CHAPTER 4
Data and Application Security
Part 2

Dr. Mohamed Marie

Topics in this Chapter:

- Types of Applications (Part 1)
- Application Models and Technologies (Part 1)
- **Application Threats and Countermeasures (Part 2)**
- Security in the Software Development Life Cycle (Part 3)
- Application Security Controls
- Databases and Data Warehouses

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow √
2. Malicious software
3. Input attacks
4. Logic bombs
5. Object reuse
6. Mobile code
7. Social engineering
8. Back door

Threats in the Software Environment

1. Buffer Overflow

- Software applications usually function by soliciting and accepting input from a user (or another application) through an interface. An attacker can attempt to disrupt the function of a software application by providing more data to the application than it was designed to handle. A buffer overflow attack occurs when someone attempts to disrupt a program's operation in this manner.

Threats in the Software Environment

1. Buffer Overflow

Types of Buffer Overflow Attacks There are several specific types of buffer overflow attacks, discussed here.

- **Stack Buffer Overflow:** In this type of attack, the program writes more data to a buffer located on the stack than was allocated for it. This causes the corruption of other data in the stack, which results in the program's malfunction.
- **NOP Sled Attack:** The NOP sled attack is a specific stack overflow attack where the attacker overflows the stack with harmless NOP (no-op) instructions. The point of the NOP sled attack is to improve the chances that the attacker will be able to find an attack point. By flooding the stack with lots of NOPs, the program will encounter and "slide down" the NOPS until it reaches the pointer that the attacker placed in the buffer.
- **Heap Overflow:** The heap is the dynamically allocated memory space created by a program for storage of variables. Usually a heap overflow attack will result in the corruption of other variables that are already on the heap.
- **Jump-to-Register Attack:** In this attack, the return pointer is overwritten with a value that will cause the program to jump to a known pointer stored in a register that points to the input buffer.

Threats in the Software Environment

1. Buffer Overflow
Buffer Overflow Countermeasures:

- Several tactical and strategic countermeasures are available to reduce or eliminate the risk of buffer overflow attacks. Buffer overflow countermeasures are used to either remove buffer overflow capabilities or detect and block buffer overflow activity.
  - **Choose a safe language.** C and C++ do not automatically check input buffer lengths or perform other boundary checking. Java, .NET, and many other languages have built-in boundary checking that—in most cases—prevents buffer overflows.
  - **Use of safe libraries**
  - **Executable space protection.** A feature of some operating systems, forces programs to abort if they attempt to execute code in the stack or the heap. Some CPUs support executable space protection in hardware.
  - **Stack smashing protection.** Refers to techniques used to detect changes in the stack.
  - **Application firewalls**

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software √
3. Input attacks
4. Logic bombs
5. Object reuse
6. Mobile code
7. Social engineering
8. Back door

Threats in the Software Environment

2. Malicious Software

Malicious software, also known as malicious code, is a class of software that comes in many forms and performs a variety of damaging actions. **The purposes of malware include:**

- **Propagation.** Sometimes the ability for malware to propagate—that is, to spread from system to system—is the only purpose for particular malware programs.
- **Damage and destruction of information**. Malware can alter or delete files on target systems.
- **Steal information.** Malware can locate and steal valuable information such as e-mail addresses, userids and passwords, bank account numbers, and credit card numbers.
- **Usage monitoring.** Malware can implant the means to record subsequent communications, keystrokes and mouse clicks, and send this data back to the malware's owneroperator.
- **Denial of Service.** Malware can consume all available resources on a target system, rendering it essentially useless for its intended use.
- **Remote control**. Malware can implant a **bot** onto a target system that allows an attacker to remotely control the system. Large collections of bots are called ***bot armies***, and the people who build and control bot armies are known as ***bot herders or botnet operators***.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

- Viruses
- Worms
- Trojan horses
- Rootkits
- Bots
- Spam
- Pharming
- Spyware and Ad-ware

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Viruses:

- Viruses are **computer code** fragments that **attach themselves** to a legitimate program file on a computer. The virus can only run when the legitimate program is run. viruses generally require human intervention to propagate. A user must run a program in order to make the virus spread. Viruses used to propagate through file sharing.

