

UNIT 6

Malicious Software and Intrusion

(4 Hours) — Network Security — Information Security (CACS459)

So far this course has looked at how to protect information using cryptography. But attackers don't only target the data in transit — they also try to get unwanted, harmful programs running on a victim's machine, or to break into a system directly as an unauthorised user. This unit studies both halves of that problem: malicious software (malware) — viruses, worms, and Trojan horses that run code an attacker wants run — and intrusion — attackers who try to gain unauthorised access to a system directly, and the detection systems (IDS) and decoys (honeypots) built to catch them.

6.1 Malicious Software

Malicious software (malware) is any software intentionally designed to cause damage, gain unauthorised access, steal data, or otherwise act against the interests of the computer user. Malware can be broadly classified along two independent dimensions: whether it needs a host program to live inside, and whether it can replicate itself to spread further.

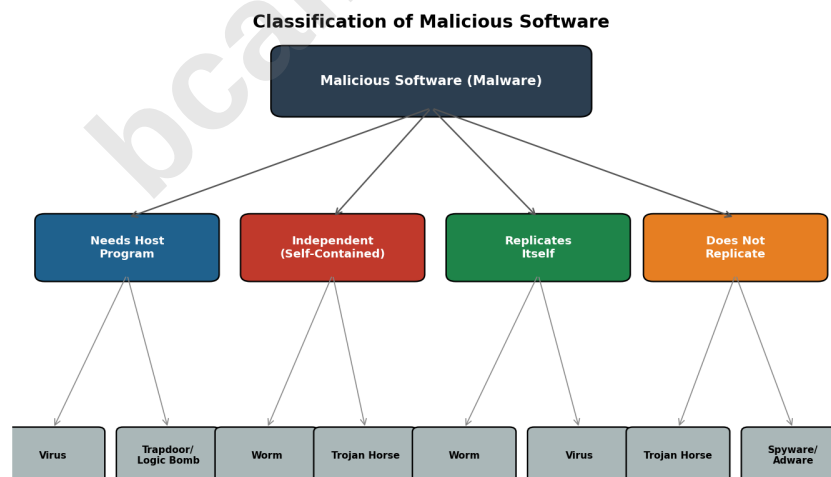


Fig. 6.1 — Malicious software classified by whether it needs a host program and whether it replicates itself; the same malware family can sometimes appear in more than one branch depending on its specific behaviour.

Category	Brief Description
Virus	Attaches itself to a host program/file and replicates when that host is executed; needs a host and replicates

Category	Brief Description
Worm	A standalone program that replicates itself and spreads across networks without needing to attach to a host program
Trojan Horse	A program that appears useful or legitimate but secretly performs unauthorised, malicious actions; does not replicate itself
Logic Bomb / Trapdoor	Code embedded in a legitimate program that triggers a malicious action under specific conditions (logic bomb), or a hidden, undocumented entry point that bypasses normal access controls (trapdoor)
Spyware / Adware	Software that secretly monitors user activity or displays unwanted advertising, generally without self-replication
Ransomware	Malware that encrypts a victim's files and demands payment for the decryption key

6.2 Virus, Its Phases, and Virus Classification

6.2.1 What is a Computer Virus?

A computer virus is a piece of code that attaches itself to a legitimate program (the host) and, when that program is executed, copies itself into other programs, potentially also performing some additional harmful action. A virus is therefore similar to a biological virus in that it cannot reproduce on its own — it needs a host to infect.

6.2.2 The Four Phases of a Virus

- **Dormant Phase:** the virus is idle, sitting inside its host program, waiting for an activating event (not all viruses have a noticeable dormant phase).
- **Propagation Phase:** the virus places identical (or possibly modified) copies of itself into other programs or specific areas of disk, infecting more hosts.
- **Triggering Phase:** the virus is activated to perform the function for which it was intended, triggered by a variety of system events (such as a particular date, the presence of another program, or a count of how many times it has self-replicated).
- **Execution Phase:** the actual payload function is performed — this could be harmless (e.g., displaying a message) or damaging (e.g., destruction of files/data).

