VAAGDEVI ENGINEERING COLLEGE
P.O.BOLLIKUNTA, WARANGAL – 506 005 (T S)
BTECH IV YEAR I SEMESTER
CSE (Artificial Intelligence & Machine Learning)

SUBJECT: WEB SECURITY

UNIT-1

Web security refers to the range of tools, controls, and measures designed to establish and preserve database confidentiality, integrity, and availability.

Web Security is very important nowadays. Websites are always prone to security threats/risks. Web Security deals with the security of data over the internet/network or web or while it is being transferred to the internet. For e.g. when you are transferring data between client and server and you have to protect that data that security of data is your web security.

Hacking a Website may result in the theft of Important Customer Data, it may be the credit card information or the login details of a customer or it can be the destruction of one's business and propagation of illegal content to the users while somebody hacks your website they can either steal the important information of the customers or they can even propagate the illegal content to your users through your website so, therefore, security considerations are needed in the context of web security.

Security Threats:

A Threat is nothing but a possible event that can damage and harm an information system. Security Threat is defined as a risk that which, can potentially harm Computer systems & organizations. Whenever an Individual or an Organization creates a website, they are vulnerable to security attacks.

Security attacks are mainly aimed at stealing altering or destroying a piece of personal and confidential information, stealing the hard drive space, and illegally accessing passwords. So whenever the website you created is vulnerable to security attacks then the attacks are going to steal your data alter your data destroy your personal information see your confidential information and also it accessing your password.

Top Web Security Threats:

Web security threats are constantly emerging and evolving, but many threats consistently appear at the top of the list of web security threats. These include:

- Cross-site scripting (XSS)
- SQL Injection
- Phishing
- Ransomware
- Code Injection
- Viruses and worms
- Spyware
- Denial of Service

Security Consideration:

- Updated Software: You need to always update your software. Hackers may be aware of vulnerabilities in certain software, which are sometimes caused by bugs and can be used to damage your computer system and steal personal data. Older versions of software can become a gateway for hackers to enter your network. Software makers soon become aware of these vulnerabilities and will fix vulnerable or exposed areas. That's why It is mandatory to keep your software updated, It plays an important role in keeping your personal data secure.
- Beware of SQL Injection: SQL Injection is an attempt to manipulate your data or your database by inserting a rough code into your query. For e.g. somebody can send a query to your website and this query can be a rough code while it gets executed it can be used to manipulate your database such as change tables, modify or delete data or it can retrieve important information also so, one should be aware of the SQL injection attack.
- Cross-Site Scripting (XSS): XSS allows the attackers to insert client-side script into web pages. E.g. Submission of forms. It is a term used to describe a class of attacks that allow an attacker to inject client-side scripts into other users' browsers through a website. As the injected code enters the browser from the site, the code is reliable and can do things like sending the user's site authorization cookie to the attacker.
- Error Messages: You need to be very careful about error messages which are generated to give the information to the users while users access the website and some error messages are generated due to one or another reason and you should be very careful while providing the information to the users. For e.g. login attempt – If the user fails to login the error message should not let the user know which field is incorrect: Username or Password.
- Data Validation: Data validation is the proper testing of any input supplied by the user or application. It prevents improperly created data from entering the information system. Validation of data should be performed on both server-side and client-side. If we perform data validation on both sides that will give us the

authentication. Data validation should occur when data is received from an outside party, especially if the data is from untrusted sources.

- Password: Password provides the first line of defense against unauthorized access to your device and personal information. It is necessary to use a strong password. Hackers in many cases use sophisticated software that uses brute force to crack passwords. Passwords must be complex to protect against brute force. It is good to enforce password requirements such as a minimum of eight characters long must including uppercase letters, lowercase letters, special characters, and numerals.

Why is it important

By definition, a data breach is a failure to maintain the confidentiality of data in a database. How much harm a data breach inflicts on your enterprise depends on a number of consequences or factors:

Compromised intellectual property: Your intellectual property—trade secrets, inventions, and proprietary practices—may be critical to your ability to maintain a competitive advantage in your market. If that intellectual property is stolen or exposed, your competitive advantage may be difficult or impossible to maintain or recover.