Several types of viruses are discussed here:

- **Master Boot Record (MBR) viruses.** Viruses attach themselves to the master boot record.
- **File infector viruses.** These are the viruses that attach themselves to executable programs (.EXE and .COM files)
- **Macro viruses.**

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Viruses:

Viruses employ several methods to avoid detection by anti-virus programs. **The methods in use include:**

- **Multipartite viruses.** These use more than one means for propagating from one system to another. For example, Ghostball infected both executable .COM program files as well as boot sectors.
- **Stealth viruses.** A stealth virus uses some means to hide itself from detection from the operating system.
- **Polymorphic viruses.** Virus creators have introduced polymorphic viruses that change themselves as they move from system to system in order to avoid detection. However, engineers in the anti-virus companies are able to solve the puzzle of polymorphic viruses and create a signature for them.
- **Encrypted viruses.** Viruses will encrypt most of their code, using a different key on each system they infect, which makes most of the body of the virus different on each detected system. However, a part of the virus—the decryption code—must remain the same; it is this portion of the virus that the antivirus software must be able to identify in order to stop the virus.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Worms:

Generally speaking, worms are like viruses, but they usually require little human intervention to spread. Instead, they have their own means of propagation built-in. Two common types of worms that are found today include:

- **Mass mailing worms.** Mass mailing worms propagate via e-mail. Generally, when a mass-mailing worm arrives in a user's inbox, the worm is activated when the recipient opens the message. The worm's malicious code could reside within the HTML code in the message, or in an attached file.
- **Port scanning worms.** A port scanning worm is able to propagate with no human intervention at all. A port scanning worm scans the network for other systems that may be vulnerable and attempt to spread to those neighboring systems. If it's able to infect a new system, it will install itself and begin the scanning to look for new victims.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Trojan Horses:

Like the ancient Greek legend, a computer-based Trojan horse is a lie.

- A Trojan horse claims to be one thing, but is instead something else—something with more malicious intent.
- The user who runs a Trojan horse program may or may not see some visual resemblance of what the program claims to be. However, the Trojan horse is also performing some additional (and probably malicious) action. It might be corrupting or destroying files, stealing data, or sending e-mails to your friends.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Rootkits:

- One of the newest forms of malware, rootkits are malware programs that are designed to avoid detection by being absolutely invisible to the operating system. Rootkits achieve this by altering the OS itself so that its presence is nearly impossible to detect.

Methods used by rootkits to avoid detection include:

- **Process hiding.** Rootkits can hide their own process(es) from users by altering the tools that are used to list processes on a system.
- **File hiding.** Rootkits can hide files as a way of avoiding detection.
- **Registry hiding.** Rootkits can hide registry entries in an attempt to function without being detected.
- **Running underneath the OS.** Rootkits can hide from the OS by running underneath or beside it.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Bots:

- Short for "**robots**," bots are sometimes a part of the malicious payload found in malware. Bots enable a "bot herder" (the owner of the bot program) to remotely control the infected computer for a variety of purposes including:
- **Relaying spam.** A technique that spam blockers use to block spam by blocking all e-mail from specific IP addresses.
- **Hosting phishing sites.**
- **Denial of Service attacks.** Bot herders can launch **Denial of Service (DoS)** attacks from bot-controlled systems by instructing those systems to launch thousands of network messages per second to a target system. A bot herder can launch a **distributed Denial of Service (DDoS)** attack by directing hundreds, thousands, or tens of thousands of bot-systems to attack the same target simultaneously.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Spam:

- **Spam is unwanted e-mail**. It accounts for well over ninety percent of all e-mail on the Internet. But more than that, spam is unsolicited **"junk mail"** that takes many forms including:
- **Unsolicited commercial e-mail (UCE).** Although UCE is also used to **market** even legitimate goods and services, users often frown on this type of advertising and frown at companies that advertise in this way.
- **Phishing.** These **two-part attacks** consist of legitimate-looking e-mail messages from large and well-known organizations (often financial institutions) that use some means to trick a user into visiting a web site. This web site will resemble a legitimate site and ask for login or other credentials or information such as credit card numbers, social insurance, numbers, or other information that will be used to defraud the user. The most common phishing scams purpose to come from banks that ask users to log in and confirm account numbers or credit card numbers.
- **Malware.** Spam is often used to directly deliver malware to users' computers, but it is also often used to lure people to web sites that contain malicious code in the form of **viruses, worms, or bots**.

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Pharming:

- In a pharming attack, an attacker directs all traffic destined for a particular web site towards an imposter web site. The attack diverts traffic by "poisoning" the organization's DNS servers or by changing the hosts file on individual users' systems.

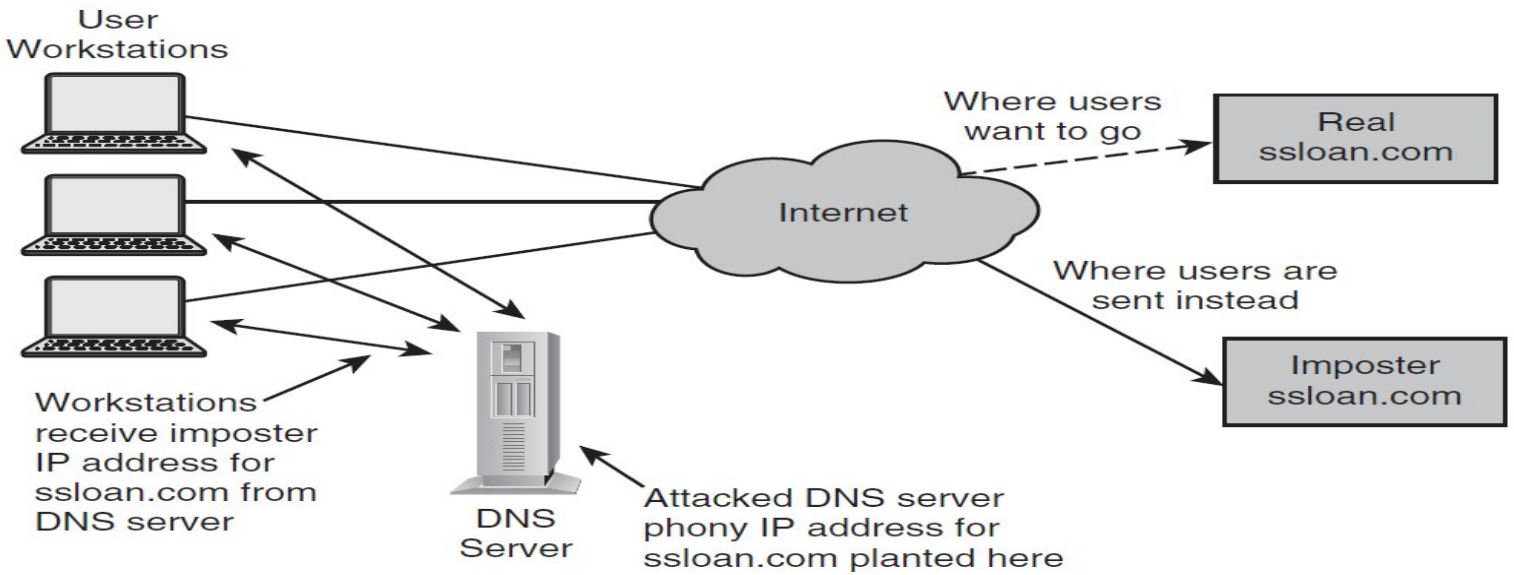

Figure 3-5 Pharming attack redirects users to a phony application server

Threats in the Software Environment

2. Malicious Software

Types of Malicious Software:

Spyware and Adware:

- Spyware and adware encompass a wide variety of means that have been developed to **track the behavior of users' Internet usage patterns**. While not strictly malicious, many find the techniques and motives used by spyware and ad-ware to be suspicious and an invasion of their privacy.