The Four Phases of a Computer Virus's Life

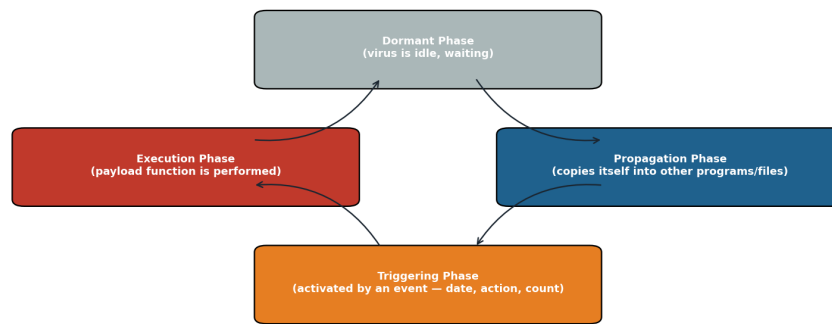


Fig. 6.2 — The four phases of a virus's lifecycle: dormant → propagation → triggering → execution, then potentially back to dormant in a newly infected host.

6.2.3 Virus Classification

Viruses can be classified along two axes: by what they infect (classification by target) and by the technique they use to avoid detection (classification by concealment strategy).

Classification (by Target)	Description
Parasitic Virus	Attaches itself to executable files and replicates whenever the infected program is run
Memory-Resident Virus	Lodges itself in main memory as part of a resident system program, infecting every program that subsequently executes
Boot Sector Virus	Infects the master boot record or boot sector of a disk, spreading whenever the system boots from that infected disk
Macro Virus	Exploits the macro/scripting features of applications such as word processors and spreadsheets; platform-independent and very common as an email attachment-based threat
Stealth Virus	Designed to actively hide itself from antivirus and detection software, e.g. by intercepting disk-access requests

Classification (by Concealment Strategy)	Description
Encrypted Virus	The virus body is encrypted (except for a small decryption routine), making the bulk of the code look different in each infected file, hindering signature-based detection
Polymorphic Virus	Mutates its decryption routine itself with every infection, so no two copies look exactly alike, while the underlying functionality stays the same

Classification (by Concealment Strategy)	Description
Metamorphic Virus	Goes further than polymorphic viruses — it rewrites its entire body (not just an encryption wrapper) with every infection, making detection by static signature extremely difficult

Note: Polymorphic and metamorphic viruses are specifically designed to defeat signature-based antivirus detection (Section 6.7) by ensuring that two copies of the 'same' virus rarely share an identical byte pattern.

6.3 Worm, Worm Propagation Model, and State of Worm Technology

6.3.1 What is a Worm?

A worm is a standalone malicious program that actively seeks out and exploits vulnerabilities to replicate itself across a network without needing to attach to a host program and without needing any direct user action to trigger each new infection (unlike a virus, which typically needs an infected file to be opened or run). A worm's life cycle mirrors a virus's (dormant → propagation → triggering → execution), but its propagation phase actively scans the network for new, vulnerable targets rather than infecting local files.

6.3.2 Worm Propagation Model

The spread of a worm across a population of vulnerable hosts is commonly modelled using a logistic (S-curve) growth model, similar to models used for epidemic disease spread. Early on, when very few hosts are infected, the worm spreads slowly simply because there are few infected hosts doing the scanning. As more hosts become infected, the rate of new infections accelerates rapidly (the steep middle section of the S-curve). Eventually, as the pool of remaining vulnerable, uninfected hosts shrinks, the growth rate slows again and the infection count approaches the total vulnerable population.

