J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

NETWORK SECURITY FUNDAMENTALS

Network security is defined as the task of keeping a collection of computer systems secure from attacks through the network. Network security is a critical part of any system connected to a public network, such as the Internet. However, the task of providing a comprehensive security solution is often misunderstood, as security is not a simple issue. Network security is a subset of computer security, which may be divided into four categories:

- *Information security* is protecting the data, which includes keeping copies in locations safe from physical threats such as fires.
- *Host security* is protecting a computer from physical access and access from the users on that system, as well as the ability to detect tampering and quickly recover from a security breach.
- *Network security* is protecting the system from attacks from other systems from the same organization.
- *Internetwork security* is protecting a system from attackers outside of the organization.

Many people consider internetwork security to be the primary emphasis of network security. However, security administrators must consider all four categories. Addressing only one category (i.e, adding a firewall) creates an unbalanced defense, which is often characterized as "locking the front door, but leaving the back door open." Connections from the Internet are an obvious threat, but others are equally important. As an example, hackers have used *social engineering* to walk into buildings assuming the identities of security officers doing a 'surprise audit.'

Goals of Security

It is useful to use systematic descriptions of security issues. There are three aspects of informational security: *Attacks* compromise systems, *mechanisms* protect systems, and *services* use mechanisms, but may be disrupted by attacks. Services are the implementation of the desired goals of security, which are as follows.

- Authority Ensures that the origin of the message is known and not falsified
- Confidentiality Ensures that only those authorized are able to see the privileged information
- Integrity Ensures that only authorized individuals are able to insert and change information
- Access control Ensures that the access to information is controlled
- Availability Ensures that only authorized individuals can use the services without disruption
- Nonrepudiation Ensures that neither the sender nor the receiver can refute the existence of the message

Why Security is Problematic

Here are some guidelines that represent important concepts governing security, and hopefully dispel some of the myths:

- *Network security is part of total security* Network security is a subset of a comprehensive security solution, and having strong network security without addressing the other issues may leave a site vulnerable to other penetration techniques.
- *Security is difficult to measure* There is no easy way to prove that security is sufficient. A compromised system indicates security was not sufficient, but this measurement comes too late. The worst case is a system that as been compromised for months or years, and the compromise never detected.
- *Security requires many experts* Distributed systems are complex, and proper analysis of security requires experts in operating systems internals, protocols, application implementation, and computer architecture. Very few are experts in all areas for all systems, and the best experts still make mistakes.
- *Nothing is 100% secure* Given enough time, resources, and incentive, any system can be compromised, and any encrypted information can be decrypted. A suitcase containing a million dollar bribe combined with a gun held to someone's forehead can be a convincing incentive. The main goal of a security administrator is to make the cost and risk of an attack too high to be useful.
- *The cost of increased security is not linear* There are diminishing returns to improving security. Doubling the budget may not double the security.
- *Principle of easiest penetration* The attacker will find any method of entry, and not the one with the strongest defense. The attacker has the advantage because they just need one successful attack, while the defender has to guard against all attacks.
- *Security is potentially only as strong as the weakest link* This is a similar to the above-noted rule and indicated why measuring security can be difficult. There may be hundreds of "weak links" using dozens of types of technologies. Experts in all of these technologies are needed to enumerate all of the weak links, and even then the list may not be complete.
- *The complexity of the system is proportional to the number of weaknesses* Simpler systems are easier to understand and therefore easier to secure.
- *Security through obscurity provides temporary protection, which degrades over time* Many believe security based on secret algorithms is sufficient. This is called "security through obscurity" and is commonly felt to *increase* security. However, this increase is temporary in nature and is not sufficient to *prevent* a security breach in the future. Given enough time and effort, any secret algorithm can be deduced and duplicated. The danger is complacency.
- *Security lapses occur when ownership is uncertain* Sites often divide responsibility between different organizations. They may cause each one to assume the other organization is addressing a problem.

Security Principles

Because of these issues, certain principles are commonly accepted.

- *Strong security is based on least privilege* Essentially this means that any action that needs privileges to perform its action should be given the least set of privileges it needs to operate. Give the process what it needs, not what it wants. And if one person needs a special feature, give it to that single person, and not everyone. The more generous privilege granting is, the more likely security will be breached. However, this increases the complexity of management.

- *The more layers of security, the stronger* Because weak points exists, additional layers increase security. An attacker must penetrate each layer, which is more difficult. This is also called *defense* in depth and it is an important principle in improving security.
- *The fewer security points, the better* By concentrating security entry points to a small number, security becomes easier. It is easier to maintain and observe fewer systems. Some refer to these systems as *choke points*. However, there are limits as this may conflict with defense in depth. If taken to the extreme, all security can be focused at a single point. If this fails, then all security is lost.
- *Security is everyone's responsibility* Security is not a task for a small group. If any single person creates a weakness, the entire system can be compromised. All users must be educated in security issues, and there must be a clear division of responsibility.
- *Peer review is critical* History has taught us that systems designed by security experts may still have a security flaw. The more resources devoted to reviewing a security mechanism without finding any flaws, the less likely security flaws will exist. Therefore consider peer review essential to adequate security. The more resources spent on peer review, the stronger the security.
- *If equipment fails, it should fail safe* Failures occur. If they do, it is better the failure does not cause a security breach. This is why many firewalls have a *default deny* stance: That which is not expressly permitted is prohibited. As an example, if a new application needs to bypass a firewall, the firewall should prevent this unless it is configured otherwise. If the firewall had a *default allow* stance, then the new application would be allowed unless the firewall explicitly blocked the application. The *default deny* stance is preferable because no action is needed to remain secure.
- *Identity is based on four proofs* Many computer systems require people to authenticate themselves to the computer. There are four general proofs that may be required: *Something you know, something you have, something you are, and someplace you are*. The strength of the identification depends on the ability of the computer system to test these four proofs. Typically, a password or personal identification number (*PIN*) is used as the first proof, or *something you know*. However, passwords may be learned; therefore this by itself may not be considered sufficient proof. *Something you have* is a physical object that only one person can possess at a time. Special devices can be used that provide tokens used to identify someone. This makes sure there is only one person using the token. Still, these devices, like passwords, can be stolen. An authentication that uses the first two proofs is called token-based authentication and sometimes called two-factor authentication system. The third proof is *something you are*. This describes biometrics, which uses sophisticated technology that identifies a characteristic of your physical body, such as voice analysis, retinal scanning, fingerprint or face recognition. It is reported that one site used fingerprint identification, until some criminals chopped off the thumbs of the CEO to gain access. The fourth proof or factor is *someplace you are*. This requires a mechanism for determining the location, perhaps using the Global Positioning System (*GPS*), or more simply making a decision based on the location or identification of the physical access mechanism. You may only be allowed to access a particular system from one special terminal, such as one in a shielded and locked room. As the number of authentication factors increase, the risk of miss-authenticating someone decreases.
- *Cryptographic solutions depend on keeping the key hidden, but nothing else* According to the generally accepted rules of cryptography (1), a secure solution is based on published algorithms the cryptographic *key* of which is secret. That is, given all of the source code, and the encrypted message, a system is secure if the time it takes to find the key (to decode the information) is longer than the time the information must remain hidden. Typically the strength of the algorithm is proportional to the size of the key used to encrypt the information, as well as the algorithm. The security of the system can be measured, because an estimate can be calculated on how long it would take to decode the information. Compare this approach to "security through obscurity," in which the strength disappears as soon as the secret is discovered, which is unpredictable.

- *Diversity increases security* Attackers often have limited experience and/or resources. Many have narrow fields of expertise. If an intruder has to compromise many different systems, it becomes harder to succeed. Therefore if different layers of filters and bastion hosts come from different vendors, security is increased. This is a temporary solution but practical at times. This also increases complexity, which tends to degrade security.
- *Security is inversely proportional to convenience* One common goal in a business environment is to increase productivity. However, this is often at odds with security. Many people would be unhappy if they had to lock up their terminal and office every time they left their office "for a second." In a network environment, people do not want to log into a computer system a dozen times a day. People often let others use their computer, yet the only way to trust a borrowed computer is to reinstall the operating system and all of the software you use yourself. A system administrator's job is particularly difficult if hundreds of computers are managed. Stronger authentication makes some tasks either more expensive or more costly with respect to time.
- *Security degrades over time* There are two reasons for this "entropy." The more a system is maintained, the more likely some human error will occur in the configuration and management of the system. Also the public knowledge and understanding of any system will increase over time, and new vulnerabilities may be found. To counter this, systems must be maintained and periodically reviewed by someone who understands all of the services the systems provide.
- *Security is a process, not a product* Because of entropy, security is often something that is addressed once and then forgotten. Security should be is an on-going process.
- *Complex systems are difficult to secure* The more complex a system is, the more likely there are security problems with that system. Try to simplify the critical system as much as possible. A server that provides six services is harder to secure than one that provides one service. And if one service provides a vulnerability, all six services could be affected. A more secure (although more expensive) solution is to use six separate servers, with each one doing one task.
- *Least common mechanism* Users should share as little as possible. The more a common program is used, the more impact a single flaw in that program can cause. Therefore security is improved if each user and service has the smallest number of privileges. However, this increases the cost of administration.
- *Security is hard to learn* Even among experts, information sharing is not free and unrestricted. Experts tend to be vague when discussing security problems to prevent hackers from learning too much. The amount of information that is needed to master security is very large, requiring expertise in every related field. This makes it difficult for a beginner to pick up, as the problems are based on internal details of operating systems, applications, and protocols.

