**Chapter 1**

**User Authentication**

This chapter examines some of the authentication functions that have been developed to support network-based use authentication. In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. User authentication is the basis for most types of access control and for user accountability. RFC 2828 defines user authentication as the process of verifying an identity claimed by or for a system entity. An authentication process consists of two steps:

* **Identification step:** Presenting an identifier to the security system. (Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.)
* **Verification step:** Presenting or generating authentication information that corroborates the binding between the entity and the identifier.”

In essence, identification is the means by which a user provides a claimed identity to the system; user authentication is the means of establishing the validity of the claim. Note that user authentication is distinct from message authentication.

**Means of User Authentication**

There are four general means of authenticating a user's identity, which can be used alone or in combination:

• **Something the individual knows:** Examples includes a password, a personal identification number (PIN), or answers to a prearranged set of questions.

• **Something the individual possesses:** Examples include electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a *token*.

• **Something the individual is (static biometrics):** Examples include recognition by fingerprint, retina, and face.

• **Something the individual does (dynamic biometrics):** Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

All of these methods, properly implemented and used, can provide secure user authentication. However, each method has problems. An adversary may be able to guess or steal a password. Similarly, an adversary may be able to forge or steal a token. A user may forget a password or lose a token. Further, there is a significant administrative overhead for managing password and token information on systems and securing such information on systems. With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.

**Authentication Protocols**

An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys. There, the focus was key distribution. Central to the problem of authenticated key exchange are two issues: confidentiality and timeliness. To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose. The second issue, timeliness, is important because of the threat of message replays.

**Replay Attacks**

Replay Attacks are where a valid signed message is copied and later resent. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not. Possible countermeasures include the use of:

• sequence numbers (generally impractical since must remember last number used with every communicating party)

• timestamps (needs synchronized clocks amongst all parties involved, which can be problematic)

• challenge/response (using unique, random, unpredictable nonce, but not suitable for connectionless applications because of handshake overhead)

**One-Way Authentication**

One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead, the e-mail message is forwarded to the receiver’s electronic mailbox, where it is buffered until the receiver is available to read it. The "envelope" or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP) .

However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail- handling mechanism. Accordingly, the e-mail message should be encrypted such that the mail- handling system is not in possession of the decryption key. A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.

**Using Symmetric Encryption**

A two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment.

Usually involves the use of a trusted key distribution center (KDC). Each party in the network shares a secret master key with the KDC. The KDC is responsible for generating session keys, and for distributing those keys to the parties involved, using the master keys to protect these session keys.

**Needham-Schroeder Protocol**

The Needham-Schroeder Protocol is the original, **basic key exchange protocol**, Used by 2 parties who both trusted a common key server, it gives one party the info needed to establish a session key with the other. Note that since the key server chooses the session key, it is capable of reading/forging any messages between A&B, which is why they need to trust it absolutely!

Note that all communications is between A&KDC and A&B, B&KDC don't talk directly (though indirectly a message passes from KDC via A to B, encrypted in B's key so that A is unable to read or alter it). Other variations of key distribution protocols can involve direct communications between B&KDC.

There is a critical flaw in the protocol, as shown. The message in step 3 can be decrypted, and hence understood, supposedly only by B. But if an opponent, X, has been able to compromise an old session key, then X can impersonate A and trick B into using the old key by simply replaying step 3. Admittedly, this is a much more unlikely occurrence than that an opponent has simply observed and recorded step 3. It can however be corrected by either using timestamps, or an additional nonce, with respective advantages and limitations, see text for discussion.

This example emphasises the need to be extremely careful in codifying assumptions, and tracking the timeliness of the flow of info in protocols. Designing secure protocols is not easy, and should not be done lightly. Great care and analysis is needed.

**One-Way Authentication**

With some refinement, the KDC strategy is a candidate for securing electronic mail. Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated. For a message with content *M*, the sequence is as shown. This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A. As specified, the protocol does not protect against replays. Some measure of defense could be provided by including a timestamp with the message. However, because of the potential delays in the e-mail process, such timestamps may have limited usefulness.