Damage to brand reputation: Customers or partners may be unwilling to buy your products or services (or do business with your company) if they don't feel they can trust you to protect your data or theirs.

Business continuity (or lack thereof): Some business cannot continue to operate until a breach is resolved.

Fines or penalties for non-compliance: The financial impact for failing to comply with global regulationssuch as the Sarbanes-Oxley Act (SAO) or Payment Card Industry Data Security Standard (PCI DSS), industry-specific data privacy regulations such as HIPAA, or regional data privacy regulations, such as Europe's General Data Protection Regulation (GDPR) can be devastating, with fines in the worst cases exceeding several million dollars *per violation*.

Costs of repairing breaches and notifying customers: In addition to the cost of communicating a breach to customer, a breached organization must pay for forensic and investigative activities, crisis management, triage, repair of the affected systems, and more.

Risk Analysis:

A security risk assessment identifies, assesses, and implements key security controls in applications. It also focuses on preventing application security defects and vulnerabilities. Carrying out a risk assessment allows an organization to view the application portfolio holistically—from an attacker‘s perspective. It supports managers in making informed resource allocation, tooling, and security control implementation decisions. Thus, conducting an assessment is an integral part of an organization‘s risk management process..

How does a security risk assessment work?

Factors such as size, growth rate, resources, and asset portfolio affect the depth of risk assessment models. Organizations can carry out generalized assessments when experiencing budget or time constraints. However, generalized assessments don‘t necessarily provide the detailed mappings betweenassets, associated threats, identified risks, impact, and mitigating controls.

If generalized assessment results don‘t provide enough of a correlation between these areas, a more in-depth assessment is necessary.

The 4 steps of a successful security risk assessment model:

- **Identification**. Determine all critical assets of the technology infrastructure. Next, diagnose sensitivedata that is created, stored, or transmitted by these assets. Create a risk profile for each.

- **Assessment**. Administer an approach to assess the identified security risks for critical assets. After careful evaluation and assessment, determine how to effectively and efficiently allocate time and resources towards risk mitigation. The assessment approach or methodology must analyze the correlation between assets, threats, vulnerabilities, and mitigating controls.

- **Mitigation**. Define a mitigation approach and enforce security controls for each risk.

- **Prevention**. Implement tools and processes to minimize threats and vulnerabilities from occurring inyour firm‘s resources.

A comprehensive security assessment allows an organization to:

- Identify assets (e.g., network, servers, applications, data centers, tools, etc.) within the organization.
- Create risk profiles for each asset.
- Understand what data is stored, transmitted, and generated by these assets.
- Assess asset criticality regarding business operations. This includes the overall impact to revenue,reputation, and the likelihood of a firm‘s exploitation.
- Measure the risk ranking for assets and prioritize them for assessment.
- Apply mitigating controls for each asset based on assessment results.

Cryptography and Web Security:

Increasingly, systems that employ cryptographic techniques are used to control access to computer systems and to sign digital messages. Cryptographic systems have also been devised to allow the anonymous exchange of digital money and even to facilitate fair and unforgivable online voting.

Roles for Cryptography:

Security professionals have identified five different roles that encryption can play in modern information systems. In the interest of sharing a common terminology, each of these different roles isidentified by a specific keyword. The roles are:

Authentication:

Digital signatures can be used to identify a participant in a web transaction or the author of an email message; people who receive a message that is signed by a digital signature can use it to verify the identity of the signer. Digital signatures can be used in conjunction with passwords and biometrics or as an alternative to them.

Authorization:

Whereas authentication is used to determine the identity of a participant, authorization techniques are used to determine if that individual is authorized to engage in a particular transaction. Cryptographic techniques can be used to distribute a list of authorized users that is all but impossible to falsify.

