- 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.

A good example of a distributed intrusion detection system is one developed at the University of California at Davis [HEBE92, SNAP91]. A similar approach has been taken for a project at Purdue [SPAF00, BALA98]. Figure 11.2 shows the overall architecture, which consists of three main components:

- **Host agent module:** An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.
- **LAN monitor agent module:** Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
- **Central manager module:** Receives reports from LAN monitor and host agents, and processes and correlates these reports to detect intrusion.

The scheme is designed to be independent of any operating system or system auditing implementation. Figure 11.3 shows the general approach that is taken. The agent captures each audit record produced by the native audit collection system. A filter is applied that retains only those records that are of security interest. These records are then reformatted into a standardized format referred to as the host audit

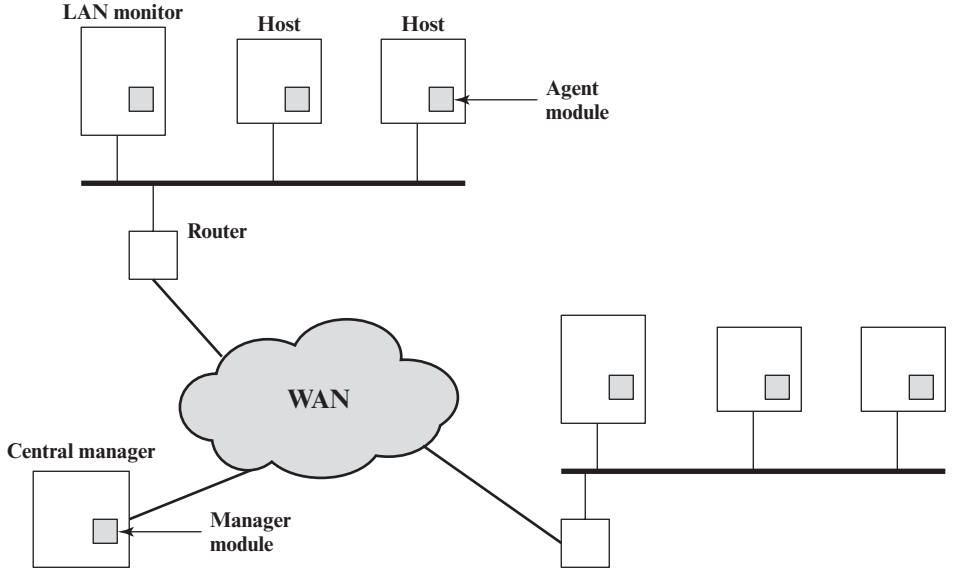

Figure 11.2 Architecture for Distributed Intrusion Detection

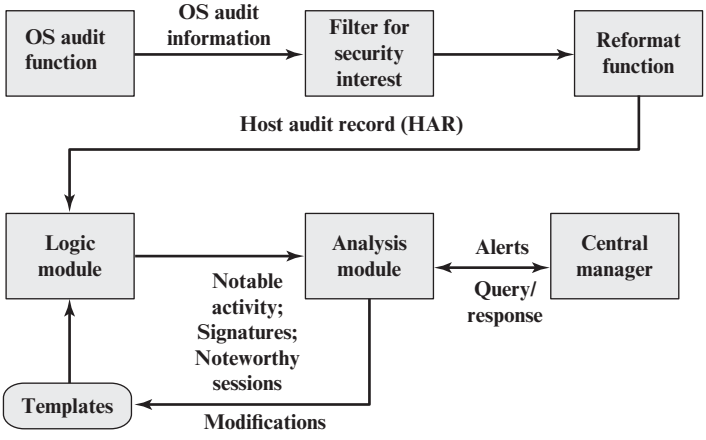

Figure 11.3 Agent Architecture

record (HAR). Next, a template-driven logic module analyzes the records for suspicious activity. At the lowest level, the agent scans for notable events that are of interest independent of any past events. Examples include failed file accesses, accessing system files, and changing a file's access control. At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures). Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.

