Introduction to Computer Security

Threats

- A threat to a computer system is any potential occurrence, malicious or otherwise, that can have an undesirable effect on the assets and resources associated with a computer system.
- The concept of a threat is significant because the generally accepted goal of computer security is to provide insights, techniques and methodologies that can be used to mitigate threats.

Vulnerabilities

- A vulnerability of a computer system is some unfortunate characteristic that makes it possible for a threat to potentially occur. In other words, the presence of vulnerabilities allows bad things to happen on a computer system.

Attacks

- An attack on a computer system is some action taken by a malicious intruder that involves the exploitation of certain vulnerabilites in order to cause an existing threat to occur
- Attacks are often heurisitic, involving some knowledge about vulnerabilities on the part of the attacker.

Disclosure Threat

- The disclosure threat involves the dissemination of information to an individual for whom that information should not be seen.
- The disclosure threat occurs whenever some secret that is stored on a computer system or in transit between computer systems is compromised to someone who should not know the secret. Sometimes the term "leak" is used in conjunction with the disclosure threat.
- Examples of such compromise can include minor embarrassments, such as when personal information is disclosed to a colleague.

Integrity Threat

- The integrity threat involves any unauthorized change to information stored on a computer system or in transit between computer systems.
- When intruders maliciously alter information, we say that the integrity of this information has been compromised.
- We also say that integrity has been compromised if an innocent mistake results in an unauthorized change. Authorized changes are those that are made by certain individuals for justifiable purposes.
- As with the disclosure threat, the integrity threat can also involve minor consequences when non-critical information is changed.

Denial of Service Threat

- The denial of service threat arises whenever access to some computer system resource is intentionally blocked as a result of mailicious action taken by another user. That is, if one user requires access to a service and another user does something malicious to prevent such access, we say that a denial of service has occurred.
- The actual blocking may be permanent so that the desired resource is never provided, or it may just cause the desired resource to be delayed long enough for it to no longer be useful. In such cases, the resource is said to have become stale.
- The most common examples of denial of service attacks involve users hogging shared resources such as printers or processors so that other users cannot use them. As long as these shared resources are not part of some critical mission, this type of attack may be benign.

Attacker Intent

- An issue that greatly complicates the prevention of threats is that the basic intent associated with an attack often cannot be determined. That is, even if a given attack can be observed, the motivation of the individual who caused some problem to be in introduced or exploited often cannot be identified.
- Cases certainly exist in which an individual confesses to performing some malicious action, but these are not common.

An Example

- To illustrate the problem, we can consider a somewhat whimsical software development scenario. Suppose that you are a member of a software development team working on the design and development of some critical software system and you happen to notice a programming error that one of your colleagues has made. To make matters more concrete, suppose that you are programming in FORTRAN and error you notice is the following statement:

```
DO 20 I = 1. 100
```

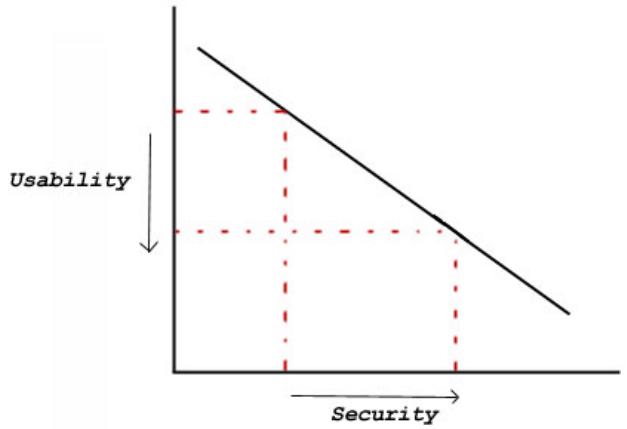
- Upon examination of the code surrounding this statement, you observe that a FORTRAN DO statement is an appropriate statement in the context of the application being coded. However, suppose that you know that the correct syntax for such a statement is as follows:

```
DO 20 I = 1, 100
```

- The correct statement uses a comma rather than a period to separate the two numeric values. The incorrect statement is potentially harmful because due to a quirk in the FORTRAN language, the value 1.100 will be assigned to the implicitly declared variable DO20I. This is because the FORTRAN language ignores spaces between characters and allows implicit variable declarations.