Confidentiality:

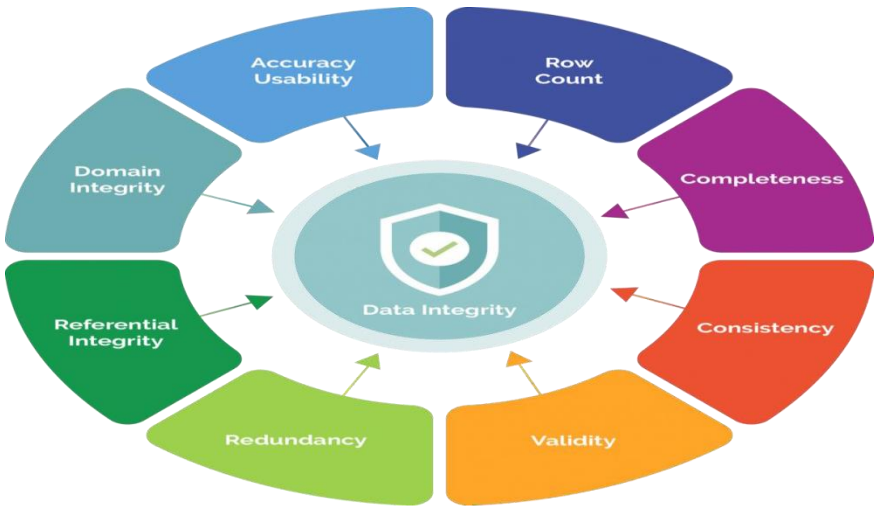
Encryption is used to scramble information sent over networks and stored on servers so that eavesdroppers cannot access the data‗s content. Some people call this quality ―privacy,‖ but most professionals reserve that word for referring to the protection of personal information (whether confidential or not) from aggregation and improper use.

Integrity:

Methods that are used to verify that a message has not been modified while in transit. Often, this isdone with digitally signed message digest codes.

Nonrepudiation:

Cryptographic receipts are created so that an author of a message cannot realistically deny sending a message (but see the discussion later in this section).Strictly speaking, there is some overlap among these areas. For example, when a message is encrypted to provide confidentiality, an unexpected byproduct is often integrity. That‗s because many encrypted messages will not decrypt if they are altered.


Accuracy Usability
Row Count
Domain Integrity
Completeness
Data Integrity
Referential Integrity
Consistency
Redundancy
Validity

WORKING CRYPTOGRAPHIC SYSTEMS AND PROTOCOLS:

A cryptographic system is a collection of software and hardware that can encrypt or decrypt information. A typical cryptographic system is the combination of a desktop computer, a web browser, a remote web server,and the computer on which the web server is running.

Cryptographic protocols and algorithms are difficult to get right, so do not create your own. Instead, where you can, use protocols and algorithms that are widely-used, heavily analyzed, and accepted as secure. When you must create anything, give the approach wide public review and make sure that professional security analysts examine it for problems. In particular, do not create your own encryption algorithms unless you are an expert in cryptology, know what you‗re doing, and plan to spend years in professional review of the algorithm. Creating encryption algorithms (that are any good) is a task for experts only.

A number of algorithms are patented; even if the owners permit ―free use‖ at the moment, without a signed contract they can always change their minds later, putting you at extreme risk later. In general, avoid all patented algorithms - in most cases there‗s an unpatented approach that is at least as good or better technically, and by doing so you avoid a large number of legal problems.

Often, your software should provide a way to reject ―too small‖ keys, and let the user set what ―too small‖ is. For RSA keys, 512 bits is too small for use. There is increasing evidence that 1024 bits for RSA keys is not enough either; Bernstein has suggested techniques that simplify brute-forcing RSA, and other work based on it (such as Shamir and Tromer‗s "Factoring Large Numbers with the TWIRL device") now suggests that 1024 bit keys can be broken in a year by a $10 Million device. You may want to make 2048 bits the minimum for RSA if you really want a secure system.