Spyware and adware take on many forms including:

- **Tracking cookies.** Many web site operators will **track users' individual visits to web sites** through the use of tracking cookies that may accompany banner ads.
- **Web beacons.** Sometimes known as "**web bugs**," web beacons are **tiny 11 pixel images that are embedded in web pages** as a means for tracking users' Internet usage. **An alternative to cookies**, web beacons are far **more difficult to detect and block**.
- **Browser helper objects (BHOs).** Sometimes they take the form of **helpful toolbars**, but at other times they are completely **invisible and "stealthy**." BHOs can be used to track use of users' Web browsers.
- **Key loggers.** A key logger actually **records a user's keystrokes** (and, often, mouse movements and clicks) and **transmit that data back** to a central location.

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Several measures are needed to block the ability for malware to enter and run on a system. These countermeasures include:

- Anti-virus
- Anti-rootkits
- Anti-spyware
- Anti-spam
- Firewalls
- Decreased privilege levels
- Penetration testing
- Hardening

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Anti-virus:

- Anti-virus programs run on a system and employ various means to **detect** the possible entry of malware and have the ability to **block** its entry. Anti-virus software can also **remove** malware if it is already present on the system.
- Anti-virus programs are found in many places in an organization as part of a defense in depth to prevent the unwanted consequences of malware. **The places where anti-virus software can be found include:**
  - **End user workstations.**
  - **E-mail servers.**
  - **File servers.**
  - **Web proxy servers.**
  - **Security appliances. All-in-one security appliances** that perform several functions including firewall, web content filter, spam filter, and anti-virus.

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Anti-rootkit Software:

- Anti-rootkit software uses techniques to find **hidden processes**, **hidden registry entries, unexpected kernel hooks, and hidden files** in order to find rootkits that may be present on a system. Anti-rootkit software programs use various means to find these hidden objects in a system.

Anti-spyware Software:

- **Software to block spyware and adware is similar to ant-ivirus software**: it monitors incoming files and examines them against a collection of signatures, and **blocks** those files that match known signatures. Like anti-virus software, anti-spyware can scan a hard drive to identify spyware, adware, and other unwanted programs, and remove them as directed by the user.

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Anti-spam Software:

- Spam blockers effectively eliminate most of the spam coming in to an organization, blocking the majority of the unwanted e-mail that carries malware, phishing scams, fraudulent advertising, and porn.
- **There are four common spam blocking architectures in use, including:**
  - **Client-based.**
  - **E-mail server-based.**
  - **Appliance-based.**
  - **Spam blocking service.** In this model, incoming e-mail is delivered to an off-site spam blocking service provider that **filters out** the spam and delivers only legitimate e-mail to corporate e-mail servers.

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Firewalls:

- Firewalls are the time-tested and still-preferred means for blocking unwanted network traffic from crossing a network boundary. Firewalls are typically used as perimeter devices, protecting organizations from unwanted traffic that originates from the Internet.

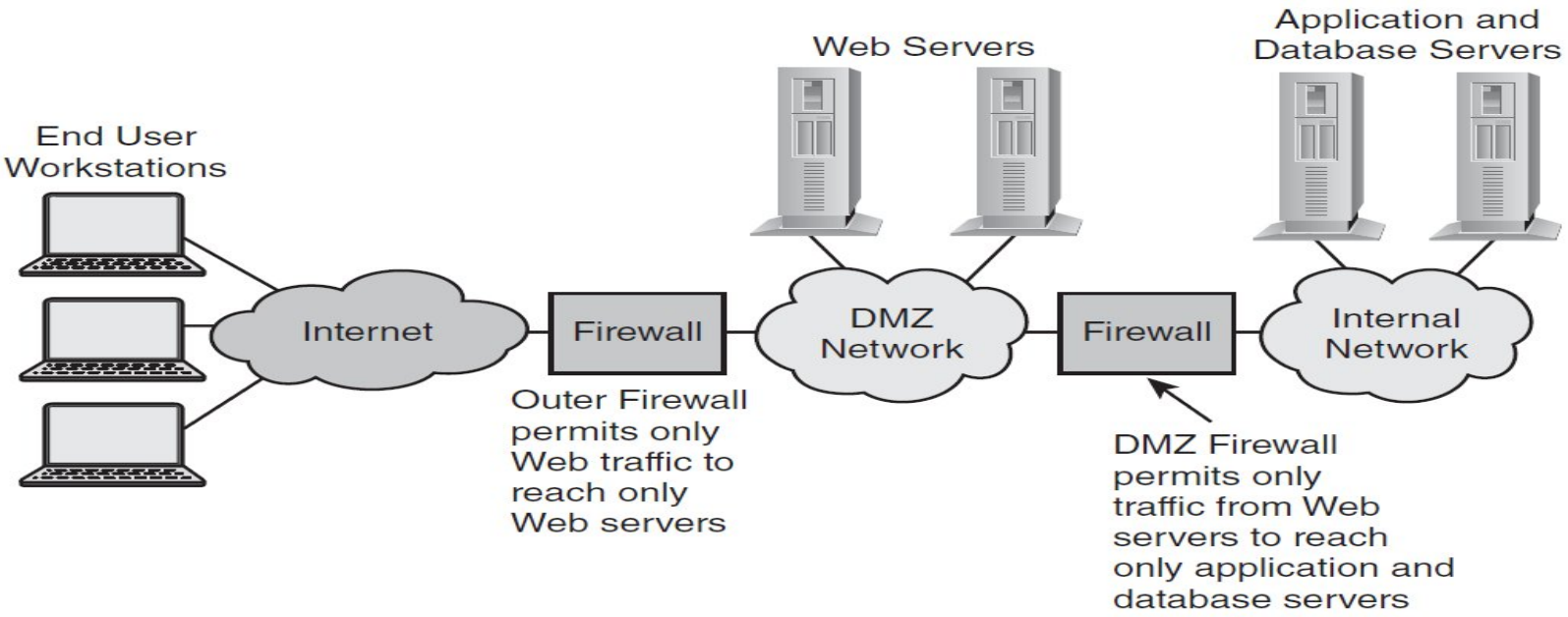

Figure 3-6 Typical DMZ network architecture protected by firewalls

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Decreased Privilege Levels:

- When malware successfully breaks into a system and is executed by the user, the malware usually is executing with the same privilege level as the user.
- A side benefit of reducing user privileges to end user level is a decreased number of tech support calls to repair uh-oh's, when often-inexperienced end users muddle up operating system configurations.

Threats in the Software Environment

2. Malicious Software

Malicious Software Countermeasures:

Penetration Testing:

- Rather than simply relying upon security configuration settings, an organization should also test the settings by using tools to simulate a hacker's attempt to find weaknesses in a system. Such tests are known as penetration tests, often known as **"pen tests**." The object of penetration testing is to discover and fix vulnerabilities before a hacker is able to discover and exploit them.

Hardening:

- Server operating systems are very complex and often are pre-configured for a wide variety of tasks. This often means that many of the programs and features that are available are activated by default. The result is a server with its necessary feature(s) activated, plus many additional unnecessary features also activated and ready to accept input from any friendly or unfriendly party.