Security and Usability

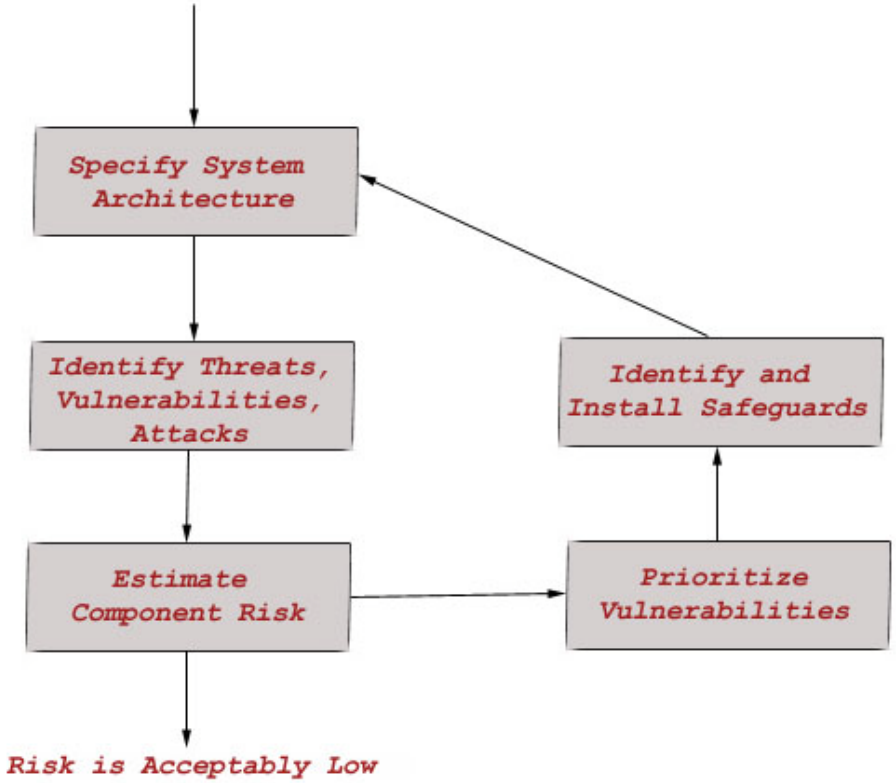
- A conflict generally occurs when the goal of information and resource sharing is combined with the goal of strict security controls between users. Certainly, the conflict can be resolved and systems do exist that deal acceptably with both issues.



Further Impediments to Security

- Retrofit:
  - Since security is a relatively recent concern, nearly all existing systems were developed without sufficient attention to threats, vulnerabilities and potential attacks. In order to make an existing system secure, one is faced with the problem of retrofitting security into existing components, mechanisms, and environments. Consider, for example, the problem of retrofitting security mechanisms into an existing operating system. If the proposed mechanisms change an established system call interface to the operating system kernel routines, then existing user level applications may no longer work.
- Assurance
  - Even if a designer or developer contends that a given system includes mechanisms that enforce suitable policies toward the mitigation of threats, users of that system will only accept this contention if convincing evidence is made available. The body of evidence that a system is secure is referred to as assurance evidence. Unfortunately, the only types of assurance evidence that are available include test results, field results and use of formal methods.
- Procedures vs. Mechanisms
  - The mitigation of threats on computer systems requires the integration of suitable procedures or mechanisms. These procedures and mechanisms may range from management policies on personnel, facilities, and operations to functional mechanisms designed into a computer system. Determination of which type of mitigation approach is most suitable is not always obvious.
- Security Requirements
  - From a system software engineering perspective, one might wonder why security cannot be provided by simply identifying a set of suitable security requirements and then building a system that meets these requirements. This is not so easy for at least two reasons. First, identifying security requirements is difficult for nontrivial systems. Second, even if security requirements have been identified, the state of the practice in developing systems that meet requirements remains less than optimal.

System Security Engineering



Security

- Integrity: information has not been altered in any way
- Privacy/secrecy: information has not been intercepted and read by anyone
- Authentication: information has come from an authorized sender
- Nonrepudiation: information has proof that the started sender initiated the transaction

Intruders and Viruses

- Software trespass can take the form of a virus, worm, or Trojan horse.
- A virus or Trojan horse may be introduced into a system by means of a diskette.
- Only the worm is a uniquely network phenomenon. System trespass is an area in which the concerns of network security and computer security overlap.

Intruders

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

Intrusion Techniques

- Typically, a system must maintain a file that associates a password with each authorized user. The password file can be protected in one of two ways:
- One-way encryption: the system stores only an encrypted form of the user's password. When the user presents a password, the system encrypts that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed-length output is produced.
- Access Control: Access to the password file is limited to one or a very few accounts.

Techniques for learning passwords

- Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
- Exhaustively try all short passwords (those of one to three characters).
- Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
- Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
- Try users' phone numbers, Social Security numbers, and room numbers
- Try all legitimate license plate numbers for this state.
- Use a Trojan horse to bypass restrictions on access.
- Tap the line between a remote user and the host system.

Password Protection

- All multiuser systems requre that a use provide not only a name or identifier (ID) but also a password. The ID provides security in the following ways:
  - The ID determines whether the user is authorized to gain access to a system.
  - The ID determines the privileges accorded to the user.
  - The ID is used in what is referred to as discretionary access control.

The Vulnerability of Passwords

- To understand the nature of the attack, let us consider a scheme that is widely used on UNIX systems, in which passwords are never stored in the clear. Rather, the following procedure is employed:
- Each user selects a password of up to eight printable characters in length. This is converted to a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. The encryption routine, known as a crypt(3), is based on DES.
- The DES algorithm is modified using a 12-bit "salt" value. Typically, this value is related to the time at which the password is assigned to the user. The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros. The output of the algorithm then serves as input for a second encryption.
- The process is repeated for a total of 25 encryptions. The resulting 64-bit output is then translated into a 11-character sequence. The ciphertext is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.

The Vulnerability of Passwords Cont'd

- The salt serves 3 purposes:
  - It prevents duplicated passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.
  - It effectively increases the length of the password without requiring the user to remember two additional characters. Hence, the number of possible passwords is increased by a factor of 4096, increasing the difficulty of guessing a password.
  - It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.

The Vulnerability of Passwords Cont'd

- When a user attempts to log on a UNIX system, the user provides an ID and a password. The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password, which are used as input to the encryption routine. If the result matches the stored value, the password is accepted.
- There are two threats to the UNIX password scheme.
  - Gain access on a machine using a guest account
  - Gain access by some other means and then run a password guessing program, called a password cracker, on that machine.
- If an opponent is able to obtain a copy of the password file, then a cracker program can be run on another machine at leisure. Password crackers rely on the fact that some people choose easily guessable passwords

The Vulnerability of Passwords Cont'd

- Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. The cracker simplly has to test the password file against lists of likely passwords. In all, nearly one-fourth of the passwords were guessed. The following strategy was used:
  - Try the user's name, initials, account name, and other relevatnt personal information. In all, 130 different permutations for each user were tried.
  - Try words from various dictionaries. The author compiled a dictionary of over 60,000 words, including the on-line dictionary on the system itself, and various other lists as shown.
  - Try various permuations on the words from step 2. This included making the first letter uppercase or a control character, making the entire word uppercase, reversing the word, changing the letter "o" to the digit "zero", and so on. These permutations added another 1 million words to the list.
  - Try various capitalization permutations on the words from step 2 that were not considered in step 3. This added almost two million additional words to the list.

Access Control

- Password Selection Strategies
- To begin, we explain the operation of the Bloom filter. A Bloom filter of order k consists of a set of k independent hash functions H1(x), H2(x), ..., Hk(x), where each function maps a password into a hash value in the range 0 - N - 1. That is,

  Hi(Xj) = y 1<= i <= k; 1 <=j <= D; 0 <= y <= N - 1

  where
  - Xj = jth word in password dictionary
  - D = number of words in password dictionary

Access Control Cont'd

- The following procedure is then applied to the dictionary:
  - A hash table of N bits is defined, with all bits initially set to 0.
  - For each password, its k hash values are calculated, and the corresponding bits in the hash table are set to 1. Thus, if $Hi(X_j) = 67$ for some (i,j), then the sixty-seventh bit of the hash table is set to 1; if the bit already has the value 1, it remains at 1.
  - When a new password is presented to the checker, its k hash values are calculated. If all the corresponding bits of the hash table are equal to 1, then the pssword is rejected. All passwords in the dictionary will be rejected. But there will also be some "false positives" (that is, passwords that are not in the dictionary but that produce a match in the hash table).
  - To see this, consider a scheme with two hash functions. Suppose that the passwords *undertaker* and *hulkhogan* are in the dictionary, but *xG%#jj98* is not. Further, suppose that
    - H1 (undertaker) = 25 H1(hulkhogan) = 83 H1(xG%#jj98) = 665
    - H2 (undertaker) = 98 H2(hulkhogan) = 665 H2(xG%#jj98) = 998

Access Control Cont'd

- If the password *xG%#jj98* is presented to the system, it will be rejected even though it is not in the dictionary. if there are too many such false positives, it will be difficult for users to select passwords. Therefore, we would like to design the hash scheme to minimize false positives. It can be shown that the probability of a false positive can be approximated by :
  - P = (1 = e -kD/N)k = (1 - e-k/R)k
- or equivalently:
  - R = -k/(ln(1 - P1/k))

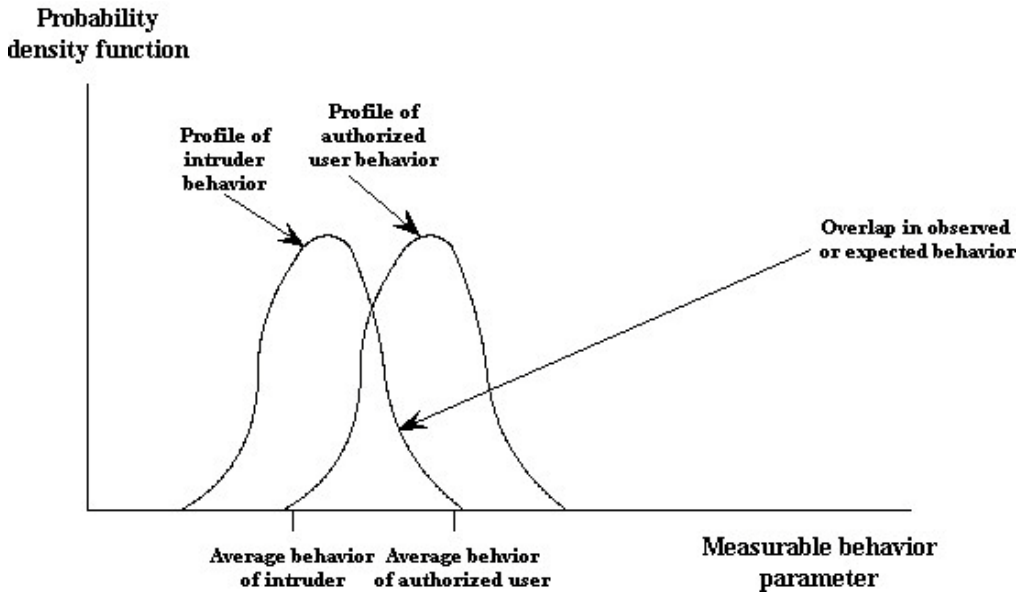
  where
  - k = number of hash functions
  - N = number of bits in hash table
  - D = number of words in dictionary
  - R = N/D, ratio of hash table size (bits) to dictionary size (words)

Intrusion Detection

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
- An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion Detection Cont'd

- Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified. The following figure suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Patterns of legitimate user behavior can be established by observing past history, and significant deviation from such patterns can be detected.



Statistical Anomaly Detection

- Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.
  - a. Audit Records: Threshold detection: This approach involves defining thresholds, independent of the user, for the frequency of occurrence of various events
  - b. Profile based: A profile of activity of each user is developed and used to detect changes in the behavior of individual accounts.

Rule-based Detection:

- Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
  - a. Anomaly detection: Rules are developed to detect deviation from previous usage patterns.
  - b. Penetration identification: An expert system approach that searches for suspicious behavior.

Audit Records

- A fundamental tool for intrusion detection is the audit record. Record of ongoing activity must be maintained by users as input to an intrusion detection system.
  - Native audit records: Virtually all multiuser operating systems include account software that collects information on user activity.
  - Detection-specific audit records: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system.

Statistical Anomaly Detection

- Metrics that are useful for profile-based intrusion detection are the following:
  - Counter: A nonnegative integer that may be incremented but not decremented unitil it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.
  - Gauge: A nonnegative integer that may be incremented or decremented. Typically, a guage is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.
  - Interval timer: The length of time between two related events. An example is the length of time between successive logins to an account.
  - Resource utilization: Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution.
- Given these general metrics, various tests can be performed to determine whther current activity fits within acceptable limits.

Distribution Intrusion Detection

- Although it is possible to mount a defense by using stand-alone intrusion detection systems on each host, a more effective defense can be achieved by coordination and cooperation among intrusion detection systems across the network.

Viruses And Related Threats

- Sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems. We are concerned with application programs as well as utility programs, such as editors and compilers.

Malicious Programs

- These threats can be divided into two categories: those that need a host program, and those that are independent. The former are essentially fragments of programs that cannot exist independently of some actual application program, utility or system program. The latter are self-contained programs that can be scheduled and run by the operating system.

- These threats can be divided into two categories: those that need a host program, and those that are independent. The former are essentially fragments of programs that cannot exist independently of some actual application program, utility or system program. The latter are self-contained programs that can be scheduled and run by the operating system.

Malicious Programs Cont'd

- ***Trap Doors***
  - A trap door is a secret entry point into a program that allows someone that is aware of the trap door to gain access without going through the usual security access procedures.
- ***Logic Bomb***
  - The logic bomb is code embedded in some legitimate program that is set to "explode" when certain conditions are met.
- ***Trojan Horse***
  - A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.
- ***Viruses***
  - A virus is a program that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.
- ***Worms***
  - Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.
- ***Bacteria***
  - Bacteria are programs that do not explicitly damage any files. Their sole purpose is to replicate themselves. A typical bacteria program may do nothing more than execute two copies of itself simultaneously on a multiprogramming system, or perhaps create two new files, each of which is a copy of the originial source file of the bacteria program.

Malicious Programs Cont'd