Cryptographic Protocols:

When you need a security protocol, try to use standard-conforming protocols such as IPSec, SSL (soon to be TLS), SSH, S/MIME, OpenPGP/GnuPG/PGP, and Kerberos. Each has advantages and disadvantages; many of them overlap somewhat in functionality, but each tends to be used in different areas:

- **Internet Protocol Security (IPSec)**. IPSec provides encryption and/or authentication at the IP packet level. However, IPSec is often used in a way that only guarantees authenticity of two communicating hosts, not of the users. As a practical matter, IPSec usually requires low-level support from the operating system (which not all implement) and an additional keyring server that must be configured. Since IPSec can be used as a "tunnel" to secure packets belonging to multiple users and multiple hosts, it is especially useful for building a Virtual Private Network (VPN) and connecting a remote machine. As of this time, it is much less often used to secure communication from individual clients to servers. The new version of the Internet Protocol, IPv6, comes withIPSec ―built in,‖ but IPSec also works with the more common IPv4 protocol. Note that if you use IPSec, don‗t use the encryption mode without the authentication, because the authentication also acts as integrity protection.
- **Secure Socket Layer (SSL) / TLS**. SSL/TLS works over TCP and tunnels other protocols using TCP, adding encryption, authentication of the server, and optional authentication of the client (but authenticating clients using SSL/TLS requires that clients have configured X.509 client certificates, something rarely done). SSL version 3 is widely used;

TLS is a later adjustment to SSL that strengthens its security and improves its flexibility. Currently there is a slow transition going on from SSLv3 to TLS, aided because implementations can easily try to use TLS and then back off to SSLv3 without user intervention.

Unfortunately, a few bad SSLv3 implementations cause problems with the backoff, so you may need a preferences setting to allow users to skip using TLS if necessary. Don't use SSL version 2, it has some serious security weaknesses.

SSL/TLS is the primary method for protecting http (web) transactions. Any time you use an https:// URL, you're using SSL/TLS. Other protocols that often use SSL/TLS include POP3 and IMAP. SSL/TLS usually use a separate TCP/IP port number from the unsecured port, which the IETF is a little unhappy about (because it consumes twice as many ports; there are solutions to this). SSL is relatively easy to use in programs, because most library implementations allow programmers to use operations similar to the operations on standard sockets like SSL_connect(), SSL_write(), SSL_read(), etc. A widely used OSS/FS imple~~mentation of SSL (as~~ well as other capabilities) is OpenSSL, available at http://www.openssl.org.

- **OpenPGP and S/MIME**. There are two competing, essentially incompatible standards for securing email: OpenPGP and S/MIME. OpenPHP is based on the PGP application; an OSS/FSimplementation is GNU Privacy ~~Guard from http://~~www.gnupg.org. Currently, their certificates are often not interchangeable; work is ongoing to repair this.

- **SSH**. SSH is the primary method of securing -remote terminals‖ over an internet, and it also includes methods for tunelling X Windows sessions. However, it's been extended to support single sign-on and general secure tunelling for TCP streams, so it's often used for securing other data streams too (such as CVS accesses). The most popular implementation of SSH is OpenSSH http://www.openssh.com, which is OSS/FS. Typical uses of SSH allows the client to authenticate that the server is truly the server, and then the user enters a password to authenticate the user (the password is encrypted and sent to the other system for verification). Current versions ofSSH can store private keys, allowing users to not enter the password each time. To prevent man-in- the-middle attacks, SSH records keying information about servers it talks to; that means that typical use of SSH is vulnerable to a man-in-the-middle attack during the very first connection, but it can detect problems afterwards. In contrast, SSL generally uses a certificate authority, which eliminates the first connection problem but requires special setup (and payment!) to the certificate authority.

- **Kerberos**. Kerberos is a protocol for single sign-on and authenticating users against a central authentication and key distribution server. Kerberos works by giving authenticated users "tickets", granting them access to various services on the network. When clients then contact servers, the servers can verify the tickets. Kerberos is a primary method for securing and supporting authentication on a LAN, and for establishing shared secrets (thus, it needs to be used with other algorithms for the actual protection of communication). Note that to use Kerberos, both the client and server have to include code to use it, and since not everyone has a Kerberos setup, this has to be optional - complicating the use of Kerberos in some programs. However, Kerberos is widely used.

