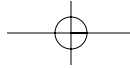


CHAPTER 5

METHODS OF ATTACK

“To an extent, it was through magic that I discovered the enjoyment in fooling people.”—Kevin Mitnick [1]

This chapter will present some of the common attack techniques that are used to compromise computer systems. Attacks often exploit security design flaws, but not always. For example, a “denial of service” attack that interrupts or overwhelms an application can be successful in putting an impregnable system out of service. In general, attacks exploit any form of technical or human weakness. Technical weaknesses can include design flaws, implementation flaws, inadequate protection features, and environmental changes or weaknesses. Human weaknesses can include poor usage practices, inexperienced or inadequately trained users, and poor physical security.



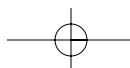
The discussion here focuses first on attacks against technical weaknesses, which are referred to here as technical attacks. The discussion then turns to the methods used against humans in order to obtain computer access. The latter is often referred to as “social engineering.”

5.1 Technical Attacks

The enumeration of attack patterns provided in this section is not exhaustive, as the patterns that are possible are limited only by the ingenuity of attackers. It is important to understand these basic patterns as a precursor to examining the software design principles presented afterwards, so that the purpose and motivation of those principles can be appreciated. The attack patterns presented here are not intended to be mutually exclusive or orthogonal. In fact, many of them overlap or are related, and real attacks often fall into more than one pattern or use multiple techniques in combination.

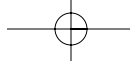
The attack patterns presented here are based primarily on the work of others (for example, [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]). The categories here are arranged in a way that is, hopefully, relevant to the way that most software developers think. This book’s Web site¹ contains references to sources and other work related to vulnerability and attack taxonomies.

¹ This book’s Web site can be found at <http://assuredbydesign.com/haa>.



TECHNICAL ATTACKS, BY SECTION

Interception	5.1.1
Related: Sniffing. Covert channel.	
Man-in-the-Middle	5.1.2
Replay	5.1.3
Modification in Place or in Transit	5.1.4
Related: File manipulation. MiM.	
Interruption	5.1.5
Saturation and Delay	5.1.6
Related: Denial-of-service	
Exploitation of Non-Atomicity	5.1.7
Related: TOCTTOU	
Coordination Interference	5.1.8
Forced Crash and Retrieval of Crash Artifacts	5.1.9
Forced Restart, Forced Re-Install	5.1.10
Environmental Interference	5.1.11
Spoofing	5.1.12
Hijacking	5.1.13
Circumvention	5.1.14
Trap Door	5.1.15
Exploit of Incomplete Traceability	5.1.16
Related: Non-repudiation	
Exploit of Incomplete Validation	5.1.17
Related: Buffer overflow	
Exploit of Incomplete Authentication or Authorization	5.1.18
Exploit of Exposure of Internal State	5.1.19
Exploit of Residual Artifacts	5.1.20
Related: "Dumpster diving" or "trashing"	
Embedded Attack	5.1.21
Related: Planting. Trojan horse. Time bomb. Logic bomb.	
Pattern Matching and Brute Force Guessing	5.1.22
Related: Exhaustive search	
Namespace Attack	5.1.23
Weak Link as Gateway	5.1.24
Trusted Resource Attack	5.1.25
Scope Attack	5.1.26
Related: Domain errors	



5.1.1 Interception

Attacks that involve any form of subversive *interception* of information can be categorized as either “eavesdropping” or “sniffing.” The term “sniffing” usually refers specifically to non-intrusive and often undetectable interception, such as by reading information that is broadcast or by attaching a passive listener to a communication channel. The term “eavesdropping” is a less technical term and applies more broadly and loosely.

This category of attacks often involves the use of a “covert channel.” A *covert channel* is any communication pathway that exists but was not intended by the designers of the system and thereby violates the system’s security policy. [13] A covert channel need not be an actual mechanism intended for *any* form of communication at all; for example, the technique of varying the load on a CPU has been used as a covert channel for the binary encoded signaling of sensitive information to another process in an undetected manner.

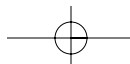
Notorious examples of eavesdropping or sniffing attacks include:

1. Sending ICMP packets in order to re-direct packets to flow through an attacker’s system, thereby allowing the attacker to read the data in the packets.
2. Reading the mailbox message files in a POP server.
3. Installing a keyboard handler that silently listens to keystrokes.
4. Reading someone’s screen on an X-Windows system or on a Windows system via a program such as RealVNC.

When an application reads the information stored by another application, this is often also referred to as “interference,” especially if that information is then acted upon in a subversive manner or is used to interfere with the operation of the other application.

5.1.2 Man-in-the-Middle

When a party succeeds in interposing itself between two endpoints and is thereby able to intercept and possibly modify the communication without either party being aware, this is referred to as a “man-in-the-middle” (MiM) attack.



MiM is related to interception, but requires that the interception occurs as the result of the interposition of a listener rather than strictly passive eavesdropping.

5.1.3 Replay

Replay involves the interception of information intended for a target system, followed by sending that information—possibly with additional information inserted—to the target system for the purpose of attacking the system.² Replay is a form of MiM attack in which the intercepted message is not modified, although it may be augmented.