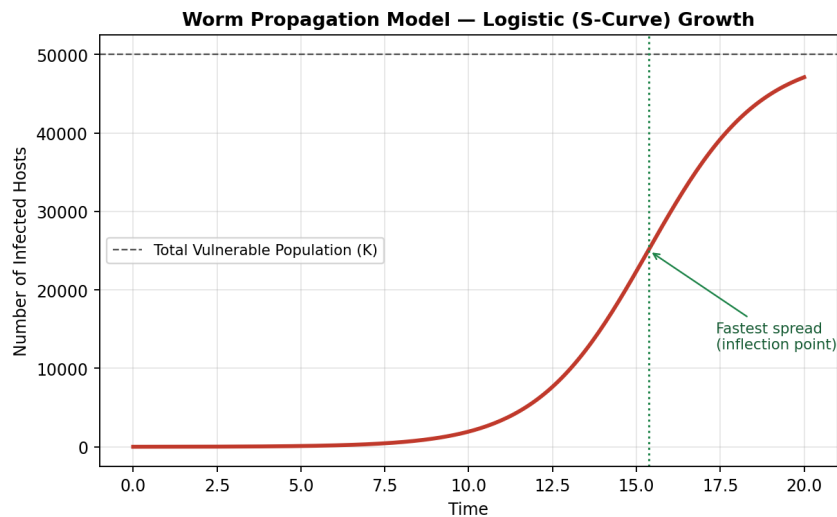


Fig. 6.3 — The classic logistic worm-propagation curve: slow initial spread, a rapid exponential-looking middle phase, then saturation as the vulnerable population is exhausted.

Key factors affecting how fast a worm spreads include: the scanning rate (how many new addresses each infected host probes per unit time), the size of the vulnerable population, the network topology and bandwidth, and any countermeasures (such as rate-limiting or patching) deployed as the outbreak is detected.

6.3.3 State of Worm Technology

Worm technology has evolved considerably since the earliest worms:

- **Multiplatform:** modern worms are increasingly written to target multiple operating systems and architectures, rather than just one.
- **Multi-exploit:** rather than relying on a single vulnerability, modern worms often carry several different exploitation techniques to maximise their chance of infecting any given target.
- **Ultrafast spreading:** techniques such as pre-generated target lists ('hit lists') let a worm infect a very large number of hosts within minutes of release, far faster than human responders can react.
- **Polymorphic and Metamorphic worms:** like their virus counterparts, modern worms increasingly change their own code with each new copy to defeat signature-based detection.
- **Zero-day exploits:** the most dangerous modern worms exploit previously unknown vulnerabilities for which no patch yet exists, maximising the window of opportunity before defences can be deployed.

Note: The transition from simple, single-exploit worms to multiplatform, multi-exploit, fast-spreading worms means that purely reactive defences (detect, then patch) are increasingly insufficient — proactive measures (network segmentation, rapid patch deployment, anomaly detection) are essential.

6.4 Trojan Horse

A Trojan Horse is a useful, or apparently useful, program or command procedure that contains hidden malicious code which, when invoked, performs unwanted or harmful functions in addition to (or instead of) its advertised purpose — named after the Trojan Horse of Greek mythology, which appeared to be a harmless gift but concealed armed soldiers inside.

Trojan Horse — Malicious Function Hidden Inside a Useful Program

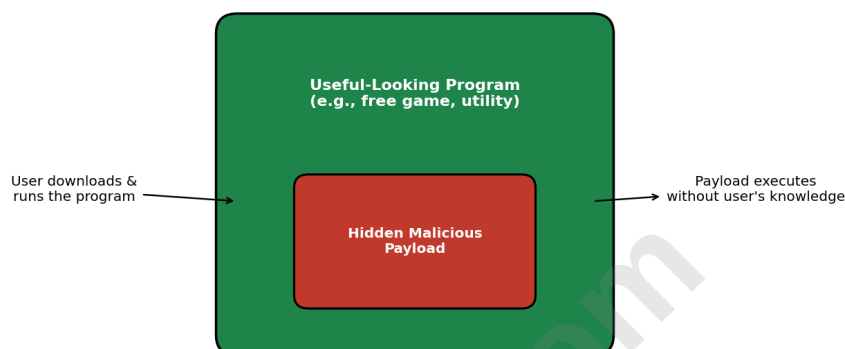


Fig. 6.4 — A Trojan horse disguises its malicious payload inside a program that appears legitimate or useful, relying on the user to voluntarily download and run it.

Unlike a virus or worm, a Trojan horse does not replicate itself; it relies entirely on social engineering to convince a user to install and run it voluntarily — for example, disguised as a free game, a pirated software 'crack', or a useful-looking utility. Once executed, common Trojan payloads include installing a backdoor (allowing remote attacker access), logging keystrokes, stealing stored credentials, or turning the infected machine into part of a botnet.

Note: Because a Trojan horse needs no self-replication mechanism, antivirus detection often relies heavily on behavioural analysis (Section 6.7) in addition to signature matching, since a well-disguised Trojan's static code can otherwise look completely benign.

6.5 Intrusion and Intruders

While malware focuses on getting unwanted code to run on a victim's machine, intrusion refers more broadly to any attempt by an unauthorised party to gain access to a computer system or network, whether through malware, stolen credentials, exploiting a software vulnerability directly, or simple guesswork.

6.5.1 Classification of Intruders

Type	Description
Masquerader (Outsider)	An individual who is not authorised to use a system at all, but penetrates it by exploiting a legitimate user's account, typically by stealing credentials
Misfeasor (Insider)	A legitimate user who accesses data, programs, or resources for which such access is not authorised, or who is authorised but misuses their privileges
Clandestine User	An individual who seizes supervisory/administrative control of a system and uses this control to evade auditing/access controls or to suppress audit collection altogether

Intrusion techniques generally aim either to obtain protected information that should have been protected (such as a password file) or to gain privileges beyond those the intruder is legitimately granted (privilege escalation). Common methods include password guessing or cracking, exploiting unpatched software vulnerabilities, and social engineering.

6.6 Intrusion Detection System (IDS)

An Intrusion Detection System (IDS) is a hardware or software system that monitors network traffic or system activity for signs of malicious activity, policy violations, or intrusion attempts, and generates alerts when such activity is detected. An IDS is purely detective in nature (it alerts; it does not on its own block traffic, unlike a firewall or an Intrusion Prevention System, IPS).

6.6.1 Types of IDS by Placement

- **Network-Based IDS (NIDS):** monitors network traffic for an entire segment or subnet, typically placed at a strategic network chokepoint (such as just behind a firewall), examining packets as they pass.
- **Host-Based IDS (HIDS):** runs on an individual host, monitoring that machine's internal activity — system calls, file system changes, log files, and so on — able to detect attacks that NIDS might miss (such as those carried out over encrypted channels or by a local insider).

Intrusion Detection System (IDS) Placement: Network-Based vs Host-Based

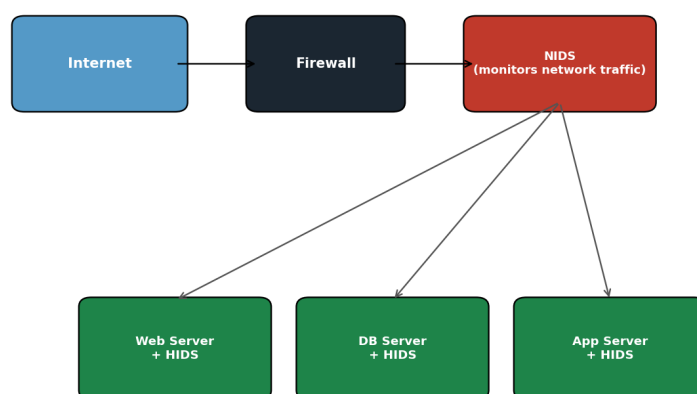


Fig. 6.5 — A typical layered IDS deployment: a Network-Based IDS monitors traffic as it crosses the network perimeter, while Host-Based IDS instances monitor activity directly on each individual critical server.

6.6.2 Components of an IDS

- Sensors: collect raw data (network packets, log entries, system call traces).
- Analysers: process the collected data to determine whether an intrusion has occurred, using the detection approaches in Section 6.7.
- User interface / Console: lets a security administrator view alerts, configure the system, and respond to detected events.

6.7 Analysis Approaches: Anomaly-Based and Signature-Based

IDS analysers generally use one (or a combination) of two fundamentally different detection approaches.

6.7.1 Signature-Based Detection

Signature-based detection (also called misuse detection) maintains a database of known attack patterns ('signatures') — specific byte sequences, packet structures, or system call sequences known to be associated with a particular attack. Incoming traffic or activity is compared against this database; a match triggers an alert.

6.7.2 Anomaly-Based Detection

Anomaly-based detection first builds a profile of normal behaviour for a system or network (using statistical methods or machine learning over a training period), then flags any activity that significantly deviates from that established baseline as potentially malicious, regardless of whether it matches a known attack pattern.