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 HARs to correlate with those from other agents.

The LAN monitor agent also supplies information to the central manager. The LAN monitor agent audits host-host connections, services used, and volume of traffic. It searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as *rlogin*.

The architecture depicted in Figures 11.2 and 11.3 is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected.

Honeypots

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to

- 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 that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect. The system is instrumented with sensitive monitors and event loggers that detect these accesses and collect information about the attacker's activities. Because any attack against the honeypot is made to seem successful, administrators have time to mobilize and log and track the attacker without ever exposing productive systems.

The honeypot is a resource that has no production value. There is no legitimate reason for anyone outside the network to interact with a honeypot. Thus, any attempt to communicate with the system is most likely a probe, scan, or attack. Conversely, if a honeypot initiates outbound communication, the system has probably been compromised.

Initial efforts involved a single honeypot computer with IP addresses designed to attract hackers. More recent research has focused on building entire honeypot networks that emulate an enterprise, possibly with actual or simulated traffic and data. Once hackers are within the network, administrators can observe their behavior in detail and figure out defenses.

Honeypots can be deployed in a variety of locations. Figure 11.4 illustrates some possibilities. The location depends on a number of factors, such as the type of information the organization is interested in gathering and the level of risk that organizations can tolerate to obtain the maximum amount of data.

A honeypot outside the external firewall (**location 1**) is useful for tracking attempts to connect to unused IP addresses within the scope of the network. A honeypot at this location does not increase the risk for the internal network. The danger of having a compromised system behind the firewall is avoided. Further, because the honeypot attracts many potential attacks, it reduces the alerts issued by the firewall and by internal IDS sensors, easing the management burden. The disadvantage of an external honeypot is that it has little or no ability to trap internal attackers, especially if the external firewall filters traffic in both directions.

The network of externally available services, such as Web and mail, often called the DMZ (demilitarized zone), is another candidate for locating a honeypot (**location 2**). The security administrator must assure that the other systems in the DMZ are secure against any activity generated by the honeypot. A disadvantage of this location is that a typical DMZ is not fully accessible, and the firewall typically blocks traffic to the DMZ that attempts to access unneeded services. Thus, the firewall either has to open up the traffic beyond what is permissible, which is risky, or limit the effectiveness of the honeypot.

A fully internal honeypot (**location 3**) has several advantages. Its most important advantage is that it can catch internal attacks. A honeypot at this location can also detect a misconfigured firewall that forwards impermissible traffic from the Internet to the internal network. There are several disadvantages. The most serious of these is if the honeypot is compromised so that it can attack other internal systems. Any further traffic from the Internet to the attacker is not blocked by the firewall because it is regarded as traffic to the honeypot only. Another difficulty for this honeypot location is that, as with location 2, the firewall must adjust its filtering to allow traffic to the honeypot, thus complicating firewall configuration and potentially compromising the internal network.

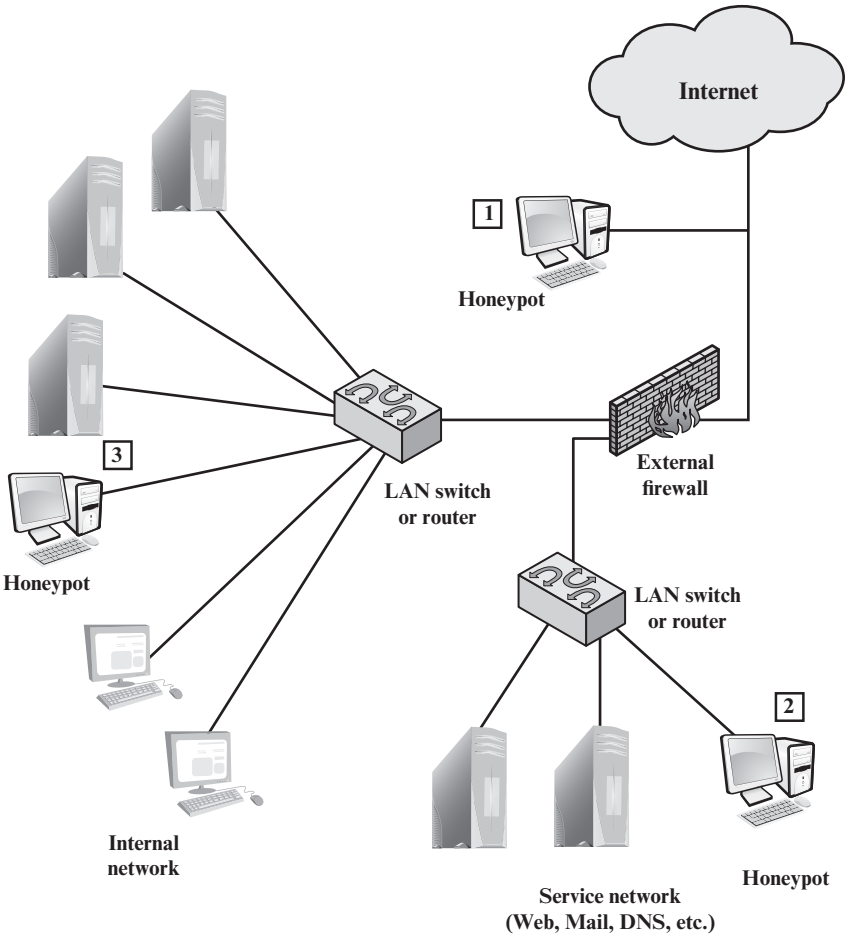

Figure 11.4 Example of Honeypot Deployment

Intrusion Detection Exchange Format

To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group. The purpose of the working group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to management systems that may need to interact with them.

The working group issued the following RFCs in 2007:

- **Intrusion Detection Message Exchange Requirements (RFC 4766):** This document defines requirements for the Intrusion Detection Message Exchange Format (IDMEF). The document also specifies requirements for a communication protocol for communicating IDMEF.

- **The Intrusion Detection Message Exchange Format (RFC 4765):** This document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented, an XML Document Type Definition is developed, and examples are provided.
- **The Intrusion Detection Exchange Protocol (RFC 4767):** This document describes the Intrusion Detection Exchange Protocol (IDXP), an application-level protocol for exchanging data between intrusion detection entities. IDXP supports mutual authentication, integrity, and confidentiality over a connection-oriented protocol.

Figure 11.5 illustrates the key elements of the model on which the intrusion detection message exchange approach is based. This model does not correspond to any particular product or implementation, but its functional components are the key elements of any IDS. The functional components are as follows:

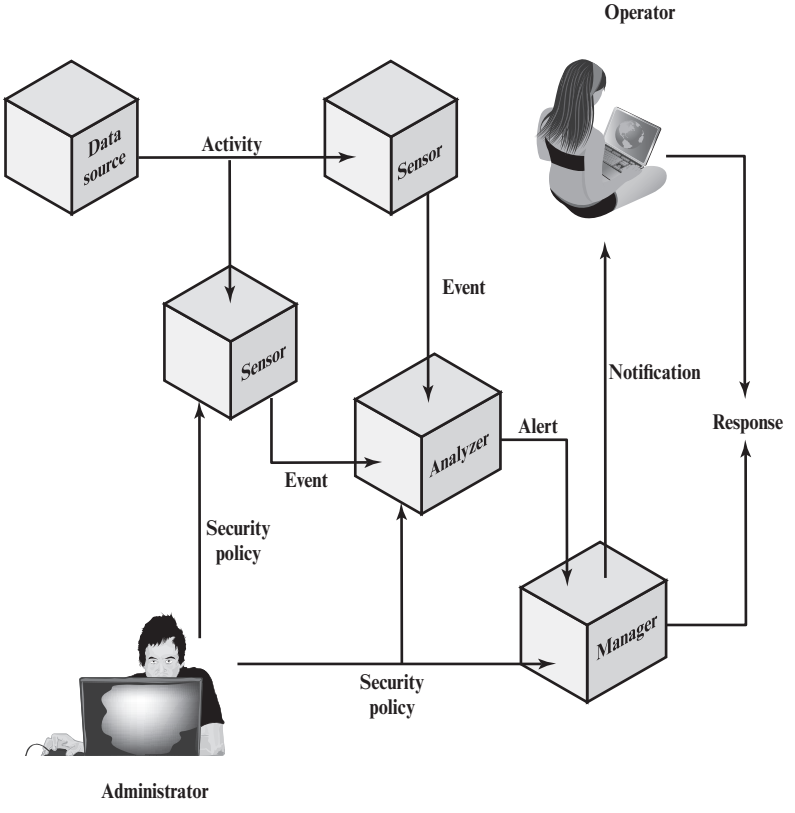

Figure 11.5 Model for Intrusion Detection Message Exchange

- **Data source:** The raw data that an IDS uses to detect unauthorized or undesired activity. Common data sources include network packets, operating system audit logs, application audit logs, and system-generated checksum data.
- **Sensor:** Collects data from the data source. The sensor forwards events to the analyzer.
- **Analyzer:** The ID component or process that analyzes the data collected by the sensor for signs of unauthorized or undesired activity or for events that might be of interest to the security administrator. In many existing IDSs, the sensor and the analyzer are part of the same component.
- **Administrator:** The human with overall responsibility for setting the security policy of the organization, and, thus, for decisions about deploying and configuring the IDS. This may or may not be the same person as the operator of the IDS. In some organizations, the administrator is associated with the network or systems administration groups. In other organizations, it's an independent position.
- **Manager:** The ID component or process from which the operator manages the various components of the ID system. Management functions typically include sensor configuration, analyzer configuration, event notification management, data consolidation, and reporting.
- **Operator:** The human that is the primary user of the IDS manager. The operator often monitors the output of the IDS and initiates or recommends further action.

In this model, intrusion detection proceeds in the following manner. The sensor monitors data sources looking for suspicious **activity**, such as network sessions showing unexpected telnet activity, operating system log file entries showing a user attempting to access files to which he or she is not authorized to have access, and application log files showing persistent login failures. The sensor communicates suspicious activity to the analyzer as an **event**, which characterizes an activity within a given period of time. If the analyzer determines that the event is of interest, it sends an **alert** to the manager component that contains information about the unusual activity that was detected, as well as the specifics of the occurrence. The manager component issues a **notification** to the human operator. A **response** can be initiated automatically by the manager component or by the human operator. Examples of responses include logging the activity; recording the raw data (from the data source) that characterized the event; terminating a network, user, or application session; or altering network or system access controls. The **security policy** is the predefined, formally documented statement that defines what activities are allowed to take place on an organization's network or on particular hosts to support the organization's requirements. This includes, but is not limited to, which hosts are to be denied external network access.

The specification defines formats for event and alert messages, message types, and exchange protocols for communication of intrusion detection information.

11.3 PASSWORD MANAGEMENT

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

In this subsection, we outline the main forms of attack against password-based authentication and briefly outline a countermeasure strategy. The remainder of Section 11.3 goes into more detail on the key countermeasures.

Typically, a system that uses password-based authentication maintains a password file indexed by user ID. One technique that is typically used is to store not the user's password but a one-way hash function of the password, as described subsequently.

We can identify the following attack strategies and countermeasures:

- **Offline dictionary attack:** Typically, strong access controls are used to protect the system's password file. However, experience shows that determined hackers can frequently bypass such controls and gain access to the file. The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that ID/password combination. Countermeasures include controls to prevent unauthorized access to the password file, intrusion detection measures to identify a compromise, and rapid reissuance of passwords should the password file be compromised.
- **Specific account attack:** The attacker targets a specific account and submits password guesses until the correct password is discovered. The standard countermeasure is an account lockout mechanism, which locks out access to the account after a number of failed login attempts. Typical practice is no more than five access attempts.
- **Popular password attack:** A variation of the preceding attack is to use a popular password and try it against a wide range of user IDs. A user's tendency is to choose a password that is easily remembered; this unfortunately makes the

password easy to guess. Countermeasures include policies to inhibit the selection by users of common passwords and scanning the IP addresses of authentication requests and client cookies for submission patterns.

- **Password guessing against single user:** The attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password. Countermeasures include training in and enforcement of password policies that make passwords difficult to guess. Such policies address the secrecy, minimum length of the password, character set, prohibition against using well-known user identifiers, and length of time before the password must be changed.
- **Workstation hijacking:** The attacker waits until a logged-in workstation is unattended. The standard countermeasure is automatically logging the workstation out after a period of inactivity. Intrusion detection schemes can be used to detect changes in user behavior.
- **Exploiting user mistakes:** If the system assigns a password, then the user is more likely to write it down because it is difficult to remember. This situation creates the potential for an adversary to read the written password. A user may intentionally share a password, to enable a colleague to share files, for example. Also, attackers are frequently successful in obtaining passwords by using social engineering tactics that trick the user or an account manager into revealing a password. Many computer systems are shipped with preconfigured passwords for system administrators. Unless these preconfigured passwords are changed, they are easily guessed. Countermeasures include user training, intrusion detection, and simpler passwords combined with another authentication mechanism.
- **Exploiting multiple password use:** Attacks can also become much more effective or damaging if different network devices share the same or a similar password for a given user. Countermeasures include a policy that forbids the same or similar password on particular network devices.
- **Electronic monitoring:** If a password is communicated across a network to log on to a remote system, it is vulnerable to eavesdropping. Simple encryption will not fix this problem, because the encrypted password is, in effect, the password and can be observed and reused by an adversary.

The Use of Hashed Passwords

A widely used password security technique is the use of hashed passwords and a salt value. This scheme is found on virtually all UNIX variants as well as on a number of other operating systems. The following procedure is employed (Figure 11.6a). To load a new password into the system, the user selects or is assigned a password. This password is combined with a fixed-length **salt value** [MORR79]. In older implementations, this value is related to the time at which the password is assigned to the user. Newer implementations use a pseudorandom or random number. The password and salt serve as inputs to a hashing algorithm to produce a fixed-length hash code. The hash algorithm is designed to be slow to execute to thwart attacks. The hashed password is then stored, together with a plaintext copy of the salt, in

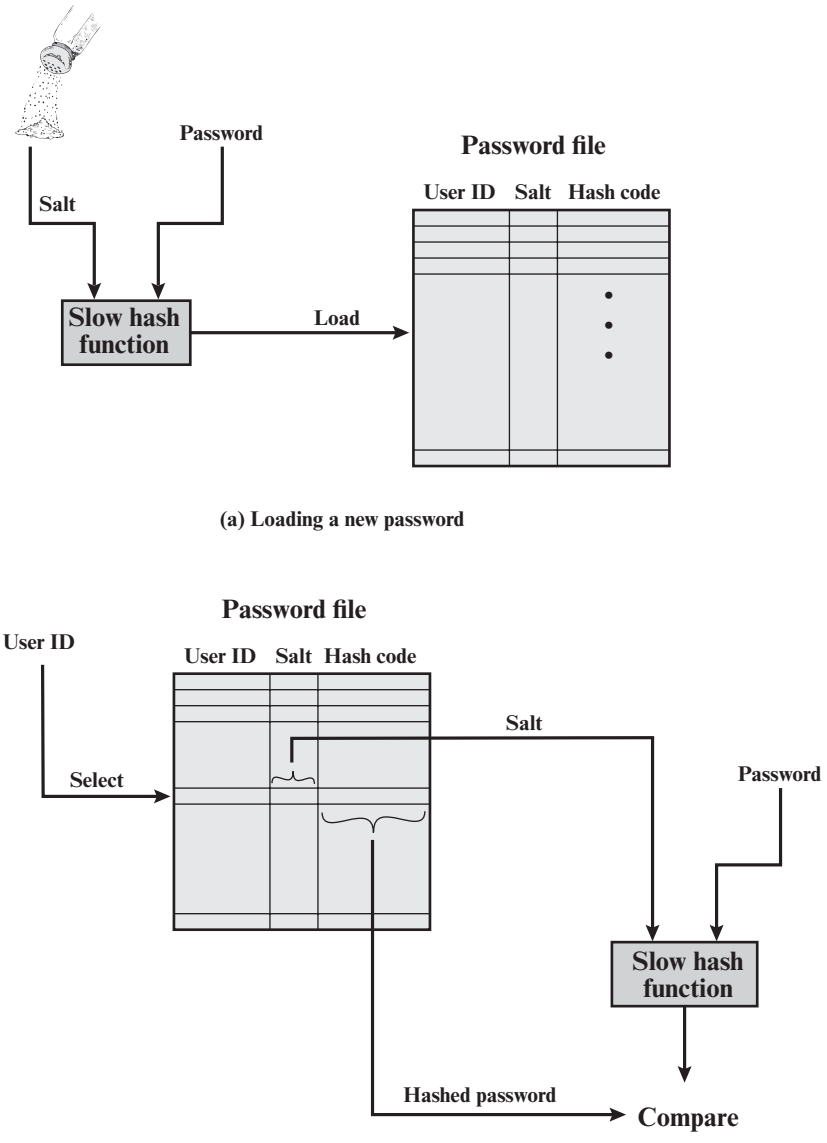

Figure 11.6 UNIX Password Scheme

the password file for the corresponding user ID. The hashed-password method has been shown to be secure against a variety of cryptanalytic attacks [WAGN00].

When a user attempts to log on to a UNIX system, the user provides an ID and a password (Figure 11.6b). 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 password are used as input to the encryption routine. If the result matches the stored value, the password is accepted.

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 different salt values. Hence, the hashed passwords of the two users will differ.
- It greatly increases the difficulty of offline dictionary attacks. For a salt of length b bits, the number of possible passwords is increased by a factor of 2^b, increasing the difficulty of guessing a password in a dictionary attack.
- It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them.

To see the second point, consider the way that an offline dictionary attack would work. The attacker obtains a copy of the password file. Suppose first that the salt is not used. The attacker's goal is to guess a single password. To that end, the attacker submits a large number of likely passwords to the hashing function. If any of the guesses matches one of the hashes in the file, then the attacker has found a password that is in the file. But faced with the UNIX scheme, the attacker must take each guess and submit it to the hash function once for each salt value in the dictionary file, multiplying the number of guesses that must be checked.

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 many 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 millions of possible passwords in a reasonable period.

UNIX Implementations Since the original development of UNIX, most implementations have relied on the following password scheme. 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 hash routine, known as crypt(3), is based on DES. A 12-bit salt value is used. The modified DES algorithm is executed 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 modification of the DES algorithm converts it into a one-way hash function. The crypt(3) 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.

This particular implementation is now considered woefully inadequate. For example, [PERR03] reports the results of a dictionary attack using a supercomputer. The attack was able to process over 50 million password guesses in about 80 minutes. Further, the results showed that for about $10,000 anyone should be able to do the same in a few months using one uniprocessor machine. Despite its known weaknesses, this UNIX scheme is still often required for compatibility with existing account management software or in multivendor environments.