Network Security Model

The network security model is a high-level description of the systems, networks, and users for a network. This includes a logical view of the functionality and structure. It discusses the roles, personal, locations, and organization of the system. It may also classify data and services and describe permissions. The model is used to clearly describe the security policy. Think of the model as defining the nouns to be used, while the policy describes the verbs that apply to the nouns.

Security Policy

The security policy specifies the goal of the security guidelines. It provides a way to specify the security requirements. Without written policies, there is no way to enforce security or measure the effectiveness of the

policy. It uses the terms defined in the model and places restrictions and assigns functionality to the elements of the model. It should provide enough information to evaluate the security of the system. The policy may also specify responsibilities of individuals and organizations.

Some of the questions that should be considered are below.

- Do guidelines and procedures exist?
- Do they have management support?
- Do they achieve consistent and complete security?
- Do they provide legal protection?
- Are roles and responsibilities clearly defined?
- Are the services and architecture understood and documented?
- Are the policies detailed enough to make sure they are enforced? Or are they from the 10,000 m level, and useless in the real world?
- Are the weaknesses understood?
- Are policies enforced and verified? How?
- Has everyone (not just the managers) agreed that they can follow the policies?

Policies should also cover management and change control, intellectual property rights, and personal privacy. Policies may also define acceptable use of the company resources, and define grounds for termination. A helpful book is *Information Security Policies Made Easy* (2).

Security Mechanisms

Mechanisms enforce policies and therefore are an implementation of a policy. However, policies are often an abstraction that describes the desired result. The policy "all users should be authenticated" is a policy, but the details may be unspecified to allow greater flexibility in the enforcement of the policy. The mechanism describes the *how* and *why*.

The security mechanism often provides an imperfect implementation of the policy. To give an example, a site's policy might state that unauthorized connections are not allowed. One mechanism may be to prevent this from happening using two-factor authentication. Another mechanism may choose to detect unusual and unauthorized connections and to terminate any found. Another mechanism is to simply "do nothing"—indicating that there is no mechanism in place to address a policy. Well-organized security organizations document the mechanism along with the policy. This helps identify the known weaknesses of a security policy.

Mechanisms used to enforce these services fall into the following categories.

Obfuscation. It is often possible to identify a particular vendor package using various probing techniques. The vendor, operating systems, and application set can be determined by examining announcement banners, login screens, active ports, and reactions to unusual network packets. Knowing the operating system can help determine future attacks. Hiding the characteristics of the system can make compromises difficult. Some recommend modifying all banners on all applications to mislead the attacker. This is security through obscurity, however, and therefore the use is limited. It will mislead or slow down an attacker but does not prevent attacks.

Prevention. Systems are often protected by the insertion of equipment or software that prevents violations. The commonest approach is to add a network firewall. A firewall will prevent attackers from connecting to unauthorized systems. An authentication system will prevent an unauthorized user from connecting to interior systems.

Detection. Detection is a passive protection mechanism. By watching log files, traffic patterns, file signatures, or any unusual activities, it can detect actions determined as suspicious. This activity is called an

intrusion detection system, or *IDS*. After detecting improper activity, a system may indicate a suspicious file or system, generate an alarm, or perhaps initiate a defense (integrating detection with reaction).

Reaction to attacks. When an intrusion is detected, a decision must be made concerning the next step to take. These steps often require planning, and sites are not always prepared when an attack/occurs. Some of the choices follow.

Gather more information. The site may decide to remain passive and gather more information. This usually requires a security team that has the ability to monitor activity and terminate it instantly. There are advantages to this approach. The team may learn the hacker's techniques, goals, and motivation. However, it requires skilled security administrators that are better trained than the hackers.

Communicate with the authorities. Tracing down the culprit may require cooperation with other authorities. However, finding the right authorities is not always easy and may be more difficult if the network is under attack. Sending e-mail to a site might only alert a hacker if the remote site has been compromised. Consider using non-networked methods of communication (phone) and having a contact list prepared ahead of time. Tracing connections may even require court orders.

Terminate the attack. The attack can be disconnected by closing a socket, terminating a process, unplugging the network connection, reconfiguring a firewall, or halting the computer. Disconnection a system from a network removes functionality the system provides to the network, alerts the attacker that the intrusion was detected, and inhibits tracing of the culprit. Restoring the system to a known state returns the service but may make forensics difficult.

Diversion. If an intrusion is detected, the system administrator may use diversion. This may be the creation of a false target (sometimes called a *honey pot*) that is more attractive to the intruder, thus diverting their attention. It might simply be a file with an attractive name or an elaborate simulation of a system that appears to be penetrated, while learning more about the attacker and the methods used. It may also provide additional time to trace the location of an intruder.

Forensics. After a penetration (actual or suspected), forensics may be necessary. This involves capturing and analyzing the state of the system. This may be dynamic (as the system is running) or static (analyzing the files on a disk). The reasons include understanding the motivation, identifying and studying hacking techniques, determining the degree of intrusion, and providing legal evidence.

Be aware that merely running a damaged system may destroy evidence. The best approach may be to halt the infected computer as quickly as possible and make a complete copy of the entire file systems, allowing you to safely analyze the data at your leisure. Foundstone (http://www.foundstone.com) makes a toolkit for Windows systems. Unix users maywish to look into The Coroner's Toolkit (*TCT*) written by Dan Farmer and Wietse Venema (http://www.fish.com/forensics/).

Legal response. Taking legal action requires planning and attention to detail. It also requires skills in forensics to find the individuals who were involved and prove they did specific damage. In some cases, systems have special banners displayed during the login sequence, which warn the visitor of the consequences of unauthorized access.

Recovery. After the attack, the next step is recovery. Since the integrity of the system cannot be trusted, it may be necessary to reload all of the software from the distribution tapes. Alternately, a file system integrity checker (e.g., Tripwire) can be used to identify the damaged files precisely.

Threat Model

A threat model is an attempt to characterize the capabilities, motives, goals, and tolerance of risk of the person or organization attempting to compromise a system. It may be an adolescent hacker, a disgruntled former employee, a foreign government, or a business competitor. The U.S. government has a different threat model than the PizzaDelivered To YourDoor.com Web Site. There is no way to prevent all potential risks. Instead,

the threat model helps prioritize the dangers by identifying those risks that are more likely to occur and the possible consequences.

Some models to be considered are as follows.

- Accident Hacker This is someone who stumbles onto confidential information. It may be someone who walks into your room and sees what is written on your screen. Some may report the situation when they discover the leak. Potential threat: disclosure of information.
- Merely curious person Some individuals are curious about particular tools or programs. They may execute a tool to see how it works or probe a site to see if a guest account exists. Potential threat: Disclosure of information.
- "Script-kiddy" This describes someone who has access to tools but does not understand how they work. However, this group is very large and given the right tool, can break into any site that is vulnerable to that particular tool. Targets are often picked randomly. The damage can still be significant because they may install a back door or allow others to abuse the system. Potential threat: Installing *rootkits* and back doors, telling others about the hole, allowing other hackers to misuse the system, using the system to stage attacks on other systems, and making forensics difficult.
- Experienced Hacker The experienced hacker often has a stronger motivation for the break-in attempt. They will try alternative break-in methods. The motivation could be a specific goal for political or financial reasons, or may be the use of the compromised system to stage attacks against other systems. Some consider failed break-in attempts as making the target more desirable. Potential threat: all of the above, loss of confidential information, loss of finances, public relations damage, loss of credibility in the market.
- Insider An insider may cause great damage to a company and is usually hard to detect. Confidential information may be leaked to competitors. Other motivations may be blackmail or revenge—both of which may cause damage to the infrastructure. The inside hacker may be coerced by outsiders into participating in the break-in. Insiders may also be embezzlers. Potential threat: all of the above, long-term loss of information and/or finances, loss of backups.
- Professional A professional hacker may be a security analyst being paid to break into a site to verify the security, or may be paid by another corporation or government. Potential threat: all of the above.
- Industrial Espionage agent The motivation for corporate espionage may be to steal secrets, or to sabotage the company through corrupted databases, public relation blunders, or simple feed misinformation. Potential threat: all of the above.
- Government worker Government-based hackers may have huge resources available and can cause great damage to a company. Potential threat: all of the above, damage to integrity of company, damage to government.

Probes, Attacks and Threats

Before the security of a system can be analyzed, the potential threats must be understood, as well as the mechanisms of attack. Threats are occurrences of something that can weaken or damage a computer system. Threats include disasters such as fire and loss of power as well as direct attacks. Other threats are subtler and do no obvious damage, yet provide additional information that can be used to plan an attack. These are called probes—operations that do not compromise the system, yet reveal information that can be used to plan and execute additional attacks in an efficient manner.

The attacker, or hacker (some prefer to use the term *cracker*, but hacker is common usage), often has a set of tools used to examine, probe, and compromise a system. The number of easily available tools is staggering. Understanding the commonly available tools and how they work help in determining the potential risk of a computer network.

Probes. Probes may or may not set off alarms. The difference between a probe and a curious individual may only be a matter of frequency and duration. Some probes gather information over a period of months or years to escape detection. Other probes may be coordinated with other large-scale attacks and thus be hidden in the noise. One of the most successful probing techniques is social information.

Social engineering requires that the attacker bluff an employee into revealing information or perform some operation that makes access easier. An attacker can pretend to be someone in authority, a customer, a vendor, a consultant, or a new hire. The attacker may also pretend to be a user who forgot his/her password, or assume the role of a system administrator, and ask a user to perform some action.

Successful penetrations have occurred when hackers have printed fake credentials and boldly claim to be performing a security audit. Other roles involve helpful customers asking secretaries and administration assistants for information because their boss needs help. Every employee must be alert for such actions and should notify a security representative if someone has a strange request. Contact points that are publicly known are particularly vulnerable. This includes public phone numbers, operators, the SOA account in the Domain Name Service, the WHOIS database, and common e-mail accounts like postmaster, hostmaster, news, etc.

An important part of any attack is gathering information. Programs such as *nmap* can determine the type and perhaps the revision of the operating system. This is called *OS fingerprinting*. Other examples of probe software are *finger*, *showmount*, *rpcinfo* or *nmbstat*, and *nmap*.

Social engineering is surprisingly easy, because all it takes is a winning personality. The attacker is rarely caught because the conversation should seem normal and physical access is not necessary. In many cases, social engineering can be the actual attack; if hackers can convince someone to grant privileges, they will gain access into the system.

Social engineering can be accomplished over the network as well. Innocent employees can receive faked electronic mail. Inside may be instructions to change passwords or install programs. This can be combined with web access, as e-mail discussing software updates can contain references to "update" programs that are really Trojan horses.

Another type of probe is called *trashing*. The attacker gains information by examining material in trash bins. Passwords, accounts, telephone numbers, and manuals can all provide useful clues and be combined with social engineering.

General Classifications of Attacks. Attacks may be classified by two major categories: *passive* and *active*. Passive attacks do not introduce any new data into the network and are therefore difficult to determine when they happen. The first common technique is *release of message contents*. Timing analysis may determine the length of a cryptographic key. The second form of passive attack is *traffic analysis* and uses knowledge of data size, frequency, source and destination to gain insight. This can be defeated by using circuitous routes with true source and destination hidden from sniffers, such as onion routers (discussed later).

Active attacks may be broken down into three categories: *interruption* (which affects availability), *modification* (which affects integrity), and *fabrication* (which affects authenticity). Modification can be broken down into four divisions: *masquerade* (pretending to be someone else), *replay* (repeating the transmission of data captured earlier), *modification of messages*, and *denial of service*. This later attack can be relatively simple. It can be accomplished by sending unexpected data or initiating thousands of connections, without terminating any of them. The latter can cause system resources dedicated to each connection to be consumed, preventing authorized individuals from using the system.

Mechanisms of attacks. Network attacks are possible because of flaws in a computer system. There are many reasons for these flaws.

Conceptual errors and carelessness. One of the most common problems is human error. As stated before, complex systems are harder to maintain. Computer systems tend to be very complex, which increases the chance of human error. The error may be caused by carelessness, lack of resources, or lack of understanding.

In some cases, people assume the problem is someone else's domain. Consequently, no one addresses the problem. One variation is the Trojan horse program, where someone is tricked into executing a program.

Faulty Software. The second most common problem is a security flaw in a computer program. If you can gain access to a program then you can exploit this flaw. Access may be facilitated by having an account on a computer system, or by connecting to a program that provides network services.

There are many ways a program can have a security flaw.

Stack overflow. Writing a secure program that properly separates data from instructions is difficult. This can be exploited by using a buffer-overflow-based exploit. In this case, a large amount of data is presented to the system, which is not programmed to handle so much data. Normally this will cause the computer to execute random code. However, a hacker can insert code with the data and trick the computer into executing unexpected code. This allows the attacker to execute arbitrary code, which can therefore grant extra privileges.

Any service on the Internet has probably been vulnerable to a stack overflow attack at some point in time. This is accomplished by finding a way to specify a parameter to a program, making it much larger than the programmer expected. An example might be specifying a username 2048 bytes long, instead of 8. If the program does not explicitly check for this error, the excess data may be thrown onto the stack, and a smart programmer can use this to trick the system into executing arbitrary code. There exist many stack overflow exploit kits that make it easy to overflow some parameter and execute random code. If the process is privileged, the code can execute commands that weaken the security of the system.

Most break-ins occur because a privileged server is vulnerable to a stack-overflow attack, the exploit is available, and the program has not been patched. Exploits are commonly published on the *Bugtraq* mailing list, so anyone can gain access using the exploit.

Heap Overflow. The heap is used to store data in memory explicitly requested by the application. If the programmer is careless, someone can send unanticipated data that modify critical locations in the heap.

Trojan Horse. If someone has access to a system, they can run a program that asks for passwords. This might be a web page or a login screen. It may be a misspelling of a common word. If the user is an administrator, it allows someone to weaken the security of a system. Some people put the Current directory in their searchpath or a directory that might allow someone besides the superuser to replace a file. It is therefore possible to get a privileged account to execute the wrong file. The author's *trojan.pl* program can check if a Unix account is vulnerable to Trojan horses.

Password capturing and cracking. There exist several programs that can take encrypted password files (from Unix NT systems), and by using a dictionary attack, decode the passwords. Most Unix systems use shadow passwords, but if they use NIS, it is often trivial to get this encrypted password file (by using *ypcat.*). Windows NT systems have potentially similar weakness. There is a program called *pwdump.exe* that can sometimes obtain a password file. This requires loose permissions on the account. However, if someone has created a startup disk, there exists another copy of the password file. The program *l0phtcrack* combines a dictionary search with a password sniffer to learn passwords.

Race Condition. Some systems allow set user identification (*UID*) shell scripts. Some pass the script to the shell using a file descriptor (e.g., Solaris 5.x). Others pass a filename that is insecure. It is possible to change the script after the shell has started yet before it has read the script. If so, this race condition can allow someone to trick a privileged account to access the wrong file.

Temporary Files. If a user has access to a system and the system opens a temporary file, it is often possible to trick the program into opening the wrong file. It may be the encrypted password file, and this can reveal the contents of this file. Some temporary files are made world writable, and it is possible to create a symbolic link to point to a critical file. This problem can be exploited if the OS opens a symbolic link, which can point to another symbolic link. This can be nested, and by the time the system opens the final file, it may have changed. Programs must be careful so that the file they finally open is the same as the one they think they have opened. This is done by using system calls to check the status of the file before and after opening the file. Some systems create temporary files that exist inside a subdirectory that is writable by the user and only

the user. Instead of opening the file /tmp/123 the system would open the file /tmp/username/123. This is not an easy problem to fix, and some systems do not have libraries to open a temporary file securely.

Chroot. Some accounts are established in a chroot() environment. If the user does escape the server, then accounts are found inside a subset of the entire filesystem. There exist techniques that can break out of such an environment, and this is easier if the process is privileged. One trivial technique involves repeated cd.. commands followed by chroot/ commands.

Keystroke Watchers. Programs exist that can capture all keystrokes on a computer and can therefore capture passwords.

Cryptographic attacks. Cryptographic systems have many weaknesses that can cause a system to be compromised.

Weak cryptographic algorithms. The first problem is caused by the choice of the cryptographic algorithm. It takes decades of study to verify the strength of a cryptographic algorithm. Even the best cryptographers may design a weak cryptosystem, and the flaws may not be discovered for 30 years. Other algorithms can weaken over time, as computers become more powerful. The Data Encryption Standard (*DES*) has been broken. That is, an encrypted message was decrypted using massive computer systems. A dedicated engine can be built that can crack DES-encrypted message. Stronger systems require more time. A message that takes 1000 years to decrypt is considered secure for the short term. The only provable secure cryptosystem is the One Time Pad, where the length of the key is equal to the length of the message. However, distributing the key becomes the problem.