Replay often (but not always) involves the use of an intercepted bearer credential of some kind, such as a password or session credential. If an attacker intercepts information that is used to access a resource, such as credentials, the attacker might be able to impersonate a trusted party and thereby access the resource. In this scenario, the attacker “replays” the intercepted information, leading the receiver to believe that the attacker is a trusted party. An example of this kind of replay is intercepting someone’s browser session cookie or authentication header and using it to masquerade as the user’s session.

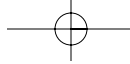
Replay may be coupled with a timing attack (see Section 5.1.6, “Saturation and Delay”) against a credential-validation server to thwart the detection of credential expiration.

5.1.4 Modification in Place or in Transit

Many attacks rely on the ability to modify data in a persistent store or while it is in transit. A credential store or password file is an obvious target for an attack, as is a password on its way to an end user.

In business computing environments, it is far more common that persistent data sources are attacked rather than data in transit because the latter requires network-level penetration, a higher level of sophistication, and constitutes a MiM attack. Attacks against persistent stores have been categorized by others as “File Manipulation” attacks [15].

² See [14] for a taxonomy of variations of the replay attack.



The practice of modifying hidden tags in a Web form can be considered to be modification in transit and is not a MiM attack because the client is usually the attacker.

5.1.5 Interruption

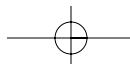
If a system's security depends on the completion of certain precursor processes, and those processes can be interrupted such that the system assumes that they completed, it might be possible to put the system into an insecure state by interrupting the precursor processes. For example, overwhelming a logging service such that it fails might be used to prevent a trace of intrusion activity from being recorded. This is often considered to be a denial of service, but often there is no service involved—merely an interconnected process—and so the term “interruption” seems more appropriate. Denial of service also can be caused by overwhelming an application, rather than by interrupting it. See the next attack description.

5.1.6 Saturation and Delay

In some systems, security relies on the existence of a service that will detect intrusion. In that case, all that is needed is to delay the response of the intrusion detection system long enough to allow an attack to complete or to force the service request to time out so that the requester uses cached data. This can often be accomplished by overwhelming the service or intrusion detection system. This is a type of attack that is commonly referred to as a denial of service, but the actual technique is a saturation technique; denial of service is the immediate effect [16] on the service or intrusion detection system, and there is then a security consequence as a result of the failure of the system to detect intrusion.

The intrusion detection system might not be specifically designed as an intrusion detection system per se, but might merely be, for example, a normal service that is designed to shut down if any anomalous behavior is detected; for example, if packets with the same sequence number are received.

Delay is a powerful technique because it takes time to identify an intruder, and if delay can be achieved, the attacker has time to cover their tracks and leave—and possibly enter through another means or mount an attack from a different compromised host location.



Besides their use as a means of penetrating a system, saturation and delay can be attack objectives in their own right. Saturation or delay perpetrated *for the purpose of making a system inaccessible or unusable* (i.e., making it “unavailable”) is properly known as a *denial of service* attack. However, note that denial of service can be achieved in other ways; for example, by interfering with any process that is critical to an application.

5.1.7 Exploitation of Non-Atomicity

If a software process or thread of execution accesses objects or resources that can be accessed by other processes or threads, or by other activities of the same thread, there is a possibility that logically concurrent access by more than one process, thread, or activity might interfere and make it possible to compromise the state of the system.♦

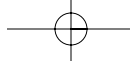
Some languages, such as Java, provide low-level primitives for controlling concurrent access to language objects. Databases provide locking mechanisms for serializing concurrent access to file-based data. A proper design for a concurrent system usually employs these mechanisms to ensure that interference does not occur. This often means designing software routines that access shared resources in such a way that the accesses are “atomic”—that is, that their effect is all-or-nothing, and intermediate states are unobservable.

If there is non-atomicity in the system, it is sometimes possible to force the system to perform steps out of sequence, thereby putting it into a state that was not anticipated by its designers. This is often referred to as a “race condition.”

A special case of this is when the interference is purposefully performed after a resource access rule is checked but before the resource is accessed: During that interval, a change is made to a critical context value, such as a user identity, causing the system to perform a function in a different context than the context that was authorized. This is referred to as a “time of check to time of use” (TOCTTOU) attack.

Attacks based on non-atomicity are often coupled with a denial of service attack that slows a system down and thereby “opens a window of vulnerability.”

♦ Non-atomicity is the *underlying cause* of a great many kinds of security vulnerabilities. This is a result of the very design of the von Neumann computing model that performs one computation at a time, instruction by instruction. As a result of this processing paradigm, any form of non-hardware-supported authorization check must be performed at discrete predetermined points in time instead of continuously, whereas security attacks can be launched at unexpected points in time.



5.1.8 Coordination Interference

Non-atomicity and delay pretty much cover attacks related to synchronization, but a related class of attacks deserves its own consideration for systems that are inherently asynchronous or independent and that depend on presumed event sequences, timing, or timestamps. Independent systems that cooperate are sometimes assumed to perform actions in a certain sequence or with certain effects, and interference with one of these systems can result in inconsistent effects that cause a failure in a different system.