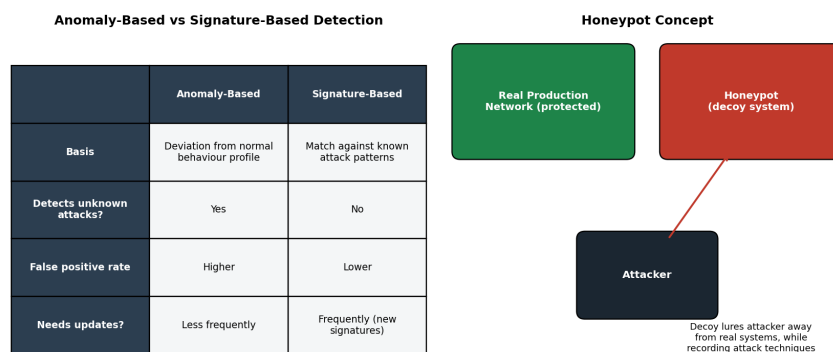


Fig. 6.6 — Left: comparison of anomaly-based and signature-based detection approaches. Right: the honeypot concept — a decoy system used to lure and study attackers away from real production systems.

Note: Signature-based detection is fast and produces few false alarms but is blind to new ('zero-day') attacks with no existing signature; anomaly-based detection can catch novel attacks but tends to generate more false positives, since legitimate but unusual behaviour can also look anomalous. Most modern IDS products combine both approaches.

6.8 Honeypots

A honeypot is a decoy computer system, deliberately set up to look like an attractive, vulnerable target, with no legitimate production purpose of its own. Its entire value comes from being attacked: any activity directed at a honeypot is, by definition, unauthorised (since no legitimate user has any reason to access it), making honeypots an extremely low-noise source of intelligence about attacker behaviour, tools, and techniques.

- Low-interaction honeypots: simulate only specific services or protocols, limiting what an attacker can actually do, and are simpler and safer to operate.
- High-interaction honeypots: provide a real, fully functioning operating system and services for the attacker to interact with, yielding much richer intelligence on attacker behaviour, but at higher operational risk and complexity (since the attacker has more genuine capability inside the decoy).

Honeypots serve two complementary purposes: they can divert an attacker's attention and effort away from genuinely valuable production systems, and they let defenders study exactly what techniques, tools, and exploits an attacker is using in a controlled, observed environment, which can then inform signatures and defences for the real network.

Note: A network of multiple, coordinated honeypots spread across an organisation (or even the wider internet) is called a honeynet, used to gather broader threat intelligence on attack trends rather than just individual incidents.

Key Takeaways — Unit 6: Malicious Software and Intrusion

- ✓ Malware is classified by whether it needs a host program (virus, logic bomb) versus being independent (worm, Trojan, most spyware), and by whether it replicates (virus, worm) versus not (Trojan horse, most spyware).
- ✓ A virus moves through four phases — dormant, propagation, triggering, execution — and is classified by target (parasitic, memory-resident, boot sector, macro, stealth) and by concealment strategy (encrypted, polymorphic, metamorphic).
- ✓ A worm replicates itself across a network without needing a host file or user action; its spread typically follows a logistic (S-curve) growth model, and modern worm technology has become multiplatform, multi-exploit, and increasingly zero-day-driven.
- ✓ A Trojan horse hides malicious functionality inside an apparently useful program and relies entirely on social engineering, since it does not self-replicate.
- ✓ Intruders are classified as masqueraders (outsiders using stolen credentials), misfeasors (insiders misusing legitimate access), or clandestine users (those who seize administrative control to hide their tracks).
- ✓ An Intrusion Detection System (IDS) monitors network traffic (NIDS) or individual hosts (HIDS) for signs of intrusion, using signature-based detection (matches known attack patterns, fast but blind to new attacks) and/or anomaly-based detection (flags deviations from a learned normal-behaviour baseline, catches novel attacks but with more false positives).
- ✓ Honeypots are decoy systems with no legitimate purpose other than to attract and study attackers, diverting them from real systems while gathering intelligence on their methods.

— End of Unit 6 —