**Kerberos**

Kerberos is an authentication service developed as part of Project Athena at MIT, and is one of the best known and most widely implemented **trusted third party** key distribution systems.

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption. Two versions of Kerberos are in common use: v4 & v5.

**Kerberos Requirements**

In a more open environment, in which network connections to other machines are supported, an approach that requires the user to prove his or her identity for each service invoked, and also require that servers prove their identity to clients, is needed to protect user information and resources housed at the server. Kerberos supports this approach, and assumes a distributed client/server architecture that employs one or more Kerberos servers to provide an authentication service. The first published report on Kerberos [STEI88] listed the following requirements:

• **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

• **Reliable**: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

• **Transparent**: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

• **Scalable**: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

To support these requirements, Kerberos is a trusted third-party authentication service that uses a protocol based on that proposed by Needham and Schroeder

**Kerberos v4 Overview**

The core of Kerberos is the Authentication and Ticket Granting Servers – these are trusted by all users and servers and must be securely administered. The protocol includes a sequence of interactions between the client, AS(Authentication Server), TGT- Ticket Granting Servers and desired server. Version 4 of Kerberos makes use of DES(Data Encryption Standard), in a rather elaborate protocol, to provide the authentication service. Viewing the protocol as a whole, it can be difficult to see the need for the many elements contained therein.

**Kerberos 4 Overview**

****

Stallings Figure diagrammatically summarizes the Kerberos v4 authentication dialogue, with 3 pairs of messages.

**Kerberos Realms(Domain)**

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers is referred to as a Kerberos realm. A Kerberos realm is a set of managed nodes that share the same Kerberos database, and are part of the same administrative domain. If have multiple realms, their Kerberos servers must share keys and trust each other.

****

Stallings Figure shows the authentication messages where service is being requested from **another domain**. The ticket presented to the remote server indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request. One problem presented by the foregoing approach is that it does not scale well to many realms, as each pair of realms need to share a key.

**Remote User Authentication**

we presented one approach to the use of public-key encryption for the purpose of session key distribution. This protocol assumes that each of the two parties is in possession of the current public key of the other. It may not be practical to require this assumption.

A protocol using timestamps is provided that uses a central system, referred to as an authentication server (AS), because it is not actually responsible for secret key distribution. Rather, the AS provides public-key certificates. The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS. The timestamps protect against replays of compromised keys. This protocol is compact but, as before, requires synchronization of clocks. Note the authors themselves spotted a flaw in it and submitted a revised version of the algorithm Protocols that appeared secure were revised after additional analysis.

**One-Way Authentication**

We have already presented public-key encryption approaches that are suited to electronic mail, including the straightforward encryption of the entire message for confidentiality, authentication or both .These approaches require that either the sender know the recipient's public key (confidentiality) or the recipient know the sender's public key (authentication) or both (confidentiality plus authentication). In addition, the public-key algorithm must be applied once or twice to what may be a long message.

If confidentiality is the primary concern, then better to encrypt the message with a one-time secret key, and also encrypt this one-time key with B's public key. If authentication is the primary concern, then a digital signature may be enough (but could be replaced by an opponent). To counter such a scheme, both the message and signature can be encrypted with the recipient's public key. The latter two schemes require that B know A's public key and be convinced that it is timely. An effective way to provide this assurance is the digital certificate.

**Federated Identity Management**

Federated identity management is a relatively new concept dealing with the use of a common identity management scheme across multiple enterprises and numerous applications and supporting many thousands, even millions of users. Identity management is a centralized, automated approach to provide enterprise-wide access to resources by employees and other authorized individuals, defining an identity for each user (human or process), associating attributes with the identity, and enforcing a means by which a user can verify identity. Its principal elements are:

• **Authentication:**confirmating user corresponds to the user name provided.

• **Authorization:** granting access to services/resources given user authentication.

• **Accounting:** process for logging access and authorization.

• **Provisioning:** enrollment of users in the system.