Delay is often used to achieve this type of interference, for example, by interfering with or spoofing a timing service. Interference with a messaging service can result in certain events not being registered that are relied upon.

5.1.9 Forced Crash and Retrieval of Crash Artifacts

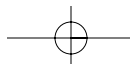
When systems fail, they often leave traces of their internal operation or leave resources in an inconsistent and potentially unprotected or insecure (for example, unencrypted) state. Access to protected information can, therefore, sometimes be achieved by forcing a system to crash and then examining the artifacts that remain.

A crash can sometimes be achieved by exploitation of incomplete validation of inputs or exposure of internal objects that can be modified during an attack to force the system into a state that causes failure.

The most common type of artifact left behind is a file containing an image of the process. Such an image often contains sensitive data, such as unencrypted credentials or the details and relative addresses of stack variables and program code.

5.1.10 Forced Restart, Forced Re-Install

One way of inserting malicious software into a system is by compromising a system's bootup or installation configuration. If the system is then caused to crash or become unusable so that it will have to be re-started, or corrupt so that it will have to be



re-installed (with a compromised installation), the compromised configuration will be started or installed, respectively.

This is an extremely powerful and subtle technique. It is unfortunately true that “backup” resources are usually much less protected than primary resources. Thus, by silently implanting a trojan horse (see Section 5.1.21, “Embedded Attack”) in a backup resource (or in an emergency response tool) and then merely forcing the primary resource to crash or be crippled, the compromised backup resource will be installed.³

This technique is closely related to the “Trusted Resource Attack” (discussed in Section 5.1.25).

5.1.11 Environmental Interference

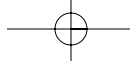
The normal operation of programs usually presumes the availability of resources, such as memory, threads, sockets, and file space. Software designers often do not anticipate the failure conditions that can occur when these resources are unavailable or are exhausted. This can result in certain functions not completing that are expected to complete. If the system’s security depends on these functions (for example, a system log) it might be possible to attack a system without traceability or without intrusion detection completing.

5.1.12 Spoofing

Spoofing involves forging or corrupting (destroying the *integrity* of) a resource or artifact for the purpose of pretending to be—i.e., for the purpose of *masquerading as*—something or someone else. There are many variations on spoofing, and it can be done at any level of a system, from the network level through the application level. Some examples are:

- Forging IP packet source addresses.
- Forging ARP packets to fool a router into thinking that your machine has someone else’s IP address.
- Creating misleading Web pages that fool a user into thinking that they are at a different site [17].

³ Tools exist that protect bootup configurations by using encryption.

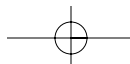


102 HIGH-ASSURANCE DESIGN

- Sending a name resolution request to a DNS server, forcing it to forward the request to a more authoritative server, and then immediately sending a forged response—causing the first DNS server to cache the forged response and supply that address to its clients. (Attacks that use this technique as a method of tricking users into accessing sites that mimick trusted sites, for the purpose of obtaining user credentials or other personal identity information, are often referred to as “pharming”.)
- Replacing a trusted file or program with a file or program that mimics the original one but that contains malicious data or code. This is also a kind of “trusted resource attack” (see Section 5.1.25).
- Forging or constructing an application object through abnormal means in order to instantiate an illegitimate instance that appears to be legitimate.

Spoofing of domain names may be made easier by the fact that names expressed in international character sets might be allowed. It used to be the case that a domain name had to consist of 7-bit ASCII characters. Unicode is now being considered for domain names. Because many Unicode glyphs have the same appearance, it is possible to have two domain names that look identical, but are actually composed of different Unicode characters. For example, the character glyph “a” represents the Unicode 16-bit hexadecimal value x0061 from the Basic Latin set, but it also represents the value x0430 from the Cyrillic set. This means that if you receive an email containing a link to “abc.com,” you can no longer be sure where the link might take you. This horribly regrettable situation will hopefully be remedied by new browser and email program security features that call attention to the use of links containing mixed or different character sets. A Web site can also help by enabling the browser to authenticate it using SSL, but that requires users to type “https” instead of “http.”

Spoofing often exploits an unsophisticated end user. For example, many Web users do not adequately understand or manage their browser security policies. Common ways of exploiting weakly secured browsers to spoof users include creating hidden windows from which attacks on other windows are launched, as well as manipulating the appearance and contents of the window to make it appear as if it were another kind of window, and modifying other windows that show legitimate content.



Spoofing is especially effective when coupled with delay or interruption because many spoofing schemes involve preventing a legitimate service from responding before an illegitimate one does. It is also powerful in combination with a forced restart or forced re-install or any kind of interference requiring an operator emergency response. This is because diagnostic tools or incident response tools can often be attacked more easily than the system itself. When users are in a crisis, they usually do not question the integrity of tools they invoke to help them respond to their crisis. Thus, the combination of attacking poorly-protected tools or configurations, followed by an attack that forces the system to fail and the tools to be used, is extremely powerful. This is in fact the very technique used by the thieves in the movie *Ocean's Eleven*: The thieves accomplish the removal of the bank's assets by carrying them away in the equipment bags of a spoofed SWAT team, having compromised the 911 channel and thereby enabling the spoof SWAT team to respond to the robbery. See also "Distraction and Diversity," discussed later in this chapter.

