code to use the fixed password. Now, no code needed to be added to the login program. Thus, an analyst inspecting the login program source code would see nothing amiss. If the analyst compiled the login program from that source, she would believe the executable to be uncorrupted.

The extra code is visible in the compiler source. To eliminate this problem, Thompson modified the compiler. This second version checked to see if the compiler (actually, the C preprocessor) was being recompiled. If so, the code to modify the compiler so as to include both this Trojan horse and the login Trojan horse code would be inserted. He compiled the second version of the compiler and installed the executable. He then replaced the corrupted source with the original version of the compiler. As with the login program, inspection of the source code would reveal nothing amiss, but compiling and installing the compiler would insert the two Trojan horses.

Thompson took special pains to ensure that the second version of the compiler was never released. It remained on the system for a considerable time before someone overwrote the executable with a new version from a different system. Thompson's point1 was that "no amount of source-level verification or scrutiny will protect you from using untrusted code," a point to be reiterated later.

ZOMBIES

A bot is a type of malware which allows an attacker to gain complete control over the affected computer. Computers that are infected with a bot are generally referred to as 'zombies'. Networks of Zombie Computers are referred as Botnets or zombie army.

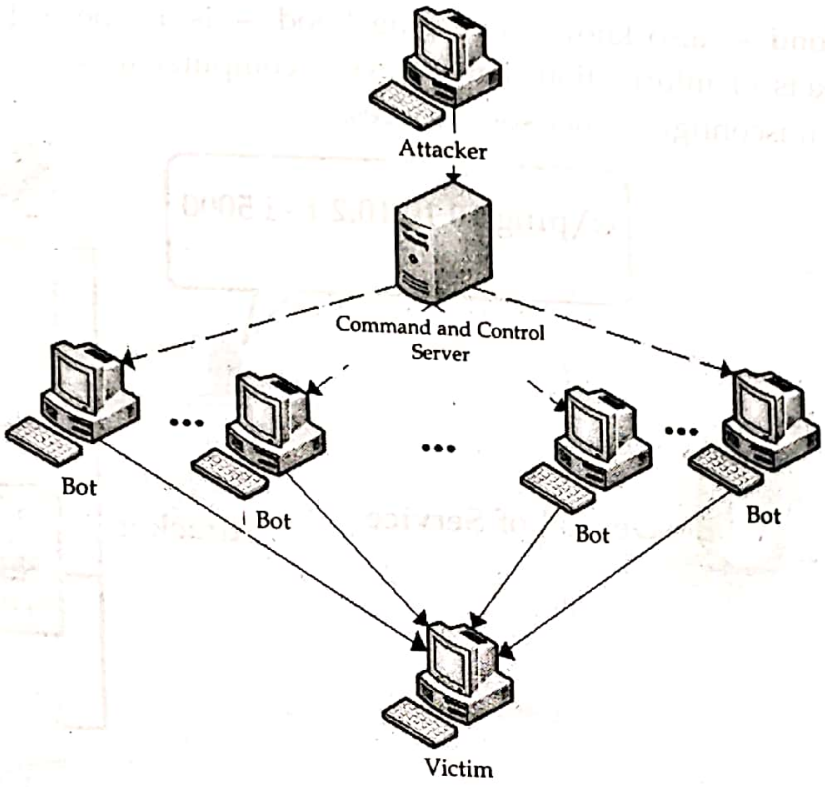

Figure 7.5: Botnets

Typically, a zombie is a home-based PC whose owner is unaware that the computer is being exploited by an external party. The increasing prevalence of high speed connections makes home computers appealing targets for attack. Inadequate security measures make access relatively easy for an attacker. For example, if an Internet port has been left open, a small Trojan horse program can be left there for future activation.

There are a few other kinds of zombies: In one form of **denial of service attack,** a zombie is an insecure Web server on which malicious people have placed code that, when triggered at the same time as other zombie servers, will launch an overwhelming number of requests toward an attacked Web site, which will soon be unable to service legitimate requests from its users. A **pulsing zombie** is one that launches requests intermittently rather than all at once.

DENIAL OF SERVICE ATTACKS

A **denial-of-service (DoS)** attack is a type of cyber attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to addition users. A DoS attack is characterized by using a single computer to launch the attack.

DoS attacks generally take one of two forms. They either flood web services or crash them.

- **Flooding attacks:** Flooding is the more common form DoS attack. It occurs when the attacked system is overwhelmed by large amounts of traffic that the server is unable to handle. The system eventually stops.

An **ICMP flood** – also known as a ping flood – is a type of DoS attack that sends spoofed packets of information that hit every computer in a targeted network, taking advantage of misconfigured network devices.

here attacker typically sends
large number of ping packets to attacked computer
infect single computer at a time
but can also infect mulitple computers by
using large number or multiple ping packets

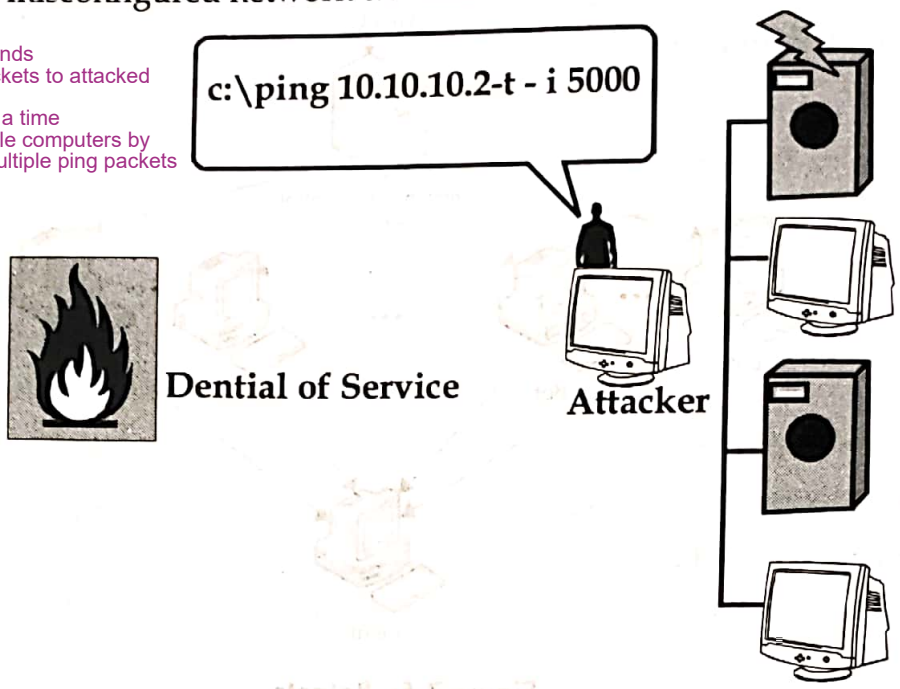

Figure 7.6: Ping of death (Ping flooding)

A **SYN flood** is a variation that exploits vulnerability in the TCP connection sequence. This is often referred to as the three-way handshake connection with the host and the server. Here's how it works:

The targeted server receives a request to begin the handshake. But, in a SYN flood, the handshake is never completed. That leaves the connected port as occupied and unavailable to process further requests. Meanwhile, the cybercriminal continues to send more and more requests, overwhelming all open ports and shutting down the server.

crash attack refers to crash of the system
this may occur due to traffic overload to the server or requests .basically when ping attack and flooding are not resolved then this may lead to crash

- **Crash attacks:** Crash attacks occur less often, when cybercriminals transmit bugs that exploit flaws in the targeted system. The result? The system crashes.

Crash attacks – and flooding attacks – prevent legitimate users from accessing online services such as websites, gaming sites, email, and bank accounts.

The first DoS attack was done by 13-year-old David Dennis in 1974. Dennis wrote a program using the "external" or "ext" command that forced some computers at a nearby university research lab to power off. DoS attacks have evolved into the more complex and sophisticated **"distributed denial of service" (DDoS)** attacks. Distributed Denial of Service (DDoS) attack is an attack where multiple compromised systems simultaneously attack a single system; thereby, causing a DOS attacks for the users of the target. One of the biggest DDoS incidents ever occurred in late October 2016, when a massive attack on Domain Name System (DNS) service provider Dyn temporarily shuttered some of the most popular sites on the Internet. The attack impacted users in much of the East Coast of the United States and data centers in Texas, Washington and California. Tens of millions of IP addresses hit Dyn's infrastructure during the attack.

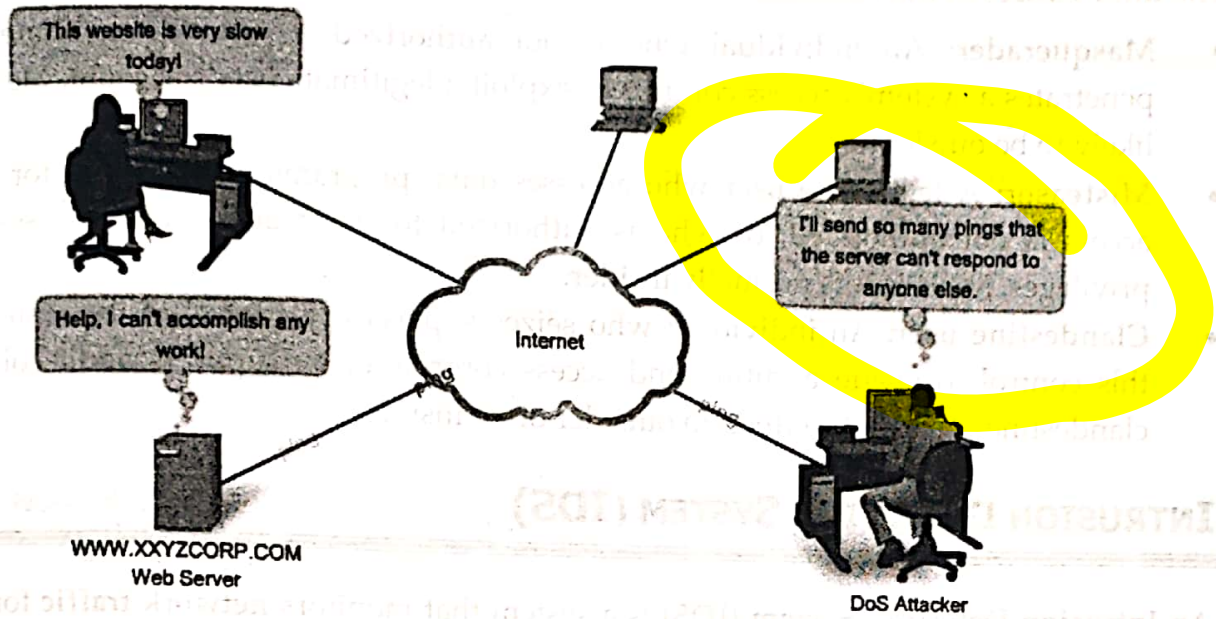

Figure 7.7: DDoS attack

INTRUDERS

An Intruder is a person who attempts to gain unauthorized access to a system, to damage that system, or to disturb data on that system. In summary, this person attempts to violate Security by interfering with system Availability, data Integrity or data Confidentiality.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.

Following are the examples of intrusion:

- Performing a remote root compromise of an e-mail server
- Defacing a Web server
- Guessing and cracking passwords
- Copying a database containing credit card numbers
- Viewing sensitive data, including payroll records and medical information, without authorization
- Running a packet sniffer on a workstation to capture usernames and passwords
- Using a permission error on an anonymous FTP server to distribute pirated software and music files
- Dialing into an unsecured modem and gaining internal network access
- Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password
- Using an unattended, logged-in workstation without permission

The three classes of intruders are;

- **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account. Masquerader is likely to be outsider.
- **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges. Misfeasor is normally insider.
- **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection. The clandestine user can be either an outsider or an insider.

INTRUSION DETECTION SYSTEM (IDS)

An **Intrusion Detection System (IDS)** is a system that monitors **network traffic** for suspicious activity and issues alerts when such activity is discovered. It is a software application that scans a network or a system for harmful activity or policy breaching. Any malicious venture or violation is normally reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system integrates outputs from multiple sources and uses alarm filtering techniques to differentiate malicious activity from false alarms.

Although intrusion detection systems monitor networks for potentially malicious activity, they are also disposed to false alarms. Hence, organizations need to fine-tune their IDS products when they first install them. It means properly setting up the intrusion detection systems to recognize what normal traffic on the network looks like as compared to malicious activity. Intrusion prevention systems also monitor network packets inbound the system to check the malicious activities involved in it and at once sends the warning notifications.

Intrusion detection systems come in different flavors and detect suspicious activities using different methods, including the following:

- **Network Intrusion Detection System (NIDS):** Network intrusion detection systems (NIDS) are set up at a planned point within the network to examine traffic from all devices on the network. It performs an observation of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the collection of known attacks. Once an attack is identified or abnormal behavior is observed, the alert can be sent to the administrator. An example of an NIDS is installing it on the subnet where firewalls are located in order to see if someone is trying to crack the firewall.
- **Host Intrusion Detection System (HIDS):** Host intrusion detection systems (HIDS) run on independent hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the administrator if suspicious or malicious activity is detected. It takes a snapshot of existing system files and compares it with the previous snapshot. If the analytical system files were edited or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission critical machines, which are not expected to change their layout.

euta matra device lai herne
ani tyo device ko paila ko files rw
ahele ko system files compare garne
if match vyenw vne administrator
lai inform

INTRUSION DETECTION APPROACHES

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors.

Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders.

Thus, there is an element of compromise and art in the practice of intrusion detection.

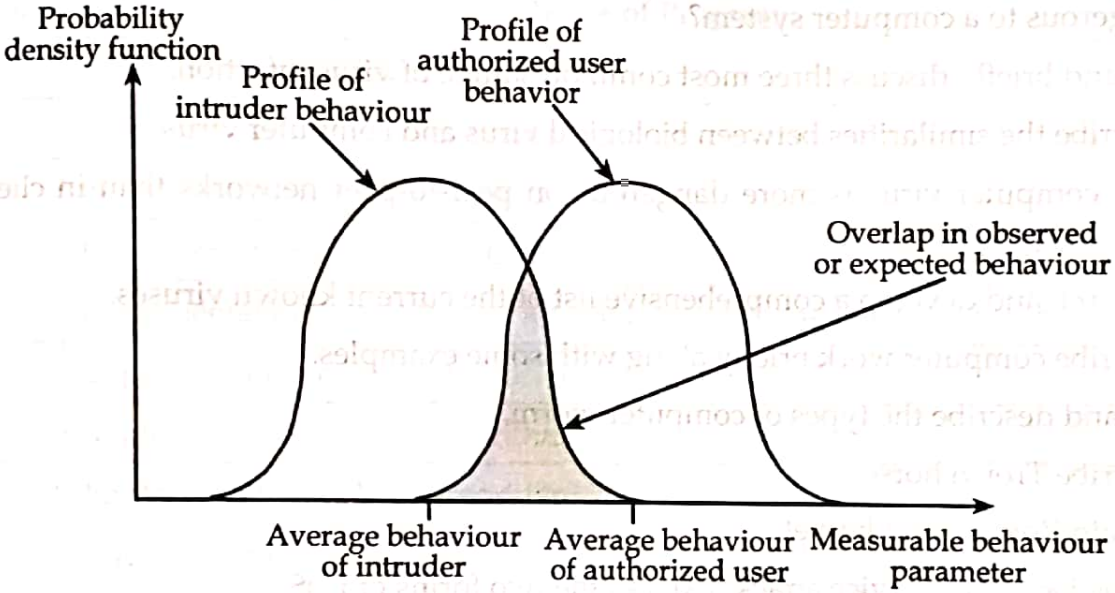

Figure 7.8: Profiles of Behavior of Intruders and Authorized Users

Following approaches are used for intrusion detection:

- **Statistical anomaly detection:** Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.
  - **Threshold detection:** This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
  - **Profile based:** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
- **Rule-based Detection:** Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
  - **Anomaly detection:** Rules are developed to detect deviation from previous usage patterns.
  - **Penetration identification:** An expert system approach that searches for suspicious behavior.

Discussion Exercise

1. What do you mean by malicious software? Describe some terminologies of malware.
2. Classify the malicious software according to the need of host machine.
3. What sort of malware is known as computer virus?
4. Write the phases of computer viruses along with pseudocode.
5. What are the major differences between a boot virus and a macro virus? Which is more dangerous to a computer system?
6. List and briefly discuss three most common source of virus infection.
7. Describe the similarities between biological virus and computer virus.
8. Why computer virus is more dangerous on peer-to-peer networks than in client-server networks?
9. Research and develop a comprehensive list of the current known viruses.
10. Describe computer work briefly along with some examples.
11. List and describe the types of computer worm.
12. Describe Trojan horse.
13. Explain Zombie and botnet.
14. Define Denial of Service attack. Explain the two forms of DoS.
15. What do you mean by Distributed Denial of Service (DDoS) attack?
16. Are IDSs similar to firewalls? Why are system intrusions dangerous?
17. Explain Intrusion Detection Approaches.

❑❑❑