• **Workflow automation:** movement of data in a business process.

• **Delegated administration:** use of role-based access control to grant permissions.

• **Password synchronization:** Creating a process for single sign-on (SSO) or reduced sign-on (RSO).

• **Self-service password reset:** enable user to modify their password

• **Federation:** process where authentication and permission will be passed on from one system to another, usually across multiple enterprises, reducing the number of authentications needed by the user.

Kerberos contains a number of the elements of an identity management system.

**Identity Management**



**Identity federation is, in essence, an extension of identity management** to multiple security domains. Federated identity management refers to the agreements, standards, and technologies that enable the portability of identities, identity attributes, and entitlements across multiple enterprises and numerous applications and supporting many thousands, even millions, of users. Stallings illustrates entities and data flows in a generic federated identity management architecture. The identity provider acquires attribute information through dialogue and protocol exchanges with users and administrators. Identity management enables the user to provide this information once, so that it is maintained in a single place and released to data consumers in accordance with authorization and privacy policies. Service providers are entities that obtain and employ data maintained and provided by identity providers, often to support authorization decisions and to collect audit information. A service provider can be in the same domain as the user and the identity provider. The power of this approach is for federated identity management, in which the service provider is in a different domain.



Federated identity management uses a number of standards as the building blocks for secure identity exchange across different domains or heterogeneous systems. In essence, organizations issue some form of security tickets for their users that can be processed by cooperating partners. Identity federation standards are thus concerned with defining these tickets, in terms of content and format, providing protocols for exchanging tickets and performing a number of management tasks. These tasks include configuring systems to perform attribute transfers and identity mapping, and performing logging and auditing functions. The principal underlying standard for federated identity is the Security Assertion Markup Language (SAML), which SAML is an XML-based language that defines the exchange of security information between online business partners. SAML conveys authentication information in the form of assertions about subjects. Assertions are statements about the subject issued by an authoritative entity.

SAML is part of a broader collection of standards being issued by OASIS (Organization for the Advancement of Structured Information Standards) for federated identity management. For example, WS-Federation enables browser-based federation; it relies on a security token service to broker trust of identities, attributes, and authentication between participating Web services. The challenge with federated identity management is to integrate multiple technologies, standards, and services to provide a secure, user-friendly utility. The key, as in most areas of security and networking, is the reliance on a few mature standards widely accepted by industry. Federated identity management seems to have reached this level of maturity.

**INTRUDERS**

Unauthorized intrusion into a computer system or network is one of the most serious threats to computer security. User trespass can take the form of unauthorized logon to a machine or, in the case of an authorized user, acquisition of privileges or performance of actions beyond those that have been authorized. Software trespass can take the form of a virus, worm, or Trojan horse.

All these attacks relate to network security because system entry can be achieved by means of a network. However, these attacks are not confined to network-based attacks. A user with access to a local terminal may attempt trespass without using an intermediate network. A virus or Trojan horse may be introduced into a system by means of a diskette. Only the worm is a uniquely network phenomenon. Thus, system trespass is an area in which the concerns of network security and computer security overlap.

At first, we examine the nature of attacks to systems. One of the common threats to security of a system is intruder, also referred to as hacker or cracker. Intruders are classified into three classes.

1. **Masquerader:** Unauthorized individual who penetrates a system’s access control to exploit an authorized user’s account.
2. **Misfeasor:** Genuine user accesses data for which he is not authorized or he is authorized for access but misuses his or her privileges.
3. **Clandestine user:** A person who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

Masquerader is mostly an outsider whereas misfeasor is generally an insider. Clandestine user can be outsider or insider.

Intruder attacks range from benign to the serious. Sometimes people explore internets and see what is out there. This is benign type. There are cases where individuals attempt to read/modify data in the system, though unauthorized. This is serious attack. These attacks are still not under total control. We point out some techniques for intrusion.

**Intrusion techniques**

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Generally this requires the intruder to acquire information that is protected. Often this information is in the form of a user password.

With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with noprotection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one of two ways:

* **One-way function:** The system stores only the value of a function based on the user'spassword. When the user presents a password, the system transforms 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 one-way 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.

If one or both of these countermeasures are in place, some effort is needed for a potential intruder to learn passwords. On the basis of a survey of the literature and interviews with a number of password crackers, the techniques for learning passwords are:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords (those of one to three characters).
3. 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.
4. 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.
5. Try users' phone numbers, Social Security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.
7. Use a Trojan horse (described in next unit) to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

The first six methods are various ways of guessing a password. If an intruder has to verify the guess by attempting to log in, it is a tedious and easily countered means of attack. For example, a system can simply reject any login after three password attempts, thus requiring the intruder to reconnect to the host to try again. Under these circumstances, it is not practical to try more than a handful of passwords. However, the intruder is unlikely to try such crude methods. For example, if an intruder can gain access with a low level of privileges to an encrypted password file, then the strategy would be to capture that file and then use the encryption mechanism of thatparticular system at leisure until a valid password that provided greater privileges was discovered.

Guessing attacks are feasible, and indeed highly effective, when a large number of guesses can be attempted automatically and each guess verified, without the guessing process being detectable.

The seventh method of attack, the Trojan horse, can be particularly difficult to counter. An example of a program that bypasses access controls was cited in the literature. A low-privilege user produced a game program and invited the system operator to use it in his or her spare time. The program did indeed play a game, but in the background it also contained code to copy the password file, which was unencrypted but access protected, into the user's file. Because the game was running under the operator's high-privilege mode, it was able to gain access to the password file.

The eighth attack listed namely line tapping, is a matter of physical security. It can be countered with link encryption techniques. Other intrusion techniques do not require learning a password. Intruders can get access to a system by exploiting attacks such as buffer overflows on a program that runs with certain privileges. Privilege escalation can be done this way as well.

We turn now to a discussion of the two principal countermeasures: detection and prevention. Detection is concerned with learning of an attack, either before or after its success. Prevention is a challenging security goal and an uphill battle at all times. The difficulty stems from the fact that the defender must attempt to thwart all possible attacks, whereas the attacker is free to try to find the weakest link in the defense chain and attack at that point.

**INTRUSION DETECTION**

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

1. 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. Even if the detectionis not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

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. Of course, we cannot expect that there will be a crisp, exact distinction between an attack by an intruder and the normal use of resources by an authorized user. Rather, we must expect that there will be some overlap.

Fig. suggests, in very abstract terms, the nature of the task confronting thedesigner of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.



**Figure:** Profiles of Behavior of Intruders and Authorized User

There are studies postulating that one could, with reasonable confidence, distinguish between a masquerader and a legitimate user. Patterns of legitimate user behavior can be established by observing past history, and significant deviation from such patterns can be detected. The task of detecting a misfeasor (legitimate user performing in an unauthorized fashion) is more difficult, in that the distinction between abnormal and normal behavior may be small. Such violations would be undetectable solely through the search for anomalous behavior. However, misfeasor behavior might nevertheless be detectable by intelligent definition of the class of conditions that suggest unauthorized use. Finally, the detection of the clandestine user is still beyond the scope of purely automated techniques.

Approaches for intrusion detection are

1. **Statistical anomaly detection:** Involves the collection of data relating to the behavior oflegitimate 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.
	1. Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
	2. Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
2. **Rule-based detection:** Involves an attempt to define a set of rules that can be used to decidethat a given behavior is that of an intruder.
	1. Anomaly detection: Rules are developed to detect deviation from previous usage patterns.
	2. Penetration identification: An expert system approach that searches for suspicious behavior.

In a nutshell, statistical approaches attempt to define normal or expected behavior, whereas rule-based approaches attempt to define proper behavior.

In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders, who are unlikely to mimic the behavior patterns of the accounts they appropriate. On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, rule-based approaches may be able to recognize events and sequences that, in context,

reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks.

1. **Audit Records**

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

* **Native audit records:** Virtually all multiuser operating systems include accountingsoftware that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.
* **Detection-specific audit records:** A collection facility can be implemented thatgenerates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

A good example of detection-specific audit records reported in literature contains the following fields:

* **Subject:** Initiators of actions. A subject is typically a terminal user but might also be aprocess acting on behalf of users or groups of users. All activity arises through commands issued by subjects. Subjects may be grouped into different access classes, and these classes may overlap.
* **Action:** Operation performed by the subject on or with an object; for example, login,read, perform I/O, execute.
* **Object:** Receptors of actions. Examples include files, programs, messages, records,terminals, printers, and user- or program-created structures. When a subject is the recipient of an action, such as electronic mail, then that subject is considered an object. Objects may be grouped by type. Object granularity may vary by object type and by environment. For example, database actions may be audited for the database as a whole or at the record level.
* **Exception-Condition:** Denotes which, if any, exception condition is raised on return.
* **Resource-Usage:** A list of quantitative elements in which each element gives the amountused of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time).
* **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

The audit records provide input to the intrusion detection using statistical anomaly detection in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. An analysis of audit records over a period of time can be used to determine the activity profile of the average user. Thus, the audit records serve to define typical behavior. Second, current audit records are the input used to detect intrusion. That is, the intrusion detection model analyzes incoming audit records to determine deviation from average behavior.

As far as rule based intrusion detection, audit records are examined as they are generated, and they are matched against the rule base. If a match is found, then the user's *suspicion rating* is increased. If enough rules are matched, then the rating will pass a threshold that results in the reporting of an anomaly.

**Distributed Intrusion Detection**

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork. 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.

The major issues in the design of a distributed intrusion detection system are:

* A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.
* One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be

transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable.

* Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of collection and analysis of all audit data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure. With a decentralized architecture, there are more than one analysis centers, but these must coordinate their activities and exchange information.

**PASSWORD PROTECTION AND MANAGEMENT**

**One important element of intrusion prevention is password management**, with the goal of preventing unauthorized users from having access to the passwords of others.

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

* The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
* The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.
* The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

**The Vulnerability of Passwords**

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, 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 into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. The encryption routine, known as 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. This process is repeated for a total of 25 encryptions. The resulting 64-bit output is then translated into an 11-character sequence. The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID. This method has been shown to be secure against a variety of cryptanalytic attacks. The salt serves three purposes:

* It prevents duplicate 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.

When a user attempts to log on to 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. The salt and user-supplied passwords are used as input to the encryption routine. If the result matches the stored value, the password is accepted.

The encryption routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25. However, since the original design of this algorithm, two changes have occurred. First, newer implementations of the algorithm itself have resulted in speedups. Second, hardware performance continues to increase, so that any software algorithm executes more quickly.

Thus, there are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine. The attacker should be able to check hundreds and perhaps thousands of possible passwords with little resource consumption. In addition, 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. This enables the opponent to run through many thousands of possible passwords in a reasonable period.

Even stupendous guessing rates do not yet make it feasible for an attacker to use a dumb brute-force technique of trying all possible combinations of characters to discover a password. Instead, password crackers rely on the fact that some people choose easily guessable passwords.

Some users, when permitted to choose their own password, pick one that is absurdly short. This makes password cracking very easy.

Password length is only part of the problem. 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. This makes the job of password cracking straightforward. The cracker simply has to test the password file against lists of likely passwords. Because many people use guessable passwords, such a strategy should succeed on virtually all systems.

Password cracking programs could use one or all of the following strategies.

1. Try the user's name, initials, account name, and other relevant personal information. In all, 130 different permutations for each user were tried.
2. Try words from various dictionaries. The author compiled a dictionary of over 60,000 words, including the online dictionary on the system itself, and various other lists as shown.
3. Try various permutations 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.
4. Try various capitalization permutations on the words from step 2 that were not considered in step 3. This added almost 2 million additional words to the list.

**Access Control**

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user. There are flaws in this strategy too.

* Many systems, including most UNIX systems, are susceptible to unanticipated break-ins. Once an attacker has gained access by some means, he or she may wish to obtain a collection of passwords in order to use different accounts for different logon sessions to decrease the risk of detection. Or a user with an account may desire another user's account to access privileged data or to sabotage the system.
* An accident of protection might render the password file readable, thus compromising all the accounts.
* Some of the users have accounts on other machines in other protection domains, and they use the same password. Thus, if the passwords could be read by anyone on one machine, a machine in another location might be compromised.

Thus, a more effective strategy would be to force users to select passwords that are difficult to guess.

**Password Selection Strategies**

The lesson from the two experiments just described is that, left to their own devices, many users choose a password that is too short or too easy to guess. At the other extreme, if users are assigned passwords consisting of eight randomly selected printable characters, password cracking is effectively impossible. But it would be almost as impossible for most users to remember their passwords. Fortunately, even if we limit the password universe to strings of characters that are reasonably memorable, the size of the universe is still too large to permit practical cracking. Our goal, then, is to eliminate guessable passwords while allowing the user to select a password that is memorable. Four basic techniques are in use:

* User education
* Computer-generated passwords
* Reactive password checking
* Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines. Others may not be good judges of what is a strong password. For example, many users (mistakenly) believe that reversing a word or capitalizing the last letter makes a password unguessable.

**Computer-generated passwords** also have problems. If the passwords are quite randomin nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer-generated password schemes have a history of poor acceptance by users.

A **reactive password checking** strategy is one in which the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. This tactic has a number of drawbacks. First, it is resource intensive if the job is done right. Because a determined opponent who is able to steal a password file can devote full CPU time to the task for hours or even days, an effective reactive password checker is at a distinct disadvantage. Furthermore, any existing passwords remain vulnerable until the reactive password checker finds them.

The most promising approach to improved password security is a **proactive passwordchecker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

The trick with a proactive password checker is to strike a balance between user acceptability and strength. If the system rejects too many passwords, users will complain that it is too hard to select a password. If the system uses some simple algorithm to define what is acceptable, this provides guidance to password crackers to refine their guessing technique.

**Cryptography and Network Security - INTRUDERS**

A significant security problem for networked systems is hostile, or at least unwanted, trespass being unauthorized login or use of a system, by local or remote users; or by software such as a virus, worm, or Trojan horse.

One of the two most publicized threats to security is the intruder (or hacker or cracker), which Anderson identified three classes of:

**• Masquerader:** An individual who is not authorized to use the computer (outsider)

**• Misfeasor:** A legitimate user who accesses unauthorized data, programs, or resources (insider)

**• Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection (either)

Intruder attacks range from the kind/benign (simply exploring net to see what is there); to the serious (who attempt to read privileged data, perform unauthorized modifications, or disrupt system)

The intruder threat has been well publicized, particularly because of the famous “Wily Hacker “incident of 1986–1987, documented by Cliff Stoll. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However, there is no way in advance to know whether an intruder will be benign or malign. Intruders may use compromised systems to launch attacks on other systems, further degrading performance.

One of the results of the growing awareness of the intruder problem has been the establishment of a number of computer emergency response teams (CERTs). These cooperative ventures collect information about system vulnerabilities and disseminate it to systems managers.

*Knowing the standard attack methods is a key element in limiting your vulnerability/weakness. The basic* aim is to gain access and/or increase privileges on some system. *A basic technique for gaining access is to acquire a user (preferably administrator) password, so the attacker can login and exercise all the* access rights of the account owner.

**Password guessing** is a common attack. If an attacker has obtained a poorly protected password file, then can mount attack off-line, so target is unaware of its progress. Some O/S take less care than others with their password files. If have to actually attempt to login to check guesses, then system should detect an abnormal number of failed logins, and hence trigger appropriate countermeasures by admins/security. Likelihood of success depends very much on how well the passwords are chosen. Unfortunately, users often don’t choose well (see later).

**Password capturing**

There is also a range of ways of "capturing" a login/password pair, from the low-tech looking over the shoulder, to the use of Trojan Horse programs (eg. game program or nifty utility with a covert function as well as the overt behaviour), to sophisticated network monitoring tools, or extracting recorded info after a successful login - say from web history or cache, or last number dialled memory on phones etc. Need to educate users to be aware of whose around, to check they really are interacting with the computer system (trusted path), to beware of unknown source s/w, to use secure network connections (HTTPS, SSH, SSL), to flush browser/phone histories after use etc.

**Intrusion Detection**

Inevitably, the best intrusion prevention system will fail. A system’s second line of defense is intrusion detection, which aims to detect intrusions so can: block access & minimize damage if detected quickly; act as deterrent given chance of being caught; or can collect info on intruders to improve future security.

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. This is imperfect at best.

**Approaches to Intrusion Detection**

Can identify the following approaches to intrusion detection:

1. **Statistical anomaly detection:** collect data relating to the behavior of legitimate users, then use statistical tests to determine with a high level of confidence whether new behavior is legitimate user behavior or not.

 a. Threshold detection: define thresholds, independent of user, for the frequency of occurrence of events.

 b. Profile based: develop profile of activity of each user and use to detect changes in the behavior

1. **Rule-based detection:** attempt to define a set of rules used to decide if given behavior is an intruder

 a. Anomaly detection: rules detect deviation from previous usage patterns

 b. Penetration identification: expert system approach that searches for suspicious behavior

1. **Audit record**

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

• Native audit records: Virtually all main O/S’s include accounting software that collects information on user activity, advantage is its already there, disadvantage is it may not contain the needed information

• Detection-specific audit records: implement collection facility to generates custom audit records with desired info, advantage is it can be vendor independent and portable, disadvantage is extra overhead involved

**Statistical anomaly detection**

**Statistical anomaly detection** techniques cover threshold detection and profile-based systems.

Threshold detection involves counting no occurrences of a specific event type over an interval of time, if count surpasses a reasonable number, then intrusion is assumed. By itself, is a crude and ineffective detector of even moderately sophisticated attacks.

Profile-based anomaly detection focuses on characterizing past behavior of users or groups, and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert. Foundation of this approach is analysis of audit records.

**Analysis of audit records**

An **analysis of audit records** over a period of time can be used to determine the activity profile of the average user. Then current audit records are used as input to detect intrusion, by analyzing incoming audit records to determine deviation from average behavior. Examples of metrics that are useful for profile-based intrusion detection are: counter, gauge, interval timer, resource use. Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits, such as: Mean and standard deviation, Multivariate, Markov process, Time series, Operational; as discussed in the text.

The main advantage of the use of statistical profiles is that a prior knowledge of security flaws is not required. Thus it should be readily portable among a variety of systems.

**Rule-based techniques for Intrusion detection**

**Rule-based techniques** detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious. Can characterize approaches as either anomaly detection or penetration identification, although there is overlap. Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. Historical audit records are analyzed to identify usage patterns and to automatically generate rules that describe those patterns. Current behavior is then observed and matched against the set of rules to see if it conforms to any historically observed pattern of behavior. As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system.

Rule-based penetration identification takes a very different approach based on expert system technology. It uses rules for identifying known penetrations or penetrations that would exploit known weaknesses, or identify suspicious behavior. The rules used are specific to machine and operating system. The rules are generated by “experts”, from interviews of system administrators and security analysts. Thus the strength of the approach depends on the skill of those involved in setting up the rules.

**Base-Rate Fallacy**

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms. Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. A study of existing intrusion detection systems indicated that current systems have not overcome the problem of the base-rate fallacy.

**Distributed Intrusion Detection**

Until recently, work on intrusion detection systems focused on single-system standalone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork, where a more effective defense can be achieved by coordination and cooperation among intrusion detection systems across the network.

Porras points out the following major issues in the design of distributed IDS:

• A distributed intrusion detection system may need to deal with different audit record formats

• One or more nodes in the network will serve as collection and analysis points for the data, which must be securely transmitted to them

• Either a centralized (single point, easier but bottleneck) or decentralized (multiple centers must coordinate) architecture can be used.



Stallings Figure shows the overall architecture, consisting of three main components, of the system independent distributed IDS developed at the University of California at Davis. The components are:

• **Host agent module:** audit collection module operating as a background process on a monitored system

• **LAN monitor agent module:** like a host agent module except it analyzes LAN traffic

• **Central manager module:** Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion

**Distributed Intrusion Detection – Agent Implementation**



Stallings Figure shows the general approach that is taken. The agent captures each native O/S audit record, & applies a filter that retains only records of security interest. These records are then reformatted into a standardized format .Then a template-driven logic module analyzes the records for suspicious activity. When suspicious activity is detected, an alert is sent to the central manager. The central manager includes an expert system that can draw inferences from received data. The manager may also query individual systems for copies of to correlate with those from other agents.

**Intrusion Prevention**

**Honeypots**

Honeypots are decoy systems, designed to lure a potential attacker away from critical systems, and:

• divert an attacker from accessing critical systems

• collect information about the attacker’s activity

• encourage the attacker to stay on the system long enough for administrators to respond

These systems are filled with fabricated information designed to appear valuable but which any legitimate user of the system wouldn’t access, thus, any access is suspect.

They are instrumented with sensitive monitors and event loggers that detect these accesses and collect information about the attacker’s activities.

Have seen evolution from single host honeypots to honeynets of multiple dispersed systems.

The IETF Intrusion Detection Working Group is currently drafting standards to support interoperability of IDS info (both honeypot and normal IDS) over a wide range of systems & O/S’s.

**Password Management**

The front line of defense against intruders is the password system, where a user provides a name/login identifier (ID) and a password. The password serves to authenticate the ID of the individual logging on to the system. Passwords are usually stored encrypted rather than in the clear (which would make them more vulnerable to theft). Unix systems traditionally used a multiple DES variant with salt as a one-way hash function .More recent O/S’s use a cryptographic hash function (eg. MD5- Message Digest). The file containing these passwords hashes needs access control protections to make guessing attacks harder.

Studies have shown that users tend to choose poor passwords too often.

A study at Purdue University in 1992 observed password change choices on 54 machines, for 7000 users, and found almost 3% of the passwords were three characters or fewer in length, easily exhaustively searched!

Password length is only part of the problem, since many people pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward.

A study by Klein 1990 collected UNIX password files, containing nearly 14,000 encrypted passwords, and found nearly one-fourth of these passwords were guessable.

A strategy is needed to force users to select passwords that are difficult to guess.

**Goal is to eliminate guessable passwords while allowing user to select a memorable password.** **Four basic techniques are in use: education, computer generation, reactive checking & proactive checking.**

The **user education strategy** tells users the importance of using hard-to-guess passwords and provides guidelines for selecting strong passwords, but it needs their cooperation. The problem is that many users will simply ignore the guidelines.

**Computer-generated passwords** create a password for the user, but have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer-generated password schemes have a history of poor acceptance by users. FIPS PUB 181 defines one of the best-designed automated password generators. The standard includes not only a description of the approach but also a complete listing of the C source code of the algorithm, which generates words by forming a random set of pronounceable syllables and concatenating them to form a word.

**A reactive password checking** strategy is one in which the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. Drawbacks are that it is resource intensive if the job is done right, and any existing passwords remain vulnerable until the reactive password checker finds them.

The most promising approach to improved password security is **a proactive password checker,** where a user is allowed to select his or her own password, but the system checks to see if it is allowable and rejects it if not. The trick is to strike a balance between user acceptability and strength.

The first approach is a simple system for rule enforcement, enforcing say guidelines from user education. May not be good enough. Another approach is to compile a large dictionary of possible “bad”passwords, and check user passwords against this disapproved list. But this can be very large & slow to search. A third approach is based on rejecting words using either a Markov model of guessable passwords, or a Bloom filter. Both attempt to identify good or bad passwords without keeping large dictionaries. See text for details.