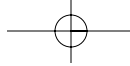
5.1.13 Hijacking

The term "hijacking" is usually used to refer to an attack that involves disconnecting a server resource in some manner from a resource channel and replacing it with a different server resource. Thus, the channel is "hijacked."

This is a variation of spoofing because users of the channel think that they are accessing the intended resource, via the channel, but are "spoofed" by the replacement resource.

5.1.14 Circumvention

This book shall define "circumvention" as any method by which an attacker bypasses intended controls, access checks, or system pathways in order to gain access to or control of protected resources. Circumvention can involve a covert channel or it can involve incompletely protected resources. Many of the attacks discussed here represent variations of circumvention.



5.1.15 Trap Door

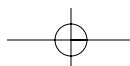
A *trap door* is a mechanism embedded within a system that allows the normal access paths or access checks of a system to be bypassed. This often takes the form of a special password that is hard-coded into the software. It can also take the form of a special diagnostic interface.

5.1.16 Exploit of Incomplete Traceability

If the *system's design* is such that it fails to record the actions of users, this can lead to a situation in which either appropriate or inappropriate actions are later untraceable or unprovable. It is important to emphasize that this is a result of the system's design—not a result of an attack directed against its logging mechanism. (An attack directed against the logging mechanism would most likely be a trusted resource attack, which is discussed later.)

The ability of a party to deny having performed *appropriate* actions or aspects of those actions (for example, the time at which they were performed) is known as *repudiation*. For example, a user might deny that she performed a particular transaction such as a purchase, and if there is no record that conclusively links her to the transaction, her denial might be successful. Another example would be denial that a message was received when in fact it was. The term *non-repudiation* refers to the ability of a system to defeat repudiation attempts, for example, by recording authenticated records (logs) of all transactions and by using communication mechanisms that provide secure acknowledgment at both endpoints.

Incomplete logs can also enable an intruder to perform *inappropriate* actions without traceability. For example, if an attacker's modification of sensitive files is not recorded in a manner that identifies the attacker's identity, the attack cannot be traced to its source. A failure to log actions in a traceable manner, therefore, represents a significant vulnerability.



5.1.17 Exploit of Incomplete Validation

If a software module does not fully check that its inputs fall within expected ranges, it might be possible to invoke the module with inputs outside of those ranges and thereby cause the program to do things that were not intended by the software designer. This might enable an attacker to circumvent normal system pathways or checks.

The infamous “buffer overflow” attack is a variation of incomplete validation, although in a buffer overflow the validation failure can be considered to be within the application framework (for example, language itself) rather than in the application design because a secure application framework should prevent buffer overflow as well as any other kind of type failure or range failure.

5.1.18 Exploit of Incomplete Authentication or Authorization

The design or configuration of a system might intentionally or unintentionally *omit* certain checks, enabling an attacker to “slip through” access control or authentication mechanisms and thereby obtain unauthorized access or control. This is most likely to be possible if authorization decisions are interspersed throughout the application code.

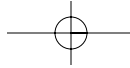
5.1.19 Exploit of Exposure of Internal State

Circumvention can also occur if resources expose their internal state, thereby allowing a client module to read or modify the resource’s internal state in unintended ways.

Inappropriate *reading* of a resource’s internal state is a breach of confidentiality because information that is intended to be private to the resource is revealed to an unintended party. This is known as a *containment failure*.♦

Inappropriate *writing* of internal state is a breach of integrity. For example, if a resource’s interface returns references (“aliases”) for internal objects instead of returning separate copies of those objects, any client of the resource might be able

♦ See Section 7.1, “Secure Resource Containers,” for more discussion of containment.



106 HIGH-ASSURANCE DESIGN

to modify the internal objects because they can obtain direct references to them. This kind of failure has been categorized by some as an “integrity failure” resulting from an “aliasing error.” [18]

This form of attack is the motivation behind the security model embedded in many browsers. In this model, often referred to as the “same origin” policy, Web pages can only affect their own contents. However, there are loopholes in the policy. For example, scripts can embed executable objects that do not adhere to the security policy, but rather adhere to a different (possibly looser) security policy.

5.1.20 Exploit of Residual Artifacts

If objects are re-used and contain artifacts of prior use, an attacker might be able to use those artifacts to obtain secret information or to obtain references to protected objects. This is analogous to an intruder searching through your trash can, a practice sometimes referred to as “dumpster-diving” or “trashing.” [19]

5.1.21 Embedded Attack

This book uses the term “embedded attack” to refer to all attacks that rely on the placement of attack software within a trusted software system. The act of setting up an embedded attack is commonly referred to as *planting*, [20] because a subversive component is “planted” on the target system. Planting can be achieved using other techniques, such as social engineering (discussed in Section 5.2, “Social Engineering”) or technical means.