AES Standards	**Key Size (in bits)**	**Block Size (in bits)**	**Number of Rounds**

AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

Digital Signature Algorithm:

Digital Signature Algorithm (DSA) is also a public key algorithm that is used only for digitally signing documents. This scheme is suitable for achieving authentication before a message or documents are shared (Forouzan, 2011). Receiving a digitally signed document, the recipient becomes confident that the sender was a genuine one and the document was not altered during the transmission. Digital signatures are applied in software distribution, financial transactions, and for documents that might be tampered with. To verify the document the receiver performs the following steps:

1. Decrypts the digital signature using the sender's public key to read the message.
2. Generates a message digest for the receiver's message using the same algorithm used by the sender.
3. If both message digests do not match, the sender's message digest is considered to be compromised.

Hash functions:

Hash functions or one-way functions are used in public-key cryptography for implementing protocols (Alawida et al., 2021). Hash functions do not need any key. They are easily computable but harder to reverse. For example, f(x) can be computed easily but the computation of x from f(x) will take many years even for all the computers of the world collectively. The value of f(x) is a fixed-length hash value computed out of x which is the plaintext. Neither the contents of the plaintext nor its length can be obtained. Hash functions are used to verify the integrity of the documents and encryption of passwords. Even a small bit of change in the contents can be easily detected because the hash values of the two versions will be absolutely different.

Cryptographic protocol:

Cryptography analyses the issues of integrity, authentication, privacy, and Nonrepudiation. Cryptographic algorithms are having academic importance (Schneier, 2007). Application of these algorithms alone cannot guarantee to achieve the goal of Cryptography. Well-defined policies and agreements between the parties involved in the communication are also required in order to make Cryptography a reliable technology for achieving its goals so that it can solve real problems in completing online tasks between trusted parties.
A cryptographic protocol is a distributed algorithm designed to precisely describe the interactions between two or more parties with the objective of implementing certain security policies. It follows some series of steps in exact sequence. Every step must be completely executed without any alteration in the agreed-uponsequence. It must be complete and able to finish a task. At least two parties are required. Any single party executing a series of steps to complete a task is not a protocol. Every party must know, understand, and follow it. They must not be able to do something beyond the specified agreement.

Arbitrated Protocols:

Arbitrated protocols use a trusted third party called an arbitrator. The arbitrator has no vested interest and cannot favor any of the involved parties. Such protocols are used to complete tasks between two or more parties not trusting each other.

Adjudicated Protocols:

The arbitrated protocols are implemented with two sub protocols to reduce the cost of third-party involvement. Some non-arbitrated protocol is used in the first level which is executed for each task. In the second level, an arbitrated protocol is used which is executed only in case of disputes occur between the involved parties during the task.

Self-Enforcing Protocols:

These protocols require no arbitrator to complete tasks or to resolve disputes. The protocol itself ensures thatthere is no dispute between the involved parties. One party can detect whenever the other party is trying to play smart and the task is stopped immediately. It is ideal that every protocol should be self-enforcing.
Similar to the attacks on Cryptographic algorithms and techniques, protocols can also be attacked by the cheaters.

Types of Protocols:

Key Exchange Protocols:

A key exchange protocol is required for two parties to reach an agreement for a shared secret key. Either oneparty can authenticate the other or both parties can authenticate each other. The protocol can agree for the generation of a random key. One party can generate the key and send it to another party or both parties can participate in the key generation.

Diffie-Hellman key exchange:

This protocol is used by the involved parties to agree on a shared key by exchanging messages through a public channel. Therefore, the key is not revealed to any unauthorized party. This is protected only against passive attacks.

Identification and Authentication Protocols:

Identification protocols are required to ensure the identity of both parties when they are online for a task. Genuine possession of their private keys needs to be verified. The level of identification by the protