Principles Of Security

1. Confidentiality:

Confidentiality is probably the most common aspect of information security. The principle of confidentiality specifies that only the sender and intended recipient should be able to access the contents of a message.

Confidentiality gets compromised if an unauthorized person is able to access a message. Protection of confidential information is needed. An organization needs to guard against those malicious actions to endanger the confidentiality of its information.

Example: Banking customers accounts need to be kept secret. Confidentiality not only applies to the storage of the information but also applies to the transmission of information. When we send a piece of the information to be stored in a remote computer or when we retrieve a piece of information from a remote computer we need to conceal it during transmission. Interception causes loss of message confidentiality.

2. Integrity:

Information needs to be changed constantly. Integrity means that changes need to be done only by authorized entities and through authorized mechanisms. When the contents of a message are changed after the sender sends it, before it reaches the intended recipient it is said that integrity of the message is lost.

Integrity violation is not necessarily the result of a malicious act; an interruption in the system such as a power surge may also create unwanted changes in some information.

Modification causes loss of message integrity.

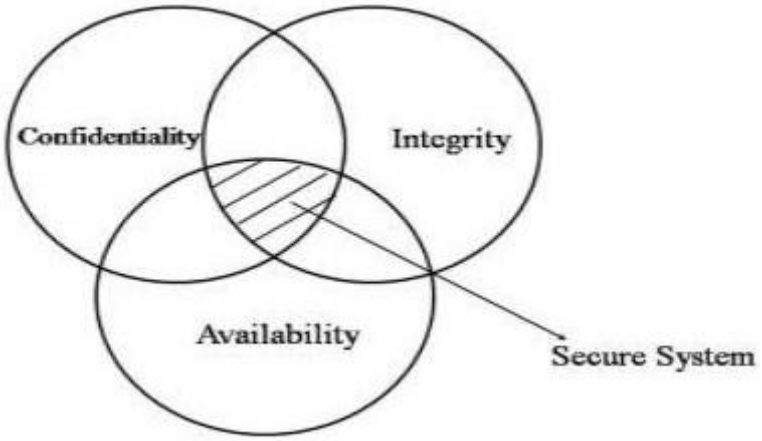
3. Availability:

The principle of availability states that resources should be available to authorized parties at all times. The information created and stored by an organization needs to be available to authorized entities. Information is useless if it is not available.

Information needs to be constantly changed which means it must be accessible to authorized entities. The unavailability of information is just as harmful for an organization as the lack of confidentiality or integrity.

Example: The situation can be difficult for a bank if the customer could not access their accounts for transactions.

Interruption puts the availability of resources in danger.



The diagram above explains the balance concept. The right balance of the three goals is needed to build a secure system. If the goals are not balanced then a small hole is created for attackers to nullify the other objectives of security. Having a highly confidential system but low availability then the system is not secure.

Example: A system can protect confidentiality and integrity but if the resource us not available the other two goals also are of no use.

Cyber Ethics

"Cyber ethics" refers to the code of responsible behavior on the Internet. Just as we are taught to act responsibly in everyday life with lessons such as "Don't take what doesn't belong to you" and "Do not harm others," we must act responsibly in the cyber world as well.

The basic rule is ***"Do not do something in cyberspace that you would consider wrong or illegal in everyday life."***

Do not use rude or offensive language.

Do not cyberbully.

Do not plagiarize.

Do not break into someone else's computer.

Do not use someone else's password.

Do not attempt to infect or in any way try to make someone else's computer unusable.

Adhere to copyright restrictions when downloading material from the Internet, including software, games, movies, or music.

Ethical Hacking

Ethical hacking and **ethical hacker** are terms used to describe hacking performed by a company or individual to help identify potential threats on a computer or network. An ethical hacker attempts to bypass system security and search for any weak points that could be exploited by malicious hackers. This information is then used by the organization to improve the system security, in an effort to minimize or eliminate any potential attacks.

For hacking to be deemed ethical, the hacker must obey the following rules:

Expressed (often written) permission to probe the network and attempt to identify potential security risks.

You respect the individual's or company's privacy.

You close out your work, not leaving anything open for you or someone else to exploit at a later time.

You let the software developer or hardware manufacturer know of any security vulnerabilities you locate in their software or hardware, if not already known by the company.

Phreaking

Phreaking is a slang term for hacking into secure telecommunication networks. The term phreaking originally referred to exploring and exploiting the phone networks by mimicking dialing tones to trigger the automatic switches using whistles or custom blue boxes designed for that purpose. Generally speaking, it was curiosity about how phone networks operated that motivated phreaks, rather than a desire to defraud telecommunications companies.

Phreaking has become synonymous with hacking now that networks have gone cellular and cracking them requires more clearly illegal methods.

Encryption

In cryptography, **encryption** is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating ciphertext that can be read only if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

Types

Symmetric key / Private key

In symmetric-key schemes,[1] the encryption and decryption keys are the same. Communicating parties must have the same key in order to achieve secure communication.

Public key

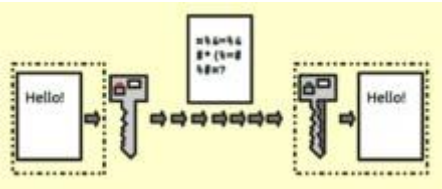


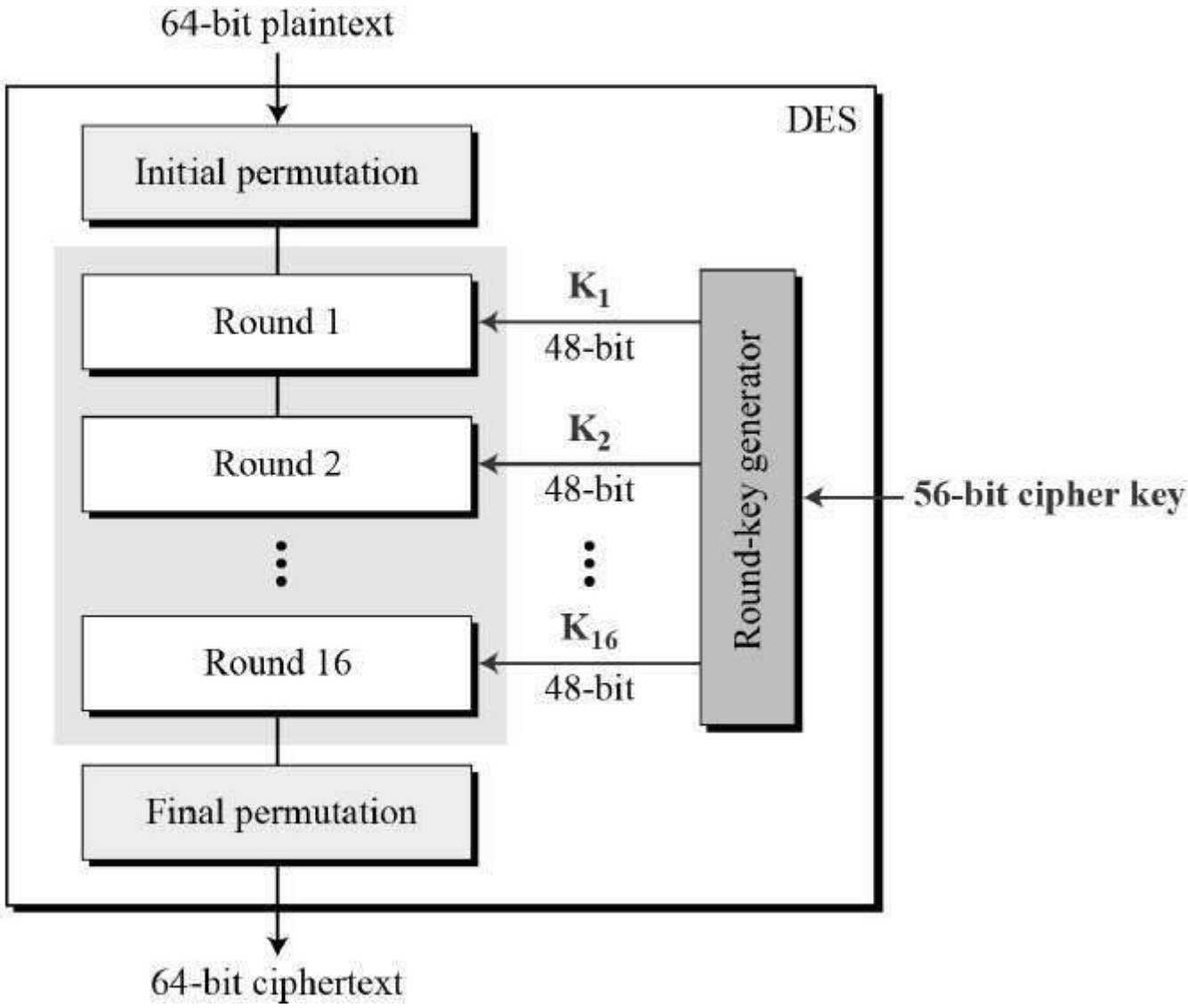
Illustration of how encryption is used within servers Public key encryption.

In public-key encryption schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read.[2] Public-key encryption was first described in a secret document in 1973; before then all encryption schemes were symmetric-key (also called private-key).

Data Encryption Standard

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −



Since DES is based on the Feistel Cipher, all that is required to specify DES is −

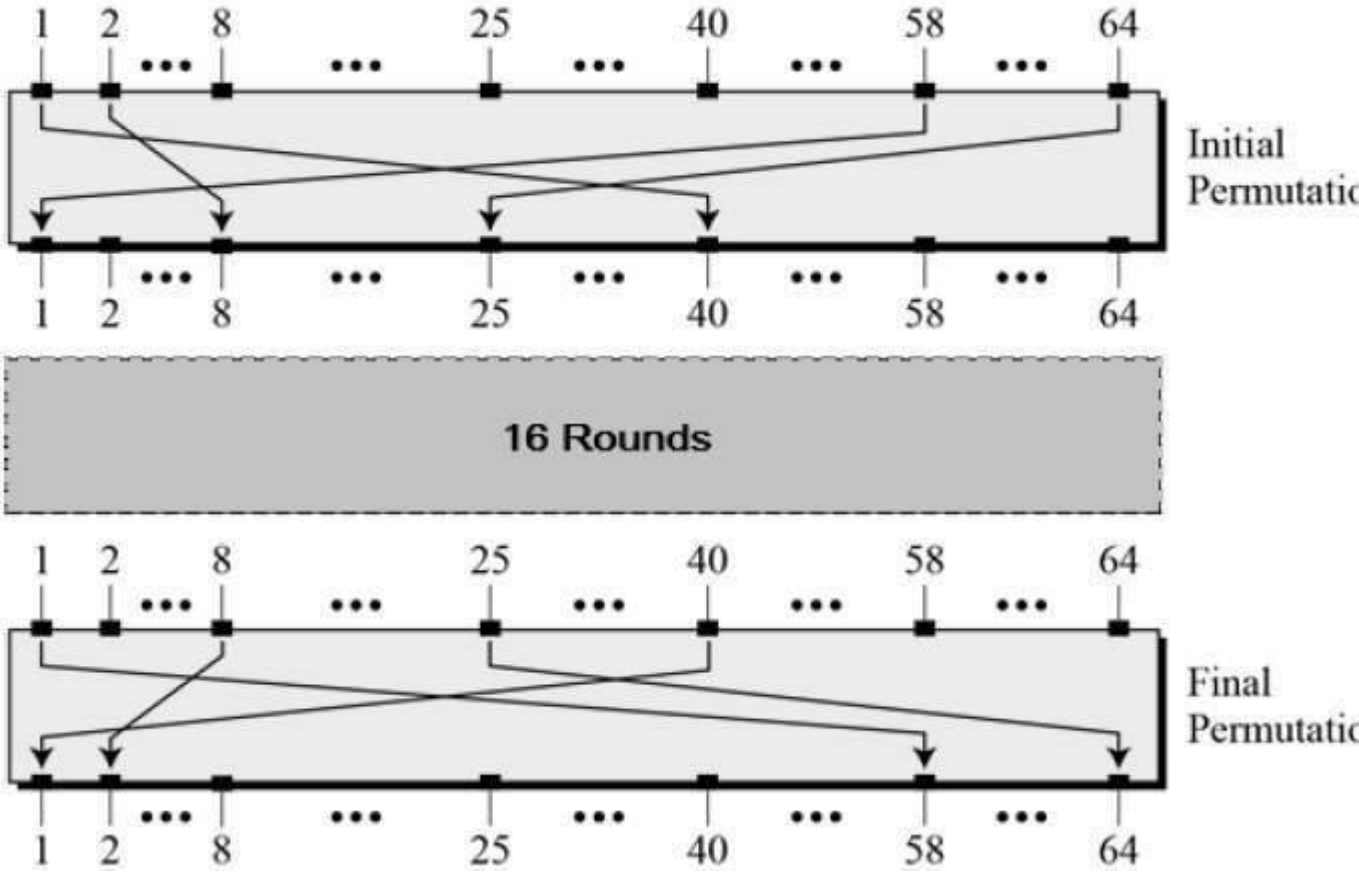
Round function

Key schedule

Any additional processing − Initial and final permutation

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −



MD5

The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating digital signatures. MD5 has been deprecated for uses other than as a non-cryptographic checksum to verify data integrity and detect unintentional data corruption.

Message digest algorithm characteristics

Message digests, also known as hash functions, are one-way functions; they accept a message of any size as input, and produce as output a fixed-length message digest.

MD5 is the third message digest algorithm created by Rivest. All three (the others are MD2 and MD4) have similar structures, but MD2 was optimized for 8-bit machines, in comparison with the two later formulas, which are optimized for 32-bit machines. The MD5 algorithm is an extension of MD4, which the critical review found to be fast, but possibly not absolutely secure. In comparison, MD5 is not quite as fast as the MD4 algorithm, but offered much more assurance of data security.

How MD5 works

The MD5 message digest hashing algorithm processes data in 512-bit blocks, broken down into 16 words composed of 32 bits each. The output from MD5 is a 128-bit message digest value.

Computation of the MD5 digest value is performed in separate stages that process each 512-bit block of data along with the value computed in the preceding stage. The first stage begins with

the message digest values initialized using consecutive hexadecimalnumerical values. Each stage includes four message digest passes which manipulate values in the current data block and values processed from the previous block. The final value computed from the last block becomes the MD5 digest for that block.

MD5 security

The goal of any message digest function is to produce digests that appear to be random. To be considered cryptographically secure, the hash function should meet two requirements: first, that it is impossible for an attacker to generate a message matching a specific hash value; and second, that it is impossible for an attacker to create two messages that produce the same hash value.

MD5 hashes are no longer considered cryptographically secure, and they should not be used for cryptographic authentication.

What is SSL?

SSL (Secure Sockets Layer) is a standard security protocol for establishing encrypted links between a web server and a browser in an online communication.

The usage of SSL technology ensures that all data transmitted between the web server and browser remains encrypted.

An **SSL certificate** is necessary to create SSL connection. You would need to give all details about the identity of your website and your company as and when you choose to activate SSL on your web server. Following this, two cryptographic keys are created - a Private Key and a Public Key.

The next step is the submission of the CSR (Certificate Signing Request), which is a data file that contains your details as well as your Public Key. The CA (Certification Authority) would then validate your details. Following successful authentication of all details, you will be issued SSL certificate. The newly-issued SSL would be matched to your Private Key. From this point onwards, an encrypted link is established by your web server between your website and the customer's web browser.

On the apparent level, the presence of an **SSL protocol** and an encrypted session is indicated by the presence of the lock icon in the address bar. A click on the lock icon displays to a user/customer details about your SSL. It's to be remembered that SSL Certificates are issued to either companies or legally accountable individuals only after proper authentication.

An **SSL Certificate** comprises of your domain name, the name of your company and other things like your address, your city, your state and your country. It would also show the expiration date of the SSL plus details of the issuing CA. Whenever a browser initiates a connection with a SSL secured website , it will first retrieve the site's SSL Certificate to check if it's still valid. It's also verified that the CA is one that the browser trusts, and also that the certificate is being used by the website for which it has been issued. If any of these checks fail, a warning will be displayed to the user, indicating that the website is not secured by a valid SSL certificate.

What is SSL/TLS Certificate?

SSL or **TLS (Transport Layer Security)** certificates are data files that bind a cryptographic key to the details of an organization. When SSL/TLS certificate is installed on a web server, it enables a secure connection between the web server and the browser that connects to it. The website's URL is prefixed with "https" instead of "http" and a padlock is shown on the address bar. If the website uses an extended validation (EV) certificate, then the browser may also show a green address bar.

What is SSL used for?

The *SSL protocol* is used by millions of online business to protect their customers, ensuring their online transactions remain confidential. A web page should use **encryption** when it expects users to submit confidential data, including personal information, passwords, or credit card details. All web browsers have the ability to interact with secured sites so long as the site's certificate is issued by a trusted CA.

Why do I need SSL certificate?

The internet has spawned new global business opportunities for enterprises conducting online commerce. However, that growth has also attracted fraudsters and cyber criminals who are ready to exploit any opportunity to steal consumer bank account numbers and card details. Any moderately skilled hacker can easily intercept and read the traffic unless the connection between a client (e.g. internet browser) and a web server is encrypted.

How Does SSL Work?

The following graphic explains how SSL Certificate works on a website. The process of how an 'SSL handshake' takes place is explained below:

An end-user asks their browser to make a secure connection to a website (e.g.https://www.example.com)

The browser obtains the IP address of the site from a DNS server then requests a secure connection to the website.

To initiate this secure connection, the browser requests that the server identifies itself by sending a copy of its SSL certificate to the browser.

The browser checks the certificate to ensure:

That it is signed by a trusted CA

That it is valid - that it has not expired or been revoked

That it confirms to required security standards on key lengths and other items.

That the domain listed on the certificate matches the domain that was requested by the user.

When the browser confirms that the website can be trusted, it creates a symmetric session key which it encrypts with the public key in the website's certificate. The session key is then sent to the web server.

The web server uses its private key to decrypt the symmetric session key.

The server sends back an acknowledgement that is encrypted with the session key.

From now on, all data transmitted between the server and the browser is encrypted and secure.

How can I tell when a site uses SSL?

A web page using SSL will display

"https://" instead of "http://" before the website's address in the browser's address bar

A padlock icon in the address bar of the browser before the address.

With an Extended Validation Certificate, the address bar also shows the registered name of the company that owns the website, the name of the issuing CA and, an additional green security indicator.

SSH

The SSH protocol (also referred to as Secure Shell) is a method for secure remote login from one computer to another. It provides several alternative options for strong authentication, and it protects the communications security and integrity with strong encryption. It is a secure alternative to the non-protected login protocols (such as telnet, rlogin) and insecure file transfer methods (such as FTP).

TYPICAL USES OF THE SSH PROTOCOL

The protocol is used in corporate networks for:

providing secure access for users and automated processes

interactive and automated file transfers

issuing remote commands

managing network infrastructure and other mission-critical system components.

HOW DOES THE SSH PROTOCOL WORK

The protocol works in the client-server model, which means that the connection is established by the SSH client connecting to the SSH server. The SSH client drives the connection setup process and uses public key cryptography to verify the identity of the SSH server. After the setup phase the SSH protocol uses strong symmetric encryption and hashing algorithms to ensure the privacy and integrity of the data that is exchanged between the client and server.

The figure below presents a simplified setup flow of a secure shell connection.

STRONG AUTHENTICATION WITH SSH KEYS

There are several options that can be used for user authentication. The most common ones are passwords and public key authentication.

The public key authentication method is primarily used for automation and sometimes by system administrators for single sign-on. It has turned out to be much more widely used than we ever anticipated. The idea is to have a cryptographic key pair - public key and private key - and configure the public key on a server to authorize access and grant anyone who has a copy of the private key access to the server. The keys used for authentication are called SSH keys. Public key authentication is also used with smartcards, such as the CAC and PIV cards used by US government.

The main use of key-based authentication is to enable secure automation. Automated secure shell file transfers are used to seamlessly integrate applications and also for automated systems & configuration management.

We have found that large organizations have way more SSH keys than they imagine, and managing SSH keys has become very important. SSH keys grant access as user names and passwords do. They require a similar provisioning and termination processes.

In some cases we have found several million SSH keys authorizing access into production servers in customer environments, with 90% of the keys actually being unused and representing access that was provisioned but never terminated. Ensuring proper policies, processes, and audits also for SSH usage is critical for proper identity and access management. Traditional identity management projects have overlooked as much as 90% of all credentials by ignoring SSH keys. We provide services and tools for implementing SSH key management.

SSH PROVIDES STRONG ENCRYPTION AND INTEGRITY PROTECTION

Once a connection has been established between the SSH client and server, the data that is transmitted is encrypted according to the parameters negotiated in the setup. During the negotiation the client and server agree on the symmetric encryption algorithm to be used and generate the encryption key that will be used. The traffic between the communicating parties is protected with industry standard strong encryption algorithms (such as AES (Advanced Encryption Standard)), and the SSH protocol also includes a mechanism that ensures the integrity of the transmitted data by using standard hash algoritms (such as SHA-2 (Standard Hashing Algorithm)).

SFTP FILE TRANSFER PROTOCOL

The SFTP (SSH File Transfer Protocol) is probably the most widely used secure file transfer protocol today. It runs over SSH, and is currently documented in draft-ietf-secsh-filexfer-02

HTTP

HTTP is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is reconstructed from the different sub-documents fetched, for instance text, layout description, images, videos, scripts, and more.

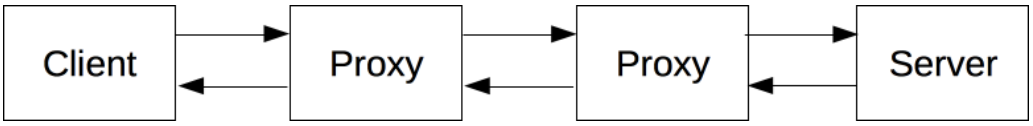
Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called *requests* and the messages sent by the server as an answer are called *responses*.

Designed in the early 1990s, HTTP is an extensible protocol which has evolved over time. It is an application layer protocol that is sent over TCP, or over a TLS-encrypted TCP connection, though any reliable transport protocol could theoretically be used. Due to its extensibility, it is used to not only fetch hypertext documents, but also images and videos or to post content to servers, like with HTML form results. HTTP can also be used to fetch parts of documents to update Web pages on demand.

Components of HTTP-based systems

HTTP is a client-server protocol: requests are sent by one entity, the user-agent (or a proxy on behalf of it). Most of the time the user-agent is a Web browser, but it can be anything, for example a robot that crawls the Web to populate and maintain a search engine index.

Each individual request is sent to a server, which will handle it and provide an answer, called the *response*. Between this request and response there are numerous entities, collectively designated as proxies, which perform different operations and act as gateways or caches, for example.



In reality, there are more computers between a browser and the server handling the request: there are routers, modems, and more. Thanks to the layered design of the Web, these are hidden in the network and transport layers. HTTP is on top at the application layer. Although important to diagnose network problems, the underlying layers are mostly irrelevant to the description of HTTP.

Client: the user-agent

The *user-agent* is any tool that acts on the behalf of the user. This role is primarily performed by the Web browser; a few exceptions being programs used by engineers, and Web developers to debug their applications.

The browser is **always** the entity initiating the request. It is never the server (though some mechanisms have been added over the years to simulate server-initiated messages).

To present a Web page, the browser sends an original request to fetch the HTML document from the page. It then parses this file, fetching additional requests corresponding to execution scripts, layout information (CSS) to display, and sub-resources contained within the page (usually images and videos). The Web browser then mixes these resources to present to the user a complete document, the Web page. Scripts executed by the browser can fetch more resources in later phases and the browser updates the Web page accordingly.

A Web page is a hypertext document. This means some parts of displayed text are links which can be activated (usually by a click of the mouse) to fetch a new Web page, allowing the user to direct their user-agent and navigate through the Web. The browser translates these directions in HTTP requests, and further interprets the HTTP responses to present the user with a clear response.

The Web server

On the opposite side of the communication channel, is the server which *serves* the document as requested by the client. A server presents only as a single machine virtually: this is because it may actually be a collection of servers, sharing the load (load balancing) or a complex piece of software interrogating other computers (like cache, a DB server, e-commerce servers, …), totally or partially generating the document on demand.

A server is not necessarily a single machine, but several servers can be hosted on the same machine. With HTTP/1.1 and the Host header, they may even share the same IP address.

Proxies

Between the Web browser and the server, numerous computers and machines relay the HTTP messages. Due to the layered structure of the Web stack, most of these operate at either the transport, network or physical levels, becoming transparent at the HTTP layer and potentially making a significant impact on performance. Those operating at the application layers are generally called **proxies**. These can be transparent, or not (changing requests not going through them), and may perform numerous functions:

caching (the cache can be public or private, like the browser cache)

filtering (like an antivirus scan, parental controls, …)

load balancing (to allow multiple servers to serve the different requests)

authentication (to control access to different resources)

logging (allowing the storage of historical information)

Basic aspects of HTTP

HTTP is simple

Even with more complexity, introduced in HTTP/2 by encapsulating HTTP messages into frames, HTTP is generally designed to be simple and human readable. HTTP messages can be read and understood by humans, providing easier developer testing, and reduced complexity for new-comers.

HTTP is extensible

Introduced in HTTP/1.0, HTTP headers made this protocol easy to extend and experiment with. New functionality can even be introduced by a simple agreement between a client and a server about a new header's semantics.

HTTP is stateless, but not sessionless

HTTP is stateless: there is no link between two requests being successively carried out on the same connection. This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions. Using header extensibility, HTTP Cookies are added to the workflow, allowing session creation on each HTTP request to share the same context, or the same state.

HTTP and connections

A connection is controlled at the transport layer, and therefore fundamentally out of scope for HTTP. Though HTTP doesn't require the underlying transport protocol to be connection-based; only requiring it to be *reliable*, or not lose messages (so at minimum presenting an error). Among the two most common transport protocols on the Internet, TCP is reliable and UDP isn't. HTTP subsequently relies on the TCP standard, which is connection-based, even though a connection is not always required.

HTTP/1.0 opened a TCP connection for each request/response exchange, introducing two major flaws: opening a connection needs several round-trips of messages and therefore slow, but becomes more efficient when several messages are sent, and regularly sent: *warm* connections are more efficient than *cold* ones.

In order to mitigate these flaws, HTTP/1.1 introduced pipelining (which proved difficult to implement) and persistent connections: the underlying TCP connection can be partially controlled using the Connection header. HTTP/2 went a step further by multiplexing messages over a single connection, helping keep the connection warm, and more efficient.

Experiments are in progress to design a better transport protocol more suited to HTTP. For example, Google is experimenting with QUIC which builds on UDP to provide a more reliable and efficient transport protocol.

What can be controlled by HTTP

This extensible nature of HTTP has, over time, allowed for more control and functionality of the Web. Cache or authentication methods were functions handled early in HTTP history. The ability to relax the *origin constraint*, by contrast, has only been added in the 2010s.

Here is a list of common features controllable with HTTP.

Cache
How documents are cached can be controlled by HTTP. The server can instruct proxies, and clients, what to cache and for how long. The client can instruct intermediate cache proxies to ignore the stored document.

Relaxing the origin constraint
To prevent snooping and other privacy invasions, Web browsers enforce strict separation between Web sites. Only pages from the **same origin** can access all the information of a Web page. Though such constraint is a burden to the server, HTTP headers can relax this strict separation server-side, allowing a document to become a patchwork of information sourced from different domains (there could even be security-related reasons to do so).

Authentication
Some pages may be protected so only specific users can access it. Basic authentication may be provided by HTTP, either using the WWW-Authenticate and similar headers, or by setting a specific session using HTTP cookies.

Proxy and tunneling
Servers and/or clients are often located on intranets and hide their true IP address to others. HTTP requests then go through proxies to cross this network barrier. Not all proxies are HTTP proxies. The SOCKS protocol, for example, operates at a lower level. Others, like ftp, can be handled by these proxies.

Sessions

Using HTTP cookies allows you to link requests with the state of the server. This creates sessions, despite basic HTTP being a state-less protocol. This is useful not only for e-commerce shopping baskets, but also for any site allowing user configuration of the output.

Conclusion

HTTP is an extensible protocol that is easy to use. The client-server structure, combined with the ability to simply add headers, allows HTTP to advance along with the extended capabilities of the Web.

Though HTTP/2 adds some complexity, by embedding HTTP messages in frames to improve performance, the basic structure of messages has stayed the same since HTTP/1.0. Session flow remains simple, allowing it to be investigated, and debugged with a simple HTTP message monitor.

Digital Signature

The digital equivalent of a handwritten signature or stamped seal, but offering far more inherent security, a digital signature is intended to solve the problem of tampering and impersonation in digital communications. Digital signatures can provide the added assurances of evidence to origin, identity and status of an electronic document, transaction or message, as well as acknowledging informed consent by the signer.

In many countries, including the United States, digital signatures have the same legal significance as the more traditional forms of signed documents. The United States Government Printing Office publishes electronic versions of the budget, public and private laws, and congressional bills with digital signatures.

How digital signatures work

Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm such as RSA, one can generate two keys that are mathematically linked: one private and one public. To create a digital signature, signing software (such as an email program) creates a one-way hash of the electronic data to be signed. The private key is then used to encrypt the hash. The encrypted hash -- along with other information, such as the hashing algorithm -- is the digital signature. The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter. This saves time since hashing is much faster than signing.

The value of the hash is unique to the hashed data. Any change in the data, even changing or deleting a single character, results in a different value. This attribute enables others to validate the integrity of the data by using the signer's public key to decrypt the hash. If the decrypted hash matches a second computed hash of the same data, it proves that the data hasn't changed

since it was signed. If the two hashes don't match, the data has either been tampered with in some way (integrity) or the signature was created with a private key that doesn't correspond to the public key presented by the signer (authentication).

A digital signature can be used with any kind of message -- whether it is encrypted or not -- simply so the receiver can be sure of the sender's identity and that the message arrived intact. Digital signatures make it difficult for the signer to deny having signed something (non-repudiation) -- assuming their private key has not been compromised -- as the digital signature is unique to both the document and the signer, and it binds them together. A digital certificate, an electronic document that contains the digital signature of the certificate-issuing authority, binds together a public key with an identity and can be used to verify a public key belongs to a particular person or entity.

If the two hash values match, the message has not been tampered with, and the receiver knows the message is from sender.

Most modern email programs support the use of digital signatures and digital certificates, making it easy to sign any outgoing emails and validate digitally signed incoming messages. Digital signatures are also used extensively to provide proof of authenticity, data integrity and non-repudiation of communications and transactions conducted over the Internet.

Digital Certificate

A digital certificate is an electronic "passport" that allows a person, computer or organization to exchange information securely over the Internet using the public keyinfrastructure (PKI). A digital certificate may also be referred to as a public key certificate.

Just like a passport, a digital certificate provides identifying information, is forgery resistant and can be verified because it was issued by an official, trusted agency. The certificate contains the name of the certificate holder, a serial number, expiration dates, a copy of the certificate holder's public key (used for encrypting messages and digital signatures) and the digital signature of the certificate-issuing authority (CA) so that a recipient can verify that the certificate is real.

To provide evidence that a certificate is genuine and valid, it is digitally signed by a root certificate belonging to a trusted certificate authority. Operating systems and browsers maintain lists of trusted CA root certificates so they can easily verify certificates that the CAs have issued and signed. When PKI is deployed internally, digital certificates can be self-signed.

IP SEC

Internet Protocol Security

is a network protocol suite that authenticates and encrypts the packets of data sent over a network. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to use during the session. IPsec can protect data flows between a pair of hosts (*host-to-host*), between a pair of security gateways (*network-to-network*), or between a security gateway and a host (*network-to-host*).[1] Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer

authentication, data-origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at the Transport Layer (TLS) and the Application layer (SSH). IPsec can automatically secure applications at the IP layer.

Virus Worms Trojans

Introduction

Viruses, worms, Trojans, and bots are all part of a class of software called malware. Malware or malicious code (malcode) is short for malicious software. It is code or software that is specifically designed to damage, disrupt, steal, or in general inflict some other “bad” or illegitimate action on data, hosts, or networks.

There are many different classes of malware that have varying ways of infecting systems and propagating themselves. Malware can infect systems by being bundled with other programs or attached as macros to files. Others are installed by exploiting a known vulnerability in an operating system (OS), network device, or other software, such as a hole in a browser that only requires users to visit a website to infect their computers. The vast majority, however, are installed by some action from a user, such as clicking an e-mail attachment or downloading a file from the Internet.

Some of the more commonly known types of malware are viruses, worms, Trojans, bots, back doors, spyware, and adware. Damage from malware varies from causing minor irritation (such as browser popup ads), to stealing confidential information or money, destroying data, and compromising and/or entirely disabling systems and networks.

Malware cannot damage the physical hardware of systems and network equipment, but it can damage the data and software residing on the equipment. Malware should also not be confused with defective software, which is intended for legitimate purposes but has errors or bugs.

Classes of Malicious Software

Two of the most common types of malware are viruses and worms. These types of programs are able to self-replicate and can spread copies of themselves, which might even be modified copies. To be classified as a virus or worm, malware must have the ability to propagate. The difference is that a worm operates more or less independently of other files, whereas a virus depends on a host program to spread itself. These and other classes of malicious software are described below.

Viruses

A computer virus is a type of malware that propagates by inserting a copy of itself into and becoming part of another program. It spreads from one computer to another, leaving infections as it travels. Viruses can range in severity from causing mildly annoying effects to damaging data or software and causing denial-of-service (DoS) conditions. Almost all viruses are attached to an executable file, which means the virus may exist on a system but will not be active or able to spread until a user runs or opens the malicious host file or program. When the host code is executed, the viral code is executed as well. Normally, the host program keeps functioning after it

is infected by the virus. However, some viruses overwrite other programs with copies of themselves, which destroys the host program altogether. Viruses spread when the software or document they are attached to is transferred from one computer to another using the network, a disk, file sharing, or infected e-mail attachments.

Worms

Computer worms are similar to viruses in that they replicate functional copies of themselves and can cause the same type of damage. In contrast to viruses, which require the spreading of an infected host file, worms are standalone software and do not require a host program or human help to propagate. To spread, worms either exploit a vulnerability on the target system or use some kind of social engineering to trick users into executing them. A worm enters a computer through a vulnerability in the system and takes advantage of file-transport or information-transport features on the system, allowing it to travel unaided.

Trojans

A Trojan is another type of malware named after the wooden horse the Greeks used to infiltrate Troy. It is a harmful piece of software that looks legitimate. Users are typically tricked into loading and executing it on their systems. After it is activated, it can achieve any number of attacks on the host, from irritating the user (popping up windows or changing desktops) to damaging the host (deleting files, stealing data, or activating and spreading other malware, such as viruses). Trojans are also known to create back doors to give malicious users access to the system.

Unlike viruses and worms, Trojans do not reproduce by infecting other files nor do they self-replicate. Trojans must spread through user interaction such as opening an e-mail attachment or downloading and running a file from the Internet.

Bots

"Bot" is derived from the word "robot" and is an automated process that interacts with other network services. Bots often automate tasks and provide information or services that would otherwise be conducted by a human being. A typical use of bots is to gather information (such as web crawlers), or interact automatically with instant messaging (IM), Internet Relay Chat (IRC), or other web interfaces. They may also be used to interact dynamically with websites.

Bots can be used for either good or malicious intent. A malicious bot is self-propagating malware designed to infect a host and connect back to a central server or servers that act as a command and control (C&C) center for an entire network of compromised devices, or "botnet." With a botnet, attackers can launch broad-based, "remote-control," flood-type attacks against their target(s). In addition to the worm-like ability to self-propagate, bots can include the ability to log keystrokes, gather passwords, capture and analyze packets, gather financial information, launch DoS attacks, relay spam, and open back doors on the infected host. Bots have all the advantages of worms, but are generally much more versatile in their infection vector, and are often modified within hours of publication of a new exploit. They have been known to exploit back doors opened by worms and viruses, which allows them to access networks that have good perimeter control. Bots rarely announce their presence with high scan rates, which damage network infrastructure; instead they infect networks in a way that escapes immediate notice.

What is Heuristic Antivirus Detection?

Several antivirus solutions and internet security suites claim to utilize heuristic detection to find malware, but what does that mean? The software providers claim heuristic technologies can find viruses that have previously been unknown, detecting and defending from new malware that has yet to be discovered and added to virus definition files. You might find a program that accomplishes this feat among premium security suites. But how can the software do that? Does this proactive detection really work? And if so, is it worth it?

Heuristic Detection Methods

Antivirus software may use one or several techniques to proactively detect malware. The main essence of each method is to analyze the suspicious file s characteristics and behavior to determine if it is indeed malware. Here are a few of the common heuristic/behavioral scanning techniques:

File Emulation: Also known as sandbox testing or dynamic scanning, file emulation allows the file to run in a controlled virtual system (or sandbox) to see what it does. Think of this approach as watching the virus in a sterile testing room with a two-way mirror. If the file acts like a virus, it s deemed a virus.

File Analysis: Think of this method as the suspected file having to go through airport security (complete with carry-on check). File analysis involves the software taking an in-depth look at the file and trying to determine its intent, destination, and purpose. Perhaps the file has instructions to delete certain files, and should be considered a virus.

Generic Signature Detection: This technique is particularly designed to locate variations of viruses. Several viruses are re-created and make themselves known by a variety of names, but essentially come from the same family (or classification). Genetic detection uses previous antivirus definitions to locate these similar cousins even if they use a slightly different name or include some unusual characters. The best way to illustrate this idea is with identical twins. They may have slightly different fingerprints, but their DNA is identical.

Pros and Cons of Heuristic Detection

Heuristic Detection is an effective way to locate unknown threats for the most up-to-date realtime protection, but there are downsides.
Obviously this sort of scanning and analysis can take some time, which may slow-down system performance.

The main concern with heuristic detection is that it often increases false positives. False positives are when the antivirus software determines a file is malicious (and quarantines or deletes it) when in reality it is perfectly fine and/or desired. Because some files may look like viruses but really aren t, they are restricted and stopped from working on your computer.

Is Heuristic Detection Worth It?

Security professionals have come a long way with heuristic detection, and are still working to find the perfect balance that provides proactive protection without causing the hassle of false positives. They have also made the processes much faster by utilizing computer resources more effectively and utilizing better scanning technology.

We expect heuristic detection to continue to be dynamic and improve in speed, efficiency, and efficacy. For now, it is a good idea to utilize a security solution that includes heuristic detection

methods. If you find they get in the way, you can usually turn them off anyways. But for utmost protection and truly proactive prevention, heuristic antivirus detection is definitely the way to go.

Firewall

In computing, a **firewall** is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.

Firewalls are often categorized as either **network firewalls** or **host-based firewalls**. Network firewalls filter traffic between two or more networks and run on network hardware. Host-based firewalls run on host computers and control network traffic in and out of those machines.

First generation: packet filters

The first reported type of network firewall is called a packet filter. Packet filters look at network addresses and ports of packets to determine if they must be allowed, dropped, or rejected.. Packet filters act by inspecting packets which transfer between computers on the Internet. When a packet does not match the packet filter's set of filtering rules, the packet filter either drops (silently discards) the packet, or rejects the packet (discards it and generate an Internet Control Message Protocol notification for the sender). Conversely, when a packet matches one or more programmed filter rules, it is allowed to pass. Elements that can be defined in a packet filter rule include a packet's source and destination addresses, protocol, and source and destination ports.

Second generation: "stateful" filters

Also known as circuit-level gateways.

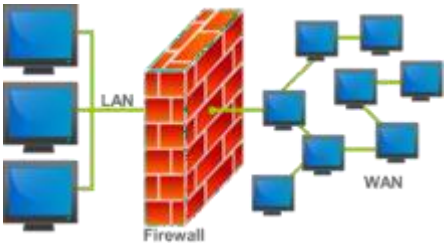
Second-generation firewalls perform the work of their first-generation predecessors but operate up to layer 4 (transport layer) of the OSI model. This is achieved by retaining packets until enough information is available to make a judgement about its state. Known as stateful packet inspection, it records all connections passing through it and determines whether a packet is the start of a new connection, a part of an existing connection, or not part of any connection. Though static rules are still used, these rules can now contain *connection state* as one of their test criteria.

Certain denial-of-service attacks bombard the firewall with thousands of fake connection packets in an attempt to overwhelm it by filling its connection state memory.

Third generation: application layer

The key benefit of application layer filtering is that it can "understand" certain applications and protocols (such as File Transfer Protocol(FTP), Domain Name System (DNS), or Hypertext Transfer Protocol (HTTP)). This is useful as it is able to detect if an unwanted application or service is attempting to bypass the firewall using a protocol on an allowed port, or detect if a protocol is being abused in any harmful way.

Types



An illustration of where a firewall would be located in a network

Firewalls are generally categorized as network-based or host-based. Network-based firewalls are positioned on the gateway computers of LANs, WANs and intranets. They are either software appliances running on general-purpose hardware, or hardware-based firewall computer appliances. Firewall appliances may also offer other functionality to the internal network they protect, such as acting as a DHCP or VPN server for that network. Host-based firewalls are positioned on the network node itself and control network traffic in and out of those machines.The host-based firewall may be a daemon or service as a part of the operating system or an agent application such as endpoint security or protection. Each has advantages and disadvantages. However, each has a role in layered security.

Firewalls also vary in type depending on where communication originates, where it is intercepted, and the state of communication being traced.

Network layer or packet filters

Network layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set. The firewall administrator may define the rules; or default rules may apply. The term "packet filter" originated in the context of BSD operating systems.

Network layer firewalls generally fall into two sub-categories, stateful and stateless.

Stateful firewalls maintain context about active sessions, and use that "state information" to speed packet processing. Any existing network connection can be described by several properties, including source and destination IP address, UDP or TCP ports, and the current stage of the connection's lifetime (including session initiation, handshaking, data transfer, or completion connection). If a packet does not match an existing connection, it will be evaluated according to the ruleset for new connections. If a packet matches an existing connection based on comparison with the firewall's state table, it will be allowed to pass without further processing.

Stateless firewalls require less memory, and can be faster for simple filters that require less time to filter than to look up a session. They may also be necessary for filtering stateless network protocols that have no concept of a session. However, they cannot make more complex decisions based on what stage communications between hosts have reached.

Newer firewalls can filter traffic based on many packet attributes like source IP address, source port, destination IP address or port, destination service like HTTP or FTP. They can filter based on protocols, TTL values, network block of the originator, of the source, and many other attributes.

Application-layer

Application-layer firewalls work on the application level of the TCP/IP stack (i.e., all browser traffic, or all telnet or FTP traffic), and may intercept all packets traveling to or from an application. They block other packets (usually dropping them without acknowledgment to the sender).

On inspecting all packets for improper content, firewalls can restrict or prevent outright the spread of networked computer worms and Trojans. The additional inspection criteria can add extra latency to the forwarding of packets to their destination.

Application firewalls function by determining whether a process should accept any given connection. Application firewalls accomplish their function by hooking into socket calls to filter the connections between the application layer and the lower layers of the OSI model. Application firewalls that hook into socket calls are also referred to as socket filters. Application firewalls work much like a packet filter but application filters apply filtering rules (allow/block) on a per process basis instead of filtering connections on a per port basis. Generally, prompts are used to define rules for processes that have not yet received a connection. It is rare to find application firewalls not combined or used in conjunction with a packet filter.

Also, application firewalls further filter connections by examining the process ID of data packets against a rule set for the local process involved in the data transmission. The extent of the filtering that occurs is defined by the provided rule set. Given the variety of software that exists, application firewalls only have more complex rule sets for the standard services, such as sharing services. These per-process rule sets have limited efficacy in filtering every possible association that may occur with other processes. Also, these per-process rule sets cannot defend against modification of the process via exploitation, such as memory corruption exploits. Because of these limitations, application firewalls are beginning to be supplanted by a new generation of application firewalls that rely on mandatory access control (MAC), also referred to as sandboxing, to protect vulnerable services.

Proxies

A proxy server (running either on dedicated hardware or as software on a general-purpose machine) may act as a firewall by responding to input packets (connection requests, for example) in the manner of an application, while blocking other packets. A proxy server is a gateway from one network to another for a specific network application, in the sense that it functions as a proxy on behalf of the network user.[2]

Proxies make tampering with an internal system from the external network more difficult, so that misuse of one internal system would not necessarily cause a security breach exploitable from outside the firewall (as long as the application proxy remains intact and properly configured). Conversely, intruders may hijack a publicly reachable system and use it as a proxy for their own purposes; the proxy then masquerades as that system to other internal machines. While use of internal address spaces enhances security, crackers may still employ methods such as IP spoofing to attempt to pass packets to a target network.

Intrusion Detection System

An **intrusion detection system** (**IDS**) is a device or software application that monitors a network or systems for malicious activity or policy violations. Any malicious activity or violation is typically reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system combines outputs from multiple sources, and uses alarm filtering techniques to distinguish malicious activity from false alarms.

Comparison with firewalls

Though they both relate to network security, an IDS differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls limit access between networks to prevent intrusion and do not signal an attack from inside the network. An IDS describes a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system. This is traditionally achieved by examining network communications, identifying heuristics and patterns (often known as signatures) of common computer attacks, and taking action to alert operators. A system that terminates connections is called an intrusion prevention system, and is another form of an application layer firewall.

Intrusion detection

IDS can be classified by where detection takes place (network or host) and the detection method that is employed.

Analyzed activity

Network intrusion detection systems

Network intrusion detection systems (NIDS) are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. It performs an analysis of passing traffic on the entire subnet, and matches the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behaviour is sensed, the alert can be sent to the administrator. An example of an NIDS would be installing it on the subnet where firewalls are located in order to see if someone is trying to break into the firewall. Ideally one would scan all inbound and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network. OPNET and NetSim are commonly used tools for simulating network intrusion detection systems. NID Systems are also capable of comparing signatures for similar packets to link and drop harmful detected packets which have a signature matching the records in the NIDS. When we classify the design of the NIDS according to the system interactivity property, there are two types: on-line and off-line NIDS, often referred to as inline and tap mode, respectively. On-line NIDS deals with the network in real time. It analyses the Ethernet packets and applies some rules, to decide if it is an attack or not. Off-line NIDS deals with stored data and passes it through some processes to decide if it is an attack or not.

Host intrusion detection systems

Host intrusion detection systems (HIDS) run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or

administrator if suspicious activity is detected. It takes a snapshot of existing system files and matches it to the previous snapshot. If the critical system files were modified or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission critical machines, which are not expected to change their configurations.

Detection method

Signature-based

Signature-based IDS refers to the detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware. This terminology originates from anti-virus software, which refers to these detected patterns as signatures. Although signature-based IDS can easily detect known attacks, it is impossible to detect new attacks, for which no pattern is available.

Anomaly-based

Anomaly-based intrusion detection systems were primarily introduced to detect unknown attacks, in part due to the rapid development of malware. The basic approach is to use machine learning to create a model of trustworthy activity, and then compare new behavior against this model. Although this approach enables the detection of previously unknown attacks, it may suffer from false positives: previously unknown legitimate activity may also be classified as malicious.

Limitations

It is not uncommon for the number of real attacks to be far below the number of false-alarms. Number of real attacks is often so far below the number of false-alarms that the real attacks are often missed and ignored.

Many attacks are geared for specific versions of software that are usually outdated. A constantly changing library of signatures is needed to mitigate threats. Outdated signature databases can leave the IDS vulnerable to newer strategies.[15]

For signature-based IDS, there will be lag between a new threat discovery and its signature being applied to the IDS. During this lag time, the IDS will be unable to identify the threat.[14]

It cannot compensate for weak identification and authentication mechanisms or for weaknesses in network protocols. When an attacker gains access due to weak authentication mechanisms then IDS cannot prevent the adversary from any malpractice.

Encrypted packets are not processed by most intrusion detection devices. Therefore, the encrypted packet can allow an intrusion to the network that is undiscovered until more significant network intrusions have occurred.

Intrusion detection software provides information based on the network address that is associated with the IP packet that is sent into the network. This is beneficial if the network address contained in the IP packet is accurate. However, the address that is contained in the IP packet could be faked or scrambled.

Tear Drop Attack

As the name suggests, the Teardrop Attack works gradually by sending the fragmented packets to a target machine. It's a type of a denial-of-service (DoS) attack which overwhelms the target machine with the incomplete data so that the victim crashes down.

Single Point Of Failure

A single point of failure (SPOF) is a potential risk posed by a flaw in the design, implementation or configuration of a circuit or system in which one fault or malfunction causes an entire system to stop operating.

In a data center or other information technology (IT) environment, a single point of failure can compromise the availability of workloads – or the entire data center – depending on the location and interdependencies involved in the failure.

Consider a data center where a single server runs a single application. The underlying server hardware would present a single point of failure for the application's availability. If the server failed, the application would become unstable or crash entirely; preventing users from accessing the application, and possibly even resulting in some measure of data loss. In this situation, the use of server clustering technology would allow a duplicate copy of the application to run on a second physical server. If the first server failed, the second would take over to preserve access to the application and avoid the SPOF.

Consider another example where an array of servers is networked through a single network switch. The switch would present a single point of failure. If the switch failed (or simply disconnected from its power source), all of the servers connected to that switch would become inaccessible from the remainder of the network. For a large switch, this could render dozens of servers and their workloads inaccessible. Redundant switches and network connections can provide alternative network paths for interconnected servers if the original switch should fail, avoiding the SPOF.

It is the responsibility of the data center architect to identify and correct single points of failure that appear in the infrastructure's design. However, it's important to remember that the resiliency needed to overcome single points of failure carries a cost (e.g. the price of additional servers within a cluster or additional switches, network interfaces and cabling). Architects must weigh the need for each workload against the additional costs incurred to avoid each SPOF. In some cases, designers may determine that the cost to correct a SPOF is costlier than the benefits of the workloads at risk.

RAID

***RAID** is short for **r**edundant **a**rray of **i**ndependent **d**isks.*

Originally, the term RAID was defined as ***r**edundant **a**rray of **i**nexpensive **d**isks*, but now it usually refers to a ***r**edundant **a**rray of **i**ndependent **d**isks*. RAID storage uses multiple disks in order to provide fault tolerance, to improve overall performance, and to increase storage capacity in a system. This is in contrast with older storage devices that used only a single disk drive to store data.

RAID allows you to store the same data redundantly (in multiple paces) in a balanced way to improve overall performance. RAID disk drives are used frequently on servers but aren't generally necessary for personal computers.

How RAID Works

With RAID technology, data can be mirrored on one or more disks in the same array, so that if one disk fails, the data is preserved. Thanks to a technique known as striping (a technique for spreading data over multiple disk drives), RAID also offers the option of reading or writing to more than one disk at the same time in order to improve performance.

In this arrangement, sequential data is broken into segments which are sent to the various disks in the array, speeding up throughput. A typical RAID array uses multiple disks that appear to be a single device so it can provide more storage capacity than a single disk.

Standard RAID Levels

RAID devices use many different architectures, called levels, depending on the desired balance between performance and fault tolerance. RAID levels describe how data is distributed across the drives. Standard RAID levels include the following:

Level 0: Striped disk array without fault tolerance

Provides data striping (spreading out blocks of each file across multiple disk drives) but no redundancy. This improves performance but does not deliver fault tolerance. If one drive fails then all data in the array is lost.

Level 1: Mirroring and duplexing

Provides disk mirroring. Level 1 provides twice the read transaction rate of single disks and the same write transaction rate as single disks.

Level 2: Error-correcting coding

Not a typical implementation and rarely used, Level 2 stripes data at the bit level rather than the block level.

Level 3: Bit-interleaved parity

Provides byte-level striping with a dedicated parity disk. Level 3, which cannot service simultaneous multiple requests, also is rarely used.

Level 4: Dedicated parity drive

A commonly used implementation of RAID, Level 4 provides block-level striping (like Level 0) with a parity disk. If a data disk fails, the parity data is used to create a replacement disk. A disadvantage to Level 4 is that the parity disk can create write bottlenecks.

Level 5: Block interleaved distributed parity

Provides data striping at the byte level and also stripe error correction information. This results in excellent performance and good fault tolerance. Level 5 is one of the most popular implementations of RAID.

Level 6: Independent data disks with double parity

Provides block-level striping with parity data distributed across all disks.

Level 10: A stripe of mirrors

Not one of the original RAID levels, multiple RAID 1 mirrors are created, and a RAID 0 stripe is created over these.

Non-Standard RAID Levels

Some devices use more than one level in a hybrid or nested arrangement, and some vendors also offer non-standard proprietary RAID levels. Examples of non-standard RAID levels include the following:

Level 0+1: A Mirror of Stripes

Not one of the original RAID levels, two RAID 0 stripes are created, and a RAID 1 mirror is created over them. Used for both replicating and sharing data among disks.

Level 7

A trademark of Storage Computer Corporation that adds caching to Levels 3 or 4.

RAID 1E

A RAID 1 implementation with more than two disks. Data striping is combined with mirroring each written stripe to one of the remaining disks in the array.

RAID S

Also called *Parity RAID*, this is EMC Corporation's proprietary striped parity RAID system used in its Symmetrix storage systems.

RAID History and Alternative Storage Options

Before RAID devices became popular, most systems used a single drive to store data. This arrangement is sometimes referred to as a SLED (single large expensive disk). However, SLEDs have some drawbacks. First, they can create I/O bottlenecks because the data cannot be read from the disk quickly enough to keep up with the other components in a system, particularly the processor. Second, if a SLED fails, all the data is lost unless it has been recently backed up onto another disk or tape.

In 1987, three University of California, Berkeley, researchers -- David Patterson, Garth A. Gibson, and Randy Katz -- first defined the term RAID in a paper titled A Case for Redundant Arrays of Inexpensive Disks (RAID). They theorized that spreading data across multiple drives could improve system performance, lower costs and reduce power consumption while avoiding the potential reliability problems inherent in using inexpensive, and less reliable, disks. The paper also described the five original RAID levels.

Today, RAID technology is nearly ubiquitous among enterprise storage devices and is also found in many high-capacity consumer storage devices. However, some non-RAID storage options do exist. One alternative is JBOD(Just a Bunch of Drives). JBOD architecture utilizes multiple disks, but each disk in the device is addressed separately. JBOD provides increased storage capacity versus a single disk, but doesn't offer the same fault tolerance and performance benefits as RAID devices.

Another RAID alternative is concatenation or spanning. This is the practice of combining multiple disk drives so that they appear to be a single drive. Spanning increases the storage capacity of a drive; however, as with JBOD, spanning does not provide reliability or speed benefits.

RAID Is Not Data Backup

RAID should not be confused with data backup. Although some RAID levels do provide redundancy, experts advise utilizing a separate storage system for backup and disaster recovery purposes.

Setting Up a RAID Array

In order to set up a RAID array, you'll need a group of disk drives and either a software or a hardware controller. Software RAID runs directly on a server, utilizing server resources. As a result, it may cause some applications to run more slowly. Most server operating systems include some built-in RAID management capabilities.

You can also set up your own RAID array by adding a RAID controller to a server or a desktop PC. The RAID controller runs essentially the same software, but it uses its own processor instead of the system's CPU. Some less expensive "fake RAID" controllers provide RAID management software but don't have a separate processor.

Alternatively, you can purchase a pre-built RAID array from a storage vendor. These appliances generally include two RAID controllers and a group of disks in their own housing.

Using a RAID array is usually no different than using any other kind of primary storage. The RAID management will be handled by the hardware or software controller and is generally invisible to the end user.