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software
3. Input attacks √
4. Logic bombs
5. Object reuse
6. Mobile code
7. Social engineering
8. Back door

Threats in the Software Environment

3. Input Attacks

- Applications and tools often request input from users. A common method of attacking an application is to provide data that causes unexpected behavior in the application.
- **Input attacks**—sometimes called **malformed input attacks** or **injection attacks**—are designed to exploit weaknesses in the application by causing unexpected behavior including:
  - **Elevation of privileges.** The attacker will input specially coded data in an attempt to cause a malfunction that will result in the attacker having a higher level of access or privilege in the application.
  - **Execution of arbitrary code.** The attacker may wish to run specific commands on the target system.
  - **Malfunction.** The attacker may wish to cause the application to malfunction and be in a disabled state for legitimate users.
  - **Abort.** The attacker may wish to cause the application to completely abort and thus be unavailable for any legitimate use.

Threats in the Software Environment

3. Input Attacks

Types of Input Attacks:

Several types of input attacks including:

- **Buffer overflow.**
- **Integer overflow.** An attack where the attacker attempts to cause an application to perform an integer operation that will create a numeric value larger than can be represented in the available storage.
- **SQL injection.** In this type of attack, the attacker inserts specially coded and delimited SQL statements into an input field in the hopes that the injected SQL will be executed on the back end.
- **Script injection.** Similar to SQL injection, an attacker inserts script language into an input field in the hopes that the scripting language will be executed.
- **Cross-site scripting (XSS).** An attack where an attacker can inject a malicious script into **HTML content** in order to steal session cookies and other sensitive information.
- **Cross-site request forgery (XSRF).** This is an attack where **malicious HTML is inserted into a Web page or e-mail that**, when clicked, causes an action to occur on an unrelated site where the user may have an active session.

Threats in the Software Environment

3. Input Attacks

Input Attack Countermeasures:

Measures that can be used to prevent input attacks include:

- **Effective input field filtering.** Input fields should be filtered to remove all characters that might be a part of an input injection.
- **Application firewall.** Application firewalls examine the **contents of packets** and **block packets** containing input attack code and other unwanted data.
- **Application vulnerability scanning.** Organizations that develop their own applications for online use should scan those applications for **input attack vulnerabilities**, in order to identify vulnerabilities prior to their being discovered and exploited by outsiders.
- **Developer training. Software developers should be trained in secure application development techniques**.

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software
3. Input attacks
4. Logic bombs √
5. Object reuse
6. Mobile code
7. Social engineering
8. Back door

Threats in the Software Environment

4. Logic Bomb

- **Logic bombs**, sometimes known as **time bombs**, are instructions deliberately placed in application code that perform some hostile action when a predetermined condition is met. Typically a logic bomb consists of code that performs some damaging action on a date in the distant future. Most often, a developer will plant a logic bomb in an application if he believes he will be terminated from employment. The logic bomb will "go off" at some later date, and the terminated programmer will feel that he got his just revenge.

Logic Bomb Countermeasures:

- **Logic bombs and back doors are very similar**: both involve unwanted code in an application. The countermeasures for logic bombs are the same as for back doors: **code reviews**, **source code control**, **source code scanning**, and **third party assessments**.

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software
3. Input attacks
4. Logic bombs
5. Object reuse √
6. Mobile code
7. Social engineering
8. Back door

Threats in the Software Environment

5. Object Reuse

- Many system resources are shared in multiprocessing systems. This includes memory, databases, file systems, and paging space. When one process utilizes a resource, the process may write some information to the resource temporarily.

Object Reuse Countermeasures:

- **Application isolation.** Applications should be isolated to individual systems. In this way, applications are less likely to encounter residual information left by other applications.
- **Server virtualization.** Often it is not feasible to isolate applications to one-per-machine. However, virtualization technology may make it more cost-effective to isolate applications by running them on virtual machines.
- **Developer training.** Software developers can be shown how to write secure software that does not leave residual code that can be used by other processes.

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software
3. Input attacks
4. Logic bombs
5. Object reuse
6. Mobile code √
7. Social engineering
8. Back door

Threats in the Software Environment

6. Mobile Code

- Also known as **executable code**, **active content**, and **downloadable content**, mobile code can be downloaded or transferred from one system for execution on another system. **Examples of mobile code include:**
  - **Active website content.** This includes **ActiveX**, **Java**, **JavaScript**, **Flash**, **Shockwave**, and so on. This content originates on a Web server and executes on a user's workstation. Depending upon the technology associated with the downloaded content, this mobile code may have restricted access to the end user's system or may have full control over it.
  - **Downloaded software.** This includes software of every kind from legitimate (and not-so-legitimate) sites. Some of this software may be purely benign, but others can be Trojan horse programs and worse. Some is outright malware, with or without a disguise.

Threats in the Software Environment

6. Mobile Code

Mobile Code Countermeasures:

- **Anti-malware.** This includes anti-virus, anti-spyware, and so on. These protective programs should be in place, properly configured, and up-to-date.
- **Reduced user privileges.** End users should not be permitted to install or execute mobile code on their workstations, except in explicitly permitted situations such as company produced mobile code.
- **Mobile code access controls.** Access controls should be in place to prevent unauthorized persons from downloading any mobile code that they are not permitted to access or use.
- **Secure workstation configuration.** Workstations should be configured to restrict mobile code except in cases where specific mobile code is permitted. This may involve centralized workstation configuration that cannot be defeated or circumvented by end users.

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software
3. Input attacks
4. Logic bombs
5. Object reuse
6. Mobile code
7. Social engineering √
8. Back door

Threats in the Software Environment

7. Social Engineering

- A **social engineering attack** is an attack on the personnel in an organization. Usually the purpose of a social engineering attack is to gain secrets from individuals that can later be used to gain unauthorized access to the organization's systems. The social engineer uses a technique known as pretexting in an effort to pretend that they are someone else.
- Social engineering owes its success to basic human nature: people are willing to help others in need and "be the hero." Social engineers prey on this weakness in feigned calls for assistance.

Social Engineering Countermeasures:

- The best countermeasure against social engineering is **education**: people in the organization, particularly those with administrative privileges (system administrators, network administrators, database administrators, and so on), need to be educated on the proper procedures for providing company sensitive information to others.

Threats in the Software Environment

The threats to software applications discussed in this section include:

1. Buffer overflow
2. Malicious software
3. Input attacks
4. Logic bombs
5. Object reuse
6. Mobile code
7. Social engineering
8. Back door √

Threats in the Software Environment

8. Back Door

- A **back door** is a mechanism that is deliberately planted in a system by an application developer that allows the developer or other person to circumvent security. Back doors may be present in an application for several reasons including:
  - **To facilitate testing during application development.** For instance, back doors can be activated by entering specific values that will cause the program to enter an interactive debug mode.
  - **To facilitate production access.** For example, a back door is created so that a developer can access an application while it is in production. This would be considered an inappropriate use of a back door, since developers should never have access to a production application or production data.
  - **To facilitate a break-in**. Sometimes back doors are inserted into an application to permit an unauthorized party to access application functions or data that the party should not have access to. This is clearly inappropriate. This use resembles a **logic bomb**.

Threats in the Software Environment

8. Back Door

Back Door Countermeasures:

- Back doors can be difficult to find, particularly if they are inserted for disreputable purposes. **Routine functional testing and QA testing** may not reveal back doors, whatever their purpose. Instead, other means are required to find them, including (same as Logic Bomb) :
  - **Code reviews.** When one developer makes changes to a software application, one or more other developers should examine the software to identify and approve of all changes. This should prevent both the "legitimate" back doors as well as illegitimate ones.
  - **Source code control.** A formal source code management system should be used that will identify and record all changes made to the code.
  - **Source code scanning.** Tools that are used to scan static source code for security vulnerabilities should be able to find back doors (or at least flag the unusual logic associated with a back door).
  - **Third-party code reviews and assessments.** A third-party organization will be more motivated to find anomalies in software than its own developers.