Key discovery. As stated before, the strength of a cryptographic system should rely on keeping the key secret and not on keeping the algorithm secret. Some systems establish a secure connection using a secret key. In some cases, the key is generated randomly. If the hacker can guess the key, they can pretend to be one of the two parties making a secure connection. This can cause a problem if the number is not randomly generated. In most cases, computers use pseudo-random-number generators (*PRNG*) instead of truly random numbers. These generators use a seed. As an example, some programmers do the equivalent of the following C code to initialize the seed of a PRNG:

```
(srand (gettimeofday ()))
```

Someone who knows when the PRNG was seeded has the ability to guess the actual seed. Because the algorithm used to seed a pseudo-random-generator is known, the seed may be known, and the key exploration can be reduced greatly, therefore reducing the advantage of a long key length.

Sometimes the length of time of a process can be used to determine the length of a key, which again shortens the search. In other systems, the length of a short message might determine the key length.

Known Plaintext attack. If the attacker knows the plaintext before encryption and can examine the results afterwards, this can provide clues to the key

Chosen Plaintext Attack. This is similar to the known plaintext attack, but the attacker gains the ability to provide plaintext to the system. Therefore the hacker can provide patterns that reduce the search space.

Password guessing. The most primitive technique is password guessing, using a brute force dictionary attack. The hacker, using a dictionary, guesses a username and/or password. This may be a local account or on a remote server. This will probably take a long time to find the right combination, but eventually the right combination may be found. Secure systems have mechanism to detect and/or prevent brute force attacks by terminating the connection, disabling the account, or slowing down the response.

Faulty protocols. A third common problem is using insecure protocols. All of the protocols created in the early days of the Internet were not secure. This includes transport protocols (TCP, IP, UDP, ICMP) and application protocols (MAIL, File Transfer, Name service). Newer protocols are being created that are secure,

but not every system uses them. There are several general techniques that can be used to attack faulty algorithms.

Confidentiality (release of message content) errors. Passing sensitive data in cleartext allows a release of message content. If the information is in cleartext (i.e., not encrypted) the contents can be learned by a program called a "sniffer". A sniffer program passively watches network traffic without disrupting it. The information released may be partial. Standard telnet and file transfer protocol (*FTP*) are vulnerable to this, as passwords are sent in the clear. Some common sniffers are *tcpdump*, *snoop*, *dsnoop*, *sniffit*, and *juggernaut*. Others include programs that display information of applications, like the X Windows server.

Masquerade (Authentication) errors.

Man-in-the-middle attack. Normally, two systems A and B connect to each other. In a man-in-the-middle attack, a third system interposes itself between the two without their knowledge. The attacker pretends to be system A to B, and pretends to be system B to A. In doing this, one of the systems can be tricked into revealing privileged information.

Host-based authentication. Some systems are protected by using filters based on IP addresses or host addresses. The authentication mechanism is flawed, that is, only specific IP addresses are allowed to connect to a system. The IP address provides the authentication needed, as it is assumed to come from a trusted source. The Berkeley r commands (rlogin, rsh, rcp) make this assumption. However, the IP address can be changed. A hacker may disable or disconnect a trusted system and change the address on a second computer under the attacker's control. The hacker may also send out packets with faked source addresses and use a sniffer to receive any responses to the system.

Another example of this flaw is any system where the remote system gives information to authenticate a user and the server must trust the remote host. The *rexec* command has this problem. Standard NFS also has this problem, because the remote client tells the server what the user's identity is. This can be faked, allowing someone to get access to another person's account.

Media-access-control layer spoofing. Some sites use a switching hub that prevents a network sniffer from seeing packets addressed to other addresses. However, systems can create forged media-access-control (*MAC*) addresses, such as Ethernet addresses. This fools the hub into thinking the systems with the spoofed MAC addresses are located on another port. Thus the hub is fooled into forwarding packets to the wrong machine. Pavel Krauz's program *hunt* demonstrates this concept.

Modification errors.

Interjecting false credentials. Some protocols do not detect or prevent intruders from interjecting false information. A query is sent to a server, which responds. However, someone else can interject a response with false information; therefore providing false information to the client, which may then trust another system Falsely.

Another example is router protocols such as RIP, which may not authenticate information. A properly formed packet from an attacker may convince a router that it is the proper authority, and to ignore any more responses from the same address. The results may be to forward all traffic for a network to the hacker's network.

Sometimes attacks can be combinations. One example of this is *ypsnarf*, which listens (using a sniffer) to login requests from systems using YP/NIS protocol, and when seen, can return a packet that says the account has the UID of zero (Superuser) and no password. The client uses this information to grant the person superuser access, and the attacker now has control of the client machine.

Hijacking (Nonblind spoofing). If the packets can be sniffed, and new packets created, then hijacking of any IP-based connection is possible. The higher-level protocols have to provide authentication, and most of

them do not. There exist tools (*Sniffit*, *Juggernaut*) that observe IP-based connections, and when desired issue a two-pronged attack. The first attack is to send forged packets to one side that causes the connection to be reset. The second attack is sent to the other side with forged headers and proper IP sequence numbers that steals the connection. Any TCP-based service is vulnerable to this attack. This includes sessions with one-time passwords, like SecurID.

Blindspoofing. This is a variation of the attack just discussed. Blind spoofing does not require a sniffer. However, it does require a predictable IP sequence number, a vulnerable protocol, and the ability to forge IP addresses. A system that is configured to allow *rlogin* access from a single site that has predictable IP sequence numbers is vulnerable to blind spoofing. Packets with forged IP addresses are sent to the server, which responds to the trusted remote site. The hacker may never sees these packets, as it does not control the remote trusted system. However, if the IP sequence number is predictable, the hacker can send the next packet with the proper IP sequence number of the server, and the server will accept the packet.

The Berkeley r commands are vulnerable to masquerade errors, because they depend on host-based authentication. It is said that Kevin Mitnick used these flaws to insert the commands

$$\text{echo} + + > \sim /.\text{rhosts}$$

into an IP connection without a sniffer (3).

Interruption attacks. Several attacks can interrupt an existing service. In general, these are called *denial of service* attacks.

Denial of Service. There exist dozens of denial of service attacks. Some that generate large quantities of electronic mail are called *Unabomber*, *KaBoom*, *Up Yours*, *Avalance* and *Voodoo*. Others use mail-formed packets or open hundreds of simultaneous connections without releasing resources. (e.g., SYN flood). Some of the programs are *bonk*, *jolt*, *land*, *nestea*, *targa*, *netear*, *syndrop*, *teardrop*, *winnuke*, *ptebug*, *eudora4*, *portmap*, *spiffit*, *rwhokill*, *overdrop*, *slmail26*, *aixttdbserver*, *biffit*, *pnbug*, *netscaetbl*, *antisentry*, and *hugweb*. If these programs are executed with your system as a target, your system will either crash or slow down to a crawl. Some of them (*smurf*) have a multiplying effect, and a single packet from one system can cause a hundred packets to be echoed to your server through innocent third parties. In 1998, a package called Xcrush was released that provided a menu-driven interface to 40 different denial-of-service attacks.

Distributed Denial of Service. In 1999, hackers developed a distributed denial of service (*DDOS*) attack called trin00 and TFN. In simple terms, several innocent sites were hacked, and special software was installed on them. These systems were called zombies, because they did not know what they were being asked to do. On command, these zombies would attack other systems, using faked IP addresses. It is very difficult to trace a packet coming from a faked source IP address, and tracking down hundreds of these systems made the task much more difficult. Other DDOS programs are TFN2K and Stacheldraht.

Traffic analysis. Even encrypted traffic can be vulnerable to traffic analysis attacks. Onion routing is a conceptual model that provides protection from this as well as from conventional sniffing. Instead of talking to a server directly, an application connects to a proxy on an onion router. This encrypts the data and sends then to another nearby onion router. Encrypted packets may contain encrypted packets inside. Each router decrypted the data addressed to itself, and eventually the packet exits somewhat randomly from the onion router network and connects to the target server. See http://www.onion-router.net/ for more information.

Miscellaneous attacks. Other attacks include physical access and modern access. There exists special equipment that can examine magnetic media and read information from disk after the data have been overwritten with ones and zeros.

After the attack. The question arises: What if someone breaks into a system? What happens next?

Log Scrubbers. The first step of a hacker is to hide any trace that they have been on the system. There exist programs that delete entries in the log, so that the system administrator cannot detect that the system has been abused. This is often done immediately after breaking in.

Rootkit. A more sophisticated technique is to install what is called a *rootkit*. These kits are readily available, and can be installed in moments. This is a set of replacement programs that have some or all of the following features:

- The programs look like the original programs.
- The programs have the same checksum as the original programs.
- The programs have the same dates as the original programs.
- The programs have been modified to hide the hacker's activity. If you try to spot hackers on your system, you cannot see them. This is because the rootkit replaces standard programs like *ps*, *netstat*, and *ls*. If the hacker is running a sniffer program, for instance, programs that list running programs will not show the sniffer is running. If the user lists filenames, particular files remain hidden.
- The programs often have a back door in the network services, so that any user whose name is something like "XYZ" will be allowed onto the system without a password and will be granted unlimited privileges. Or they may reinstall back doors that have been removed by a system administrator. This is why it is recommended that the entire operating system be reimaged. It is difficult to detect the existence of a program when you can not trust the programs you use. Tripwire provides some help in this problem, but you must set this up beforehand.

Tunnels. Hackers can install tunnels that allow communication to or through a firewall, without raising alarms. The data stream may be on an innocuous port number, or may be inside UDP datagrams. Some create tunnels to carry information inside other protocols like ICMP (ping).

Sniffers. Hackers use new accounts to break into other systems. One way they do this is to run a sniffer on a compromised machine. They transfer their attack programs to a new foothold and continue the search. One common step is to install a password-sniffing program, and the hacker can penetrate deeper and deeper into the network.

Zombie agents. As mentioned in the discussion on the distributed denial of service, software may be installed that allows hackers to attack other systems from the compromised machine. Others are general purpose software and allow the hacker to take over the system. One example is Back Orifice, for Windows-based systems.

Risk Analysis

Types of risk analysis. Risk analysis is a mechanism to identify, measure, control, and minimize risk.

Vulnerability Analysis. A vulnerability scanner is easy to operate and use, as it is usually a series of tests that are executed in sequence. There are several scanners that can be used to analyze potential risks. Two general categories are host-based scanning and network-based scanning. Host-based scanning is cooperative and runs off the system being examined. Examples include COPS and TIGER.

Network-based scanners examine remote systems and may be done without cooperation (and therefore suitable for use by hackers). The first freely available network-based vulnerability scanner released to the public was called SATAN.

The following is a list of some of the commercial vulnerability analysis tools at the time of writing this document. Some of the available host-based vulnerability Analysis tools are as follows:

- COPS 1.04—Dan Farmer
- Security Manager—NetIQ
- VigilEnt Security Agents—PentaSafe
- Norman Virus Control—Norman Data Defense Systems
- Database Scanner—Internet Security Systems
- System Scanner—Internet Security Systems
- SFProtect Enterprise Edition—Agilent Technologies
- Security Analysts—Intrusion.com
- bv-control for Windows—Bindview
- Security Configuration Manager—Microsoft

Some network-based vulnerability analysis packages are as follows:

- SATAN—Wietsa Venema and Dan Farmer
- SAINT—WorldWide Digital Solutions
- NetRecon—AXENT Technologies
- bv-control for Internet Security—BindView
- NetSonar—Cisco
- Internet Scanner—Internet Security Systems
- Nessus—Renaud Deraison and Jordan Hrycaj
- NMAP—Fyodor

Network-based scanners have to estimate if a system is vulnerable, because typically they do not actually attempt to break into a system, but they just look for the potential that a vulnerability exists. Vulnerability analysis gives a list of potential vulnerabilities and indicates some value (such has high, medium, or low) of the priorities. Commercial scanners may have thousands of potential vulnerabilities in their database, and the measurement is a list of well-known vulnerabilities with potential risks. However, it is difficult to determine priorities accurately and evaluate consequences of vulnerabilities.

Scenario Analysis. Scenario analysis is a fairly easy mechanism to estimate risk. Several potential scenarios are considered, and the likelihood of the risk is estimated. No real data are used in this analysis system.

Application Risk Analysis. Application risk analysis is used to example one or more applications. The potential risks are identified, and the potential damage or consequence of that risk is considered. This is helpful in determining priorities. It is useful for application developers but does not provide a complete view of a system.

Qualitative Approach. The qualitative model is an attempt to calculate the overall risk of a system. It uses simple formulas to estimate risk. It considers eventual risks, vulnerabilities, attacks, vectors (systems used to launch attacks), and controls. Controls may include deterrent, preventive, corrective, and detective controls.

The numbers are subjective and have approximate values, such as high, medium, or low of 1 through 10. The scales may be logarithmic, where a cost of $10 is 1, while $10 million is 9. A comprehensive list of vulnerabilities is used, with classifications described earlier. The formula may be based on a calculation such

as

$$\text{Total risk} = \sum_{n=1}^{Number\ of\ vulnerabilities} [(Likelihood) \times (cost)]$$

The cost may include such factors as

- Financial loss
- Cost of disruption
- Extent of legal liability
- Breach of confidentiality
- Extent of embarrassment

Other calculations may include a control factor, indicating the means to reduce the risk of vulnerability. Qualitative analysis is simple to calculate, but produces numbers that lack concrete values. A total score of 1500 has little meaning by itself. It can be useful in determining priorities or evaluating security over time.

Quantitative Approach. Another approach for calculating risk is to use the quantitative analysis approach. An example of this is the annual loss expectancy (*ALE*). Other formulas can be used. The advantage of this approach is the calculation of potential lost and return on investment. There are several steps.

Step (1): Identify assets. Identify all items of value. This includes items such as

- Hardware
- Services
- Software
- Intellectual property
- Reputation
- Customer base

Step (2): Determine the value of each asset. Calculate the value of each asset. This is typically a logarithmic scale ($1000, $10,000, $100,000, etc.). There may be different categories of loss, such as availability, integrity, and confidentiality. Each category must be considered. Next, the total value of each asset is calculated, summing up the loss in each category.

$$\begin{aligned}\text{Value} &= (\text{impact of loss of availability}) + (\text{impact of loss of integrity}) \\ &+ (\text{impact of loss of confidentiality})\end{aligned}$$

Step (3): Determine, the likelihood of vulnerability. Next, a calculation is used to determine the likeliness of losing that asset. The scale is often logarithmic. Table 1 is an example.

Step (4): Calculate the potential lost (cost) each year. For each asset, a calculation is made of the potential cost of the threat. Consider a server containing a customer database. If the server was disabled for a week, it would cost $100,000 (availability). The database may take months to build, and would cost $2,000,000 (integrity), and the confidentiality would cost $5,000,000. The total potential loss is $7,100,000. If the threat is a hacker using a stack overflow of an out-of-date program to gain privileges with the occurrence of once every 5 years, the loss would be $7,100,000 $\times$ 1/5 or $1,420,000 a year for that single threat. The annual lost

Table 1: Calculation of the Likelihood of Vulnerability

Likelihood	Occurrences per year	Probability per year
Never		0.0
Once in 300 year	1/300	0.00333
Once in 200 year	1/200	0.005
Once in 100 year	1/100	0.01
Once in 50 year	1/50	0.02
Once in 25 year	1/25	0.04
Once in 5 year	1/5	0.2
Once in 2 year	1/2	0.5
Yearly	1	1.0
Twice a year	2/1	2.0
Once a month	12/1	12.0
Once a week	52/1	52.0
Once a day	365/1	365

expectancy is a summation of each of the threats.

$$\text{ALE} = \sum_{n=1}^{\textit{Number of assets}} [(\text{asset value}) \times (\text{threat occurrence})]$$

Total the ALE for each service or component to get the total ALE.

Step (5): Survey applicable controls. Determine what mechanisms exist to reduce or eliminate the threat.

Step (6): Determine which threat produces the biggest ALE. In this example, consider the hacker breaking into the database to be the biggest danger. A potential control is to add an intrusion detection system. Assume this will eliminate 85% of the attacks. This specifies the effectiveness factor (0.085).

Step (7): Calculate return on investment. If an intrusion protection system costs $10,000 per year, with an effectiveness factor of 0.085, the return on investment (*ROI*) can be calculated as

$$\text{ROI} = \Big(\textit{factor}\Big) \times \Big(\text{ALE}\Big) \Big/ \Big(\text{annual cost}\Big)$$

In the example above, the value would be (0.085 × 1,420,000)/10,000; therefore the return on investment would be 12:1.

Advantages and disadvantages of the Quantitative Analysis Approach. There are several advantages of this approach. It can calculate dollars lost per year, in terms easier to understand. However, the process is more complicated, and more expensive in time and resources. Changes in the system may have a major change in the calculation. The calculations do not require expertise in security. However, many assumptions are made, especially concerning the likelihood of an attack. It is difficult to get real numbers of attack frequencies. It is also difficult to estimate value of assets. Some threats have high frequency and low impact (operator error)

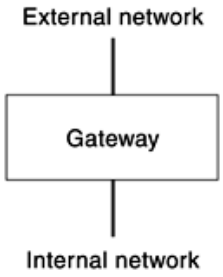

Fig. 1. This network gateway has no filter and provides no protection against attacks.

while others have low frequency and high impact (earthquake). Both may have same ALE per threat. The quantitative model cannot distinguish between the two,

Firewall Architecture

As the Internet evolves, understanding of network security evolves. Early on, systems connected to the Internet were *bastion hosts*—hardened against attacks. Systems providing connectivity often had two network addresses, one on the outside and one on the inside of the network. These systems were called *gateways*. However, these systems offered poor defense. If they were compromised, the entire network would be vulnerable to attacks. Instead, the network connection evolved into firewalls and demilitarized zones (DMZ). This adds a "defense in depth" of two or more layers.

Several components comprise the total system. See the book *Building Internet Firewalls* (4) for more information.

This section provides is a list of components that might comprise a firewall system.

Internal network. The internal network consists of all of the systems that need protection yet still allows people to productively use the resources. One approach is to make every system visible to the Internet. However, this requires every system to be secure, and according to the *principle of easiest penetration*, only one system need be compromised.

Bastion Host. A bastion host is a system that has been hardened against attack. It is stronger than normal defenses and protective devices. Generally bastion hosts are the systems that are typically most visible, usually attacked first, and will probably be the first to be compromised. In most cases, they are general-purpose systems that have been hardened using software and configuration options to protect the system from attack. A network without a firewall would require every system to be a bastion host.

The phrase *bastion host* was popularized by Marcus Ranum (4).

Demilitarized Zone. A demilitarized zone is a term for a network that is exposed to attack. Rather than protect every system, it is easier to protect a few and place them in a DMZ. However, this does not in itself protect the internal network. A DMZ is sometimes called a perimeter network.

Filters and firewalls. Filters are hardware devices that isolate one network from another. Each device uses particular rules to decide if data transfer should be allowed or not. Figure 1 shows a network gateway. This has no filter. The early firewalls had two filters and a DMZ, and looked like Fig. 2. The filters contained rules that only allowed the external network to connect to the bastion host and did not allow the bastion host to connect to the internal network. The internal network could connect to the bastion host, but the bastion host was not allowed to initiate a connection. This rule is in place to protect the internal network in case the bastion host becomes compromised. Rather than use two devices, modern firewalls often look like Fig. 3.

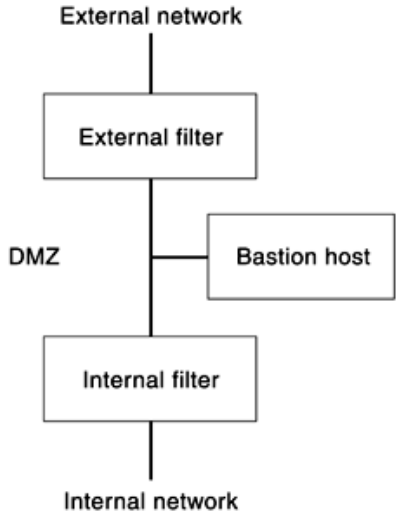

Fig. 2. This early firewall only allowed the external network to connect to the bastion host. The internal network could initiate a connection to the bastion host.

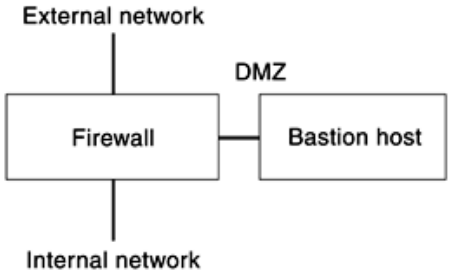

Fig. 3. This modern firewall combines the external and internal filters of Fig. 2.

No filter is perfect, and each has unique strengths and weaknesses. As vendors gain experience, the intelligence of the filters increases, as does the capabilities. Filters perform one of three actions:

(1) They can allow data to pass through them without modification.
(2) They can modify data that travel through them.
(3) They prevent data from traveling through them.

Filters that block or modify data for security reasons are called firewalls. There are two ways to classify a firewall: Layer of operation and layer of filtering. By layer, I am referring to the ISO Protocol stack. On a TCP/IP network the four layers correspond to the network layer, the IP layer, the transport layer, and the application layer.

The layer of operation specifies how the filter behaves to an outside observer. It is viewed from the outside. The layer of filtering determines what is blocked, and what is not. Each firewall may operate on one layer, but provide filtering at several layers.

Network Layer-2 Firewall. This filter operates at the data-link level or layer 2 in the ISO reference model. It is placed between IP devices and is visible at the MAC layer. On an Ethernet, the Ethernet address

(e.g., 8:0:20:11:ac:85) would be different, but substituting a firewall for a bridge is not noticed by the host. It operates to their perspective as another bridge. These devices are sometimes called secure or intelligent hubs or bridges. Some bridges perform encryption, and can be used to construct virtual private networks (*VPN*).

A variation of this is a smart hub that performs filtering. For instance, it may only allow packets to travel to a system if the destination matches the system. This prevents a network sniffer installed on one system from seeing packets addressed to other systems. However, a sniffer can still see packets addressed going to and from its own connection. A switching hub does this automatically, as long as it is safe from modification. Some firewalls can modify their MAC address and be completely invisible to the hosts.

Network Layer-3 Firewall. The layer-3 filter operates at the IP network layer. That is, it acts as a router that may block or modify certain packets. A common name for a layer-3 filter is a packet filter, or a screening router, as it always performs layer-3 filtering. It may, however, perform layer-4 or -5 filtering as well.

Packet filters are configured to prevent certain packet configurations from passing through. These devices make decisions based on IP addresses, IP ports, and header information such as flags and options. While earlier layer-3 filters acted on information in the headers, later versions performed stateful inspection of the data, essentially examining the data at layer 4 and potentially layer 5,

Some layer-3 firewalls advertise routes through them, that is, servers are unaware of the filter. Such a device will allow a system inside the firewall to make connections to the Internet, with no modifications to the protocols. They may also perform network address translation (*NAT*) to prevent internal addresses from being seen on an external network. They do this by translating internal layer-3 addresses to a set of corresponding external IP addresses.

Screening routers can be configured as an internal router or external router. They can block TCP sessions because they understand the flags in a header that specify the start of a new connection. Therefore they can be configured to allow a system on one side to connect to a system on the other side and prevent the reverse. However, if an attacker changes the normal order, or sends packets out of order, a layer-3 filter will not detect this.

Two publicly available packages allow you to turn a PC into a layer-3 IP filter. One is called Drawbridge, from Texas A&M University. The other is called KarlBridge, by Doug Karl. The same PC can be conveted to using the Linux operating system, which comes with filtering software. Another is called. *ipfwadm*, and it is part of the Linux software.

Layer-4 Firewalls. The layer-4 firewall operates on the transport layer, either at the TCP or UDP layer. Another name is a application-layer filter, application-layer gateway, or a proxy firewall. The firewall understands the status of a connection. It retains the history of a connection and understands when packets arrive out of order.

Layer-4 firewalls always perform NAT, as the devices on either side do not talk to each other directly. Instead, they talk through a proxy, which is why these firewalls are called proxy servers.

Proxy servers operate at layer 4. Three popular proxy packages are SOCKS, *screend*, and the TIS Firewall toolkit. The proxies used by web browsers are also layer-4 firewalls.

The filtering the proxy firewall performs may or may not be aware of the data corresponding to the data. TCP connections maintain state, and particular applications associated with well-known ports have well-defined data formats.

UDP packets are difficult to filter, because they have no state. Because of this, it is difficult to determine when a connection is finished and how it starts. Many sites block UDP packets because of this problem.

Difference between packet filtering and proxy firewalls. Early packet filters did little stateful inspection. As they grew in sophistication, they added more and more "state" to the rules. One of the potential problems with a layer-3 filter is that the filter must perform the same operation on packets as the device it is protecting. If it receives malformed packets or fragments of IP packets, it must reassemble and/or reject them with the same logic as the devices it is designed to protect. Fragments may contain overlapping data segments, and segments may arrive in any arbitrary order. However, there are subtle differences between

different operating systems and how they respond to strange and unusual packets. Since a single firewall may have to protect multiple operating systems, it is difficult for it to exactly match the characteristics of each and every operating system. It is possible that a packet filter will try to determine the exact state of the system being protected and be incorrect. This allows attacks to bypass the firewall. Firewalls that operate using the IP stack of the native operating system may be vulnerable to attacks that fool the filter, but penetrate to the system under protection. Proxy firewalls allow applications to be built and installed on a system in a DMZ, which intercepts and examines all data at the application layer. The proxies are typically on a second system on the DMZ. If the proxy is compromised, the internal network is protected. Stateful filters cannot handle arbitrary applications, and since a single system is used, if it is compromised, the internal network may be compromised.

Some layer-3 and layer-4 firewalls can perform layer-5 filtering. The system examines the data specific to the application and passes or blocks connections/packets based on this information. A mail server that blocks unwanted e-mail (SPAM), or access to any image file can be considered a layer-5 filter. FTP servers that block access based on names of files are also layer-5 filtes. Typically the rules for these filters differ depending on the application, so a mail filter operates differently than a FTP filter. It is easier for an application-specific proxy to have advanced rules and to make a general-purpose packet filter understand the rules.

Layer-4 firewalls can be combined with layer-3 firewalls, with the proxy server in the perimeter network, protected by a layer-3 firewall.

Intrusion Detection system. An intrusion detection system (*IDS*) is a system to give you an immediate alert if someone is actively trying to break into your network. In some cases, the detection system is also a prevention system, as it will react to attacks and block or deny the attacks. There are two kinds—network based and host based. Host-based intrusion detection systems look for tampering and attacks on a single host. Network-based IDSs use either a sniffer (to examine all packets) or else examine log files from multiple machines.

Tamper Detection. Eugene Spafford and Gene Kim developed *tripwire* at the COAST Laboratory at Purdue University in 1993. It is used to detect if any files have been modified on a system, and therefore detect a rootkit or other back doors that have been installed. Free and commercial versions are available. It usually runs on a bastion host.

Port Scanner detection. A port scan is a probe that examines the system for potential vulnerabilitites, but does not actually attempt to break into a system. Someone running a port scanner may merely be curious, while others may be planning a break-in attempt. Some people log, then ignore these attempts, and others actively contact the source of the attack to investigate the cause. Some port scanners operate at low frequencies, or with unusual options, and therefore attempt to avoid detection. These are called stealth scans. There exist several packages to detect port scanning. However, many of these cannot detect stealth scanning. (Some commercial scanners do.) Some of the public domain ones include *Klaxon*, *Courtney*, and *scan-detector*.

Other IDS packages. Intrusion detection systems fall into many categories. The detection may be host based, multihost based, or network based. The detection may be based on misuse, by detecting signatures of known attacks, or by detecting anomalies (variations from proper patterns).

Some of the first detected scans from particular packages, like *Courtney* and *Klaxon*, which detects scans made by the SATAN package. However, these did not detect scans using stealth techniques. Scanners were developed that watched for these actions, but this was still insufficient. Researchers worked on new solutions.

Some of these solutions were DIDS (University of California at Davis), EMERALD, IDES/NIDES (SRI), Haystack (which became Stalker and WebStalker), NADIR (Los Alamos National Laboratories), NetRanger (WheelGroup), OmniGuard (AXENT), RealSecure (ISS), and SWATCH—a public domain package by Todd Atkins of Standford University.

Researchers have recently developed more sophisticated intrusion detection systems. The COAST Laboratory at purdue University developed IDIOT (intrusion detection in our time) The University of California

at Davis has developed GRiDs (graph-based intrusion detection system). In 1998, COAST announced AAFID2 (autonomous agents for intrusion detection).

In 1998, the Naval Surface Warfare Center, Dahlgren Division, has developed the Cooperative Intrusion Detection Evaluation and Response project and made the software available (http://www.nswc.navy.mil/ISSEC/CID)

Some of the available host-based IDS packages include the following.

- Intruder Alert—AXENT Technologies
- Centrax—CyberSafe
- CyberWallPLUS—Network-1 Security Solutions
- Entercept—Entercept Security Technologies
- PentaSafe VigilEnt Security Agents—PentaSafe
- Tripwire Software—Tripwire
- Harvester—farm9.com
- NFR Intrusion Detection Appliance—Network Flight Recorder (NFR)
- Security Manager—NetIQ
- ManTrap—Recourse Technologies
- RealSecure—Internet Security Systems

Some available network-based IDS packages include the following.

(1) NetProwler—AXENT Technologies
(2) Dragon IDS—Network Security Wizards
(3) Shadow—the SANS Institute
(4) Network Flight Recorder—Network Flight Recorder (NFR)
(5) Anzen Flight Jacket for NFR—Anzen Computing
(6) Open View Node Security—Hewlett Packard
(7) Cisco Secure Intrusion Detection System—Cisco Systems
(8) Centrax—CyberSafe
(9) ManHunt—Recourse Technologies
(10) Harvester—farm9.com
(11) NetProwler—AXENT Technologies
(12) RealSecure—Internet Security Systems
(13) RealSecure for Nokia—Nokia
(14) SecureNet Pro—Intrusion.com

Authentication devices. Authentication devices provide additional proofs of identity. Typically a physical token is used. This may be an electronic device or something as simple as a piece of paper. One paper-based system is the S/KEY identification system, which is a list of passwords on a piece of paper. Each password is used once, and once used is crossed off the list. Other authentication systems use special token devices such as SecureNet or SecurID that generate a number that is cryptographically difficult to guess, yet must be used to access the system. Other authentication system used biometric data.

Virtual Private Network. A virtual private network (VPN) provides a secure mechanism for someone on a public network to have access to the interior network. This is done by using strong authentication and secure

protocols. In some cases, a secure tunnel is made between the outside user's system and the interior network, and all protocols are sent through this secure tunnel.

Honey pot. A honey pot is a decoy—a system that seems to be easy to penetrate and provides interesting information. But in reality, the information is false, and the main purpose is to delay and confuse the attackers.

Trends in firewalls. Firewalls can be configured in multiple layers, requiring a hacker to penetrate all of the firewalls to get to key data. Firewalls are also being inserted between workgroups, providing extra layers. Network interface cards are available that perform both encryption and filtering and can be combined with strong authentication and policy management. VPN systems are being combined with firewalls and policy management systems, allowing entire networks to have strong control of access privileges, while protecting individuals accessing their home network while using the Internet.

Summary to Network Security

This document discusses the components of a secure system, and suggests steps to take to address the issue. A written policy is the first step, as it defines the security you wish to enforce. Understand the threat model and learn about current threats and hacker techniques. Use risk analysis to determine and measure potential vulnerabilities. Increase the authentications and access control. Consider physical security and training people in secure procedures. Protect critical resources, and constantly review potential weaknesses. Follow the suggestions earlier, and increase auditing of systems. Replace insecure protocols with secure protocols.

History of Network Security Incidents

In the 1960s, *hacker* originally meant *clever programming*. Nowadays, it commonly refers to someone trying to break into computer systems. Some people have suggested the term *cracker* is more accurate, to better distinguish the evolutionary change in meaning.

Network security evolved from issues dealing with telephone security. In the late 1960s "phone phreaks" learned how to circumvent phone network security, in part for the challenge of the exercise, and in part for the ability to lower phone bill costs. In the next decade, modems and dumb terminals were used to break into mainframes. When personal computers became available, the technology became more sophisticated, as "war dialers" were created, able to automatically search for phone numbers connected to modems. Bulletin board systems (*BBSs*) became popular, and small groups set up private BBS meeting areas to spread hacking technology. The FBI apprehended several hackers (5), and hacking began to gain notoriety. As computers became networks of computers, a new breed of network-literate hackers evolved. In 1979 (6) Robert. H. Morris and Ken Thompson published a classic paper describing the Unix Password system. Later AT&T published a Unix-specific issue of their Technical Journal in October 1984, with similar articles (7). Morris also wrote articles on the weaknesses of TCP/IP.

Robert's son Robert T. Morris worked with his father, investigated the weaknesses of UUCP and published a paper discussing the sequence number attack of the TCP protocol stack in 1985 (8). Steve Bellovin wrote a thorough discussion of the weaknesses of the TCP stack in his 1989 paper (9).

In 1987, the Christmas Tree Worm spread over bitnet and the European Academic Research Network (10). In September 1987, Cliff Stoll described his experiences with a hacker from Germany (11). In 1988, the Internet Worm, created by Robert T. Morris, infected thousands of sites (3, 12). CERT was formed in November 1988, as a reaction from the Morris worm. In 1989, the WANK worm interfered with the NASA launch of Galileo, destined for Jupiter (13).

The movie *Wargames* came out in 1989. Hacking became "cool" and the popularity rose. In January 1991, Bill Cheswick experienced what he later called "An Evening with Berferd" (1). [Kevin Mitnick was captured twice (14, 15), the second time he used an IP hijacking technique described by Robert T. Morris (8)].

In 1994, Russian hacker Vladimir Levin stole $12 million from Citibank. Starting in 1996, several denials of service attacks occurred, such as the SYN Flood, Ping of Death, and the SMURF attack. Unsolicited e-mail also became a large problem. Various Windows NT protocols and applications were investigated, and numerous security problems were fixed, and more discovered.

In February 1998, an Israeli hacker known as Analyzer broke into US Department of Defense computers. In 1999, hackers developed systems using distributed that forged IP addresses, and executed coordinated attacks on victims. This is described as the distributed denial of service (*DDOS*) attacks and can be difficult to prevent, because the attacks come from compromised, and otherwise innocent, systems. In 1999 and 2000, an increase in self-replicating virus/worms occurred which were able to infect a system, and extract e-mail addresses from the victim, sending copies of itself to others in the victim's address book. The first was called the Melissa Virus.

Currently, experts believe 68% of all security breaches are caused by insider abuse. (Source Computer Security Institute, Inc. 1997). In 1996, Dan Farmer measured that 2/3 of the commercial sites on the Internet had obvious security vulnerabilities (13). A report was issued in 1996 that informed the US government that the Defense Information Systems Agency (*DISA*) may have experienced 250,000 attacks in 1995. The attacks were successful 65% of the time. Only 1 attack in 150 was detected and reported. Hacking is a serious problem, and hackers view most systems as having "lame" security. If hackers want to gain access, they probably will. An expert hacker can generate an exploit minutes after it is published on the *Bugtraq* mailing list. Vendors typically take days or weeks to respond to new threats. System administrators often have other tasks, and a home life. Every system will, therefore, have periods of vulnerability. What can be done to protect a network from attacks?

BIBLIOGRAPHY

1. B. Cheswick S. Bellovin *Firewalls and Internet Security*, Reading MA: Addison-Wesley, 1994 (a classic book that defined how firewalls should behave).
2. C. C. Wood *Information Security Policies Made Easy*, Houston, TX: Baseline Software, Inc. 1996 (describes how to write policies, with 730 examples).
3. S. Garfinkel G. Spafford *Practical UNIX and Internet Security*, 2d ed., Sebastopol, CA: O' Reilly and Associates, 1996.
4. D. B. Chapman E. D. Zwicky *Building Internet Firewalls*, Sebastopol, CA: O'Reilly and Associates, Inc., 1995. (a practical guide to implementing firewalls). (a) D. Farmer "Shall we dust Moscow?," from website: http://www.trouble.org/survey/
5. B. Landreth *Out of the Inner Circle.* Seattle: Microsoft Press, 1985 (a description of how a hacker operates).
6. R. Morris K. Thompson UNIX password security, *CACM* **22**, November 1979.
7. F. T. Gramp R. H. Morris UNIX system security, *AT&T Bell Lab. Tech. J.*, **63** (8): 1649–1672, October 1984.
8. R. Morris A weakness in the 4.2BSD UNIX TCP/IP software, Computing Science Technical Report 117, Murray Hill, NJ: AT&T Bell Laboratories, February 1985.
9. S. M. Bellovin Security problems in the TCP/IP protocol suite, *Comput. Commun. Rev.*, **19** (2): 32–48, April 1989.
10. Stallings, *William Network and Internet Security: Principles and Practice*, Englewood Cliffs NJ: Prentice-Hall, 1995.
11. C. Stoll *The Cuckoo's Egg*, Garden City: Doubleday, 1989.
12. E. Spafford *The Internet Worm Program: An Analysis*, Technical Report CSD-TR-823, Purdue University, 1988.
13. S. Dreytus *Underground*, London: Reed Books, 1997 (a discussion of the hacker's view of the world, with several cases detailed).
14. K. Haffnew J. Markoff *Cyberpunks: Outlaws and Hackers on the Computer Frontier*, New York: Simon and Schuster, 1991.
15. T. Shimomura J. Markoff *Takedown*, New York: Hyperion, 1996.

READING LIST

D. Farmer V. Wietse Improving the Security of Your Site by Breaking Into It, December 1993 from website: (ftp://coast.cs.purdue/pub/doc/general/Farmer_Venema_admin-guide-to-cracking-txt.Z)
C. Kaurman R. Perlman M. Speciner *Network Security: Private Communication in a Public World*, Englewood Cliffs, NJ: Prentice Hall, 1995, (a technical but readable description of how to implement secures protocols).
B. Schneier *Applied Cryptography*, 2d ed., New York: Wiley, 1996.
J. G. Steiner B. C. Neuman Jeffrey I. Shiller *Kerberos: An Authentication Service for Open Network Systems*, Winter 1988, USENIX Conference, Dallas.

BOOKS

AT&T Bell Lab. Tech. J. **63** (8), p, 2, October 1984 (several articles on UNIX and security).
P. H. Wood S. G. Kochan *UNIX System Security*, Hasbrouck Heights, NJ: Hayden Books, 1985. (The first book dedicated to UNIX security. The section on networking included a discussion of UUCP, HoneyDanBer, RJE, Hyperchannel and 3B Net. No mention of TCP. Contains source code for setuid programs, and chroot environments.)
D. Fiedler B. Hunter *UNIX System Administration*, Hasbrouck Heights, NJ: Hayden Books, 1986 (contains a chapter on security).
R. Farrow *UNIX System Security*, Reading, MA: Addison Wesley, 1991.
B. Sterling *Hacker Crackdown*, Bantam, 1992 (case studies of several hackers, includings. M. Garfinkel and G. Spafford, *Web Security and Commerce*, Sebastopol, CA: O'Reilly and Associates, Inc. 1997).
W. Dalton S. M. Fuler B. Kolosky J. Millecan C. Nachenberg K. S. Siyan L. Skok S. Tate *Windows NT 4: Security, Troubleshooting, and Optimization*, Indianapolis, IN: New Riders Publishing, 1996 (covers security).
S. A. Sutton *Windows NT Security Guide*, Addison Wesley, MA: Trusted System Services, Inc. 1997.
C. P. Meinel *The Happy Hacker*, American Eagle Publications, 1998 (a how-to book for hackers) www.ameaglepubs.com
L. McCarthy *Intranet Security*, Englewood Cliffs NJ: Prentice Hall, Sun Microsystems Press.

PAPERS

A. Muffet Crack Version 4.1: A Sensible password checker for UNIX, from website: http://www.users.dircon.co.uk/-crypto/
D. Farmer E. Spafford The COPS Security Checker System *Summer USENIX*, Anaheim, CA., 1990.
R. Baldwin Kuang: Rule-based security checking, MIT, June 1987 documentation in website: ftp://ftp.cert.org/pub/tools/cops/1.04/cops.tar.Z
D. Farmer W. Venema Security Administrator's Tool for Analyzing Networks, from website: http://www.fish.com/zen/satan/satan.html
W. Venema TCP WRAPPER—Network monitoring, access control, and booby traps, *Proc. 3rd USENIX Security Symp.*, Baltimore, September 1992.
G. Kim E. Spafford The Design and Implementation of Tripwire: A File System Integrity Checker, Technical Report CSD-TR-93-071, Purdue University, 1993
D. Safford D. Schales D. Hess *The TAMU Security Package: An ongoing Response to Internet Intruders in an Academic Environment*, 4th USENIX Security Symposium, 1993.
M. Blaze A Cryptographic File System for UNIX, *Proc. 1st ACM Conf. Comput. Commun. Security*, Fairfax, VA, November 1993.

MAILING LISTS

Firewalls

FWALL—Users

Bugtraq—send mail containing "subscribe bugtraq" to listserv@netspace.org

NTBugtraq—mail to listserv@ntbugtraq.com

Ntsecurity
Send mail containing “subscribe ntsecurity” to majordomo@iss.net
Academic—Firewalls
CERT—Advisory
RISKS
WWW—Security

WEB SITES

GENERAL

COAST—http://www.cs.purdue.edu/coast/coast.html—An excellent starting point. Purdue University is a leader in computer security.
CERT—http://www.cert.org—Another site to study. Read all of their advisories, and reports.
http://spider.osfl.disa.mil/cm/security/check_list/check_list.html—Contains security checklists for several operating systems.
http://www.sunworld.com/common/security-faq.html—Peter Gavin’s Solaris security FAQ.
http://www.security.org.il/ http://geek-girl.com/bugtraq/—Archives of the famous Bugtraq mailing list.
Http://www.wwdsi.com—Home of the SAINT program.
Http://www,kirch.net/unix-nt—John Kirch’s Unix versus Windows NT page.

WINDOWS NT SECURITY

http://www.microsoft.com/
http://www.ntsecurity.net/
http://www.trustedsystems.com/—Check out their Windows NT Security Guidelines
http://www.trustedsystems.com/NSA Guide.htm
http://www.iss.net/
http://iss.net/vd/bill_stout/ntexploits.htm
http://ntbugtraq.ntadvice.com—Site of Russ Cooper’s NTBUGTRAQ mailing list.
Http://www.ntbugtraq.com

“HACKER” SITES

http://www.10pht.com/—home of LOPHTCRACK
http://www.secnet.com—home of NTCRACK
http://www.rootshell.com—many exploit tools available
http://www.cultdeadcow.com/
http://www.phrack.com/—home of PHRACK magazine

RESPONSE TEAMS

CERT
http://www.cert.org/
FIRST—http://www.first.org/
AUSCERT—http://www.auscert.org.au/
CIAC—http://www.ciac.llnl.gov/
NIST CSRC
USENIX Association
System Administrators Guild

NEWSGROUPS

Comp.security.unix
- Comp.security.announce
- Comp.security.misc
- Comp.security.firewalls
- Alt.security
- Comp.admin.policy
- Comp-protocols-tcp-ip
- Comp.unix.admin
- Comp.unix.wizards

BRUCE BARNETT
General Electric