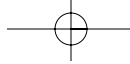
A very common form of embedded attack is known as a “trojan horse” attack. A *trojan horse* is a trusted component that is imported or installed (somehow) into the system but which contains a secret mechanism to facilitate a subsequent attack. Generally the user has rights that the program’s author (i.e., the attacker) does not, so the attacker obtains the user’s rights by “hiding” inside a trusted component. This implies that the attacker has access to the trusted component, has convinced the user that the component can be trusted, or has lured the user into installing or enabling it.

A trojan horse program can be installed (planted) as a result of a computer virus. An example of this delivery method is the “Troj/BankAsh” virus (2005), which attempts to disable anti-virus software and then monitors the user’s Internet access for banking Web sites, such as Barclays, Cahoot, Halifax, and others. If a banking Web site is accessed, the program silently monitors the user’s keystrokes in order to capture a login ID and password and other account information and then FTPs this information to a remote site.

So-called “script injection” attacks are a special case of a trojan horse attack in which a script (i.e., a program) is input in lieu of data and is then later inadvertently interpreted (executed) by the application. A trojan horse can also be used to execute a MiM attack by intercepting internal information “from the inside” and using it maliciously.

An embedded attack is sometimes implemented as a “bomb.” A *time bomb* is a subversive mechanism secretly embedded within a trusted system for the purpose of initiating an attack at a later point in time. A *logic bomb* is similar to a time bomb except that it is triggered by a sequence of program events rather than by the passage of time.

Embedded attacks are especially effective when coupled with a forced crash. An example is the compromise of a repair tool or boot script followed by causing the system to fail so that it will have to be repaired or rebooted using the compromised tool or script. This is particularly effective because “build-time” components, such as tools and scripts, are often less stringently protected than runtime systems.



5.1.22 Pattern Matching and Brute Force Guessing

Attacks that utilize sophisticated knowledge to derive or anticipate the state of a system or credentials used for authentication are often referred to as “oracle” attacks. These include deciphering, discovery of exploitable patterns, non-randomness or predictable pseudo-randomness, and exploitation of algorithmic weaknesses. A notorious example is the cracking of Netscape’s implementation of SSL by taking advantage of a weakness in its random number generation.

Attacks that merely try every possibility until they succeed are known as “brute force” or “exhaustive search” attacks. Encryption algorithms that are not sufficiently strong or that use relatively short keys can often be cracked using brute force: This is a result of the ever-decreasing cost of computing power.

5.1.23 Namespace Attack

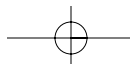
Many attacks exploit weaknesses in the name resolution process used to identify resources. These include the insertion of rogue components in a name-resolution path as well as the insertion of components with similar names that are equivalent. It is often the case that abbreviated names are used to identify resources, and a failure to canonicalize a resource name can enable an attacker to substitute other resources with the same abbreviated name but a different canonical name.

5.1.24 Weak Link as Gateway

Attackers often reach their goal by following a circuitous path: entering a weak point and then using that point as a point of trust from which to reach other points.⁴

Human resources Web sites are famous examples of this. Those sites are often poorly protected, but because they have the same domain as other organization sites, they can be used as a launching point when compromised.

⁴ See the discussion of “transitive confinement” in [21], page 441.



Virtual private networks (VPNs) represent another consideration. For example, if a network is linked to a partner via a VPN and the partner's network has a known weakness, the partner's network can first be penetrated and then used as a gateway to the target network.

5.1.25 Trusted Resource Attack

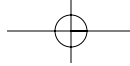
An application can often be penetrated by attacking a resource on which the application relies. Examples of this include:

- Attacking a DNS server's zone files.
- Attacking an object lookup service, for example, by covertly embedding a trojan horse within its code.
- Modifying the system time. (Some taxonomies identify this kind of attack as a category in its own right.)
- Modifying files that are used by an application but that have insufficient protection.
- Attacking log files that are used to record the actions of users.
- Attacking *other* programs that are poorly protected and that access (and ideally modify) the same resources on which the application of interest relies.⁵ Thus, this approach is *transitive* in that it involves attacking a trusted resource, in order to attack another target that uses the trusted resource. This particular pattern is an example of a "Weak Link as Gateway" attack.

This is how source code to Cisco Systems routers was stolen in 2004—by planting a compromised version of the trusted SSH program on Cisco's network to act as a trojan horse by sending users' passwords to the attacker. [23]

An effective way to attack a protected resource is to subvert resources used by those resources—with many levels of transitivity in between.

⁵ See [22] for a discussion of how users can be lured into installing tools such as compilers that contain malicious code.



110 HIGH-ASSURANCE DESIGN

This technique has easy parallels in the non-computer world. There was a movie in the 1960s called *Kaleidoscope* in which a professional card player stealthily entered the factory of a playing card manufacturer and modified the very dies used to print a popular brand of playing cards. He alone knew of the tiny modifications and was able to play poker and win. This is an example of a two-level transitive attack: He attacked a resource (the factory) used to produce the resources used by the casinos (the cards). Thus, an effective way to attack a protected resource is to subvert resources used by those resources. This includes emergency response resources. (See the “Forced Restart, Forced Re-Install” attack.)

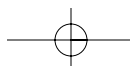
5.1.26 Scope Attack

Weaknesses in scoping mechanisms, such as a runtime container, a language-defined scoping boundary, or any other security policy domain or mechanism designed to keep objects from communicating, can be used to access objects or resources outside of their intended scope or boundary. Weaknesses that make this kind of attack possible have been described as “domain errors” [24].

A security domain or access scope can sometimes be circumvented by employing a circumvention technique such as a covert channel (discussed previously). Another kind of scoping attack is to exploit the inadvertent exposure of a resource’s internal structure (see the “Exploit of Exposure of Internal State” attack in Section 5.1.19).

5.2 Social Engineering

Malicious actions performed against *people*, including deception or the physical theft of sensitive information such as credentials, represent highly effective avenues of attack against computers. In these cases, the errors or failures are human, and for this reason approaches that utilize human failings are known as *social engineering* attacks.



“I became absorbed in everything about telephones—not only the electronics, switches, and computers, but also the corporate organization, the procedures, and the terminology. After a while, I probably knew more about the phone system than any single employee. And I had developed my social engineering skills to the point that, at seventeen years old, I was able to talk most Telco employees into almost anything, whether I was speaking with them in person or by telephone.”

—Kevin Mitnick

Deception perpetrated for the purpose of theft or subversion, known as *fraud*, usually involves misplaced confidence in the perpetrator, and for this reason fraud schemes are often referred to as confidence schemes, or “con games.” More direct attacks against humans involving the theft of information through surveillance or the actual theft of information or assets of value are also common, such as the theft of volumes of customer information.

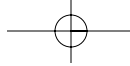
Social engineering attacks are as old as civilization. What is new is their application to obtain computer access. Most social engineering computer attacks utilize one or more of the techniques discussed next. For those who are interested, comparable “con schemes” used by non-computer crooks are listed in the sidebars.

5.2.1 Physical Theft

Physical theft of computer assets or documents that contain computer-related information can be used to compromise a system. An example is the theft of a computer or storage media containing information of value. Another example is the theft (or reclamation) of discarded assets from which computer-related information can be retrieved—so-called “dumpster-diving” (see “Exploit of Residual Artifacts”).

SOCIAL ENGINEERING METHODS, BY SECTION

Physical Theft	5.2.1
Emotional Pressure	5.2.2
Haste	5.2.3
Reliance on Inadequate Protection	5.2.4
Instilling Undeserved Trust	5.2.5
Breaking Prior Trust	5.2.6
Trusted Resource Attack	5.2.7



5.2.2 Emotional Pressure

Most social engineering techniques utilize some form of emotional pressure in order to induce the target to accept something that is unvalidated or to make a decision based on unvalidated information. This is usually done in conjunction with a technique to instill trust in the perpetrator. The emotional pressure usually takes one of these forms:

1. **Emotional investment:** Example: Claim to need help with something, but then in the middle of the effort, request help with something else that would require authorization—and explain that it would be very helpful to bypass procedure in order to expedite things.
2. **Intimidation:** Example: Telling an administrator that if they do not expedite a request by circumventing a security procedure, the company's profit might be at risk, and hence their job.
3. **Pride:** Example: Challenging an administrator to do something that requires broad-ranging privileges.
4. **Inconvenience** (for example, inconvenience of checking something out): Example: Presenting proof of validity that is unusual and appears legitimate but that would take time and effort to verify.
5. **Enticement** (perhaps sufficient to induce you to avoid safeguards): Example: Promising an insider that they will participate in a lucrative fraud scheme, when in fact the only objective is to get them to perform an act of subversion, such as allowing use of their computer account.
6. **Crisis:** Create a crisis that needs a solution and offer the solution (target accepts inadequate validation from the responder). Example: Send advertisement for a service professing to identify hackers and then slightly hack the target's computer; if contacted, induce the target to provide full and direct access to their system.

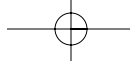
TRADITIONAL EXAMPLES OF ENTICEMENT SCHEMES

Scheme Name	Method
Nigerian Letter	Letter recipient is enticed into participating in a lucrative scheme—for a fee that is later revealed.
Pigeon Drop	Target is enticed into sharing in a valuable treasure find, provided the target puts up “good faith money.” The treasure is either worthless or taken away.
Pump-and-Dump	False rumors are generated to cause a stock price to inflate. This is an enticement scheme because the rumors are not substantiated.
Spanish Prisoner	Purported ex-prisoner entices target into sharing a stolen fortune, as long as target provides funds to get to it.
Rocks-in-the-Box	Target is enticed into buying purportedly valuable but shady merchandise for cash, only to find that the merchandise is worthless.
Country Boy	Target is enticed into defrauding an apparently naïve person, provided that the target provides “good faith money.”
Contest Winner	Target is enticed into sending money to claim a supposedly large prize.
3-Card Monte	Target is enticed into playing a rigged card game based on a controlled demonstration that it is easy to win.
Truck load scam	Target is enticed into buying shady merchandise and pays up front only to find the truck has left.
Bankruptcy fraud	Naïve creditors are enticed into accepting an early unconditional settlement of a debt when told that a bankruptcy is imminent for the debtor, but the bankruptcy application is then rescinded.

5.2.3 Haste

Most schemes to deceive involve a sense of haste or urgency. The perpetrator requires haste in order to prevent detection, because given time, credentials can be validated, and a story double-checked. Also, given time, intrusion detection processes and notification processes complete.

The most common way to induce haste in a victim is to create a form of pressure that is time-based; the time aspect may be real (verifiable) or artificial (would fail verification, but the time pressure makes verification inconvenient or seemingly risky). For example, the perpetrator might ask that things be done quickly in order to make some form of deadline.



114 HIGH-ASSURANCE DESIGN

TRADITIONAL EXAMPLES OF CRISIS-BASED PRESSURE

Scheme Name	Method
Bail bond scheme	Target is contacted and urged to provide bail money for a relative who is known to be presently inaccessible. The target is under pressure to provide the money immediately without confirming that their relative is indeed in jail.
Phony COD delivery	Target is approached at home by a legitimate-looking delivery person asking for a COD fee. In the pressure of the moment, the target pays the fee.

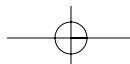
5.2.4 Reliance on Inadequate Protection

Software applications protect assets of value: information and transactions. If the protection mechanisms have loopholes or are inadequate, an attacker merely needs to discover those sources of inadequacy.

Inadequate protection can result from merely not having enough resources to respond to an emergency of large magnitude or to multiple emergencies concurrently. Many of the attacks in this category rely on overwhelming a response system. The techniques include:

1. **Diversion:** Cause an *unexpected* crisis that *overwhelms response resources*. Example: Attack a system that is not of interest but that is easy to attack so that security administrators are focusing their attention there; then attack the actual target. Another example is to cause a different kind of crisis, such as a fire.
2. **Decoys:** Overwhelm response resources, by creating a storm of false events that prevent the responders from identifying the real events.⁶ An example of the use of decoys would be a storm of packets from many compromised “decoy” systems preventing security administrators from tracking the true source of the attack.

⁶ An interesting example of the use of decoys was in the movie *The Thomas Crown Affair* (1999), in which the thief was expected to appear at a certain time and place where police were waiting, but when he appeared he had also arranged for a great many other men to appear, dressed in the same way and in such a manner that these men could not be distinguished from the thief himself—overwhelming the police to check each man and allowing the thief to escape.



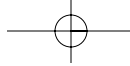
3. **Distraction:** Create innocent confusion that disrupts thought processes or safeguards. Example: Ask an administrator for help that requires that he access sensitive information in your presence and have someone else interrupt him several times.
4. **Reliance on naiveté:** Identify a new and inexperienced administrator.
5. **Reliance on ineffectiveness:** Rely on incompetence, lack of diligence, or an inability to contain or identify an attacker. Example: Falsifying fulfillment transactions that are less well protected. All technical security weaknesses also fall into this category because they represent an ineffective system.
6. **Attack the responders (or compromise them):**
Example: Implant a trojan horse in a recovery image.

TRADITIONAL EXAMPLES OF RELIANCE ON INEFFECTIVENESS

Scheme Name	Method
False Claims	Reliance on the inability of a claim processing system to validate claims of expenses.
Kiting	Reliance on the inability of a clearing operation to reconcile related transactions in real time.
Shorting	Delivering less than promised and relying on the inability of the recipient to verify the amount or quality.

Distraction and diversion are classic human behavior techniques that rely on the limited ability of people to deal with multiple situations at the same time as well as the inability of people to think through unusual situations in a timely manner. For example, if an intrusion detection system is repeatedly triggered in some manner that is identified as a “false alarm,” the response staff might temporarily disable it until it can be examined. This opens a window of vulnerability.

Distraction can be caused by any abnormal situation that diverts the attention of staff. Diversion involves deliberately triggering an alarm so that response staff will then not notice *other alarms* because they are responding to the first.



116 HIGH-ASSURANCE DESIGN

Attacking emergency responders is a very powerful technique because when they are not “on-guard” they are usually poorly protected compared with the assets that they protect.

5.2.5 Instilling Undeserved Trust

In order to convince someone to perform an action that you request, you must get them to trust you. Therefore, most social engineering schemes include a method of instilling trust and then exploiting that trust such as inducing the target to perform an unprotected action (especially when you are desperate and your guard is down).

The common methods of instilling undeserved trust are:

1. **Stolen credentials (or any validation information):**
Example: “Shoulder surfing” to obtain someone’s logon, and then using it. Another example: Enrollment through fraudulent representation. Another example: Using the customer service pathway; for example, calling to change your address and then having sensitive information mailed to you.
2. **Counterfeit credentials:** By presenting counterfeit credentials or evidence of legitimacy. Example: Posing as a system administrator and demonstrating knowledge to buttress the claim, and asking a user to provide sensitive information. This is a variation of the so-called “bank examiner” fraud in which a perpetrator poses as a bank official and tells a customer to turn over funds from their account under false pretenses. Another example: Similar domain names. Another example: Imitation PayPal link schemes, or clicking on any legitimate-looking link in an email from an unvalidated source.
3. **Juxtaposition with something legitimate:** By association or juxtaposition with something legitimate.
Example: Links to the Web page of legitimate services.

4. **Successful interactions:** Through legitimate or seemingly legitimate interactions. Example: Free services that build trust and eventually solicit sensitive information.

TRADITIONAL EXAMPLES OF INSTILLING UNDESERVED TRUST THROUGH SUCCESSFUL INTERACTIONS

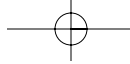
Scheme Name	Method
Ponzi scheme	Targets invest more and more, based on what they perceive to be good experiences. Their money is eventually taken.
Big store	Target witnesses lots of apparently successful transactions, and uses that as a basis for trust.
Salting the gold mine	Target trusts an investment based on the initial discovery of valuable resources or gains that were actually planted.
Sweetheart swindle	Trust is developed through courtship.

5. **Peer collusion:** Through interactions with a third party that is secretly collusive. Example: Relationships with other seemingly legitimate companies, that are actually “fronts.”
6. **Interactions with trustworthy entities:** Through interactions with an entity trusted by the target. Example: Evidence that services were obtained from a trusted security company.

5.2.6 Breaking Prior Trust

Rather than instill trust, it is often possible for a perpetrator to persuade a party that is already trusted to act in collusion with them. This is usually achieved through some form of incentive, such as bribery or mutual gain. The forms of collusion include:

1. **Collusion with an insider:** Often a result of a conflict of interest, or when roles are not sufficiently separated. Examples: An administrator altering a transaction log; a back door; inserted by a trusted programmer causing a financially favorable error in the account of an employee who has high authority to see if they report the error; transactions recorded as discounted when they are not.



118 HIGH-ASSURANCE DESIGN

2. Collusion with outsiders: Example: A collusive security monitoring service.

Indeed the members of the notorious DrinkorDie Web piracy group *often relied on moles in large corporations* and cracked security codes for Norton Antivirus, Microsoft's Word and Excel products, pirated games and design programs, and posted the entire Windows 95 operating system on the Internet two weeks before it was released. [25]

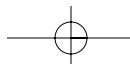
TRADITIONAL EXAMPLES OF INSIDER COLLUSION

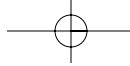
Scheme Name	Method
Embezzlement, or "Cooking the Books"	An employee modifies accounting records in order to conceal unauthorized transactions.
Kickback	An insider grants a contract based on the expectation that the contractor will secretly provide a gift.
Salami	An unnoticeable portion ("slice") of proceeds is taken on an ongoing basis.
Under-Ring	A transaction is entered for less than the actual amount charged, and the difference is pocketed.
Employee Account Fraud	The insider has both a work relationship and a business relationship with the organization, and uses employee access to business records to modify the account.
Fictitious Refunds, Fictitious Sales, Negative Invoicing	False refunds, sales, and invoices are submitted by an outsider and facilitated by the insider.
"Ghost" employees	A managerial employee budgets for staff who do not exist, and pockets their payroll checks.
"Salting Cash"	Insiders who would be willing to compromise their organization are identified by causing errors in their favor and observing if they report the error.

5.2.7 Trusted Resource Attack

This form of attack has already been discussed in the context of technical attacks. It is included here because it can be used to attack non-computing assets, and also because of its tremendous power and importance.

Example: Compromise the tools used by emergency response personnel.





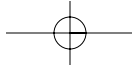
5.3 Summary

The methods of attack that are available are broad-ranging and insidious, yet many of them are available to even amateur hackers through the use of tools widely available on the Internet. For this reason, securing applications today is no small challenge. The principles presented in the chapters that follow help to guide the design of applications that are resistant to attack.

This chapter discussed the various kinds of attack, including categories and examples of social engineering attacks. The next chapter presents concepts for trust, which is an important foundation for understanding Chapter 7, in which authorization and containment are discussed at length.

5.4 Exercises

1. Which type of data is attacked more frequently: persistent data, or data in transit?
2. An attack that involves disconnecting a server resource in some manner from a resource channel and replacing it with a different server resource is known as what kind of attack?
3. What type of subversive action must be completed before a trojan horse attack can be performed?
4. Explain what non-repudiation is.
5. What type of system deficiency does a buffer overflow attack exploit?
6. What type of organizational deficiency does a diversion rely on?
7. Varying CPU load to secretly signal internal sensitive information to an external listening process is an example of what kind of mechanism?



120 HIGH-ASSURANCE DESIGN

8. What type of system deficiency does a race condition exploit?
9. Inserting a rogue reference into a path variable exploits what kind of system weakness?

Answers and hints for these exercises can be found at <http://www.assuredbydesign.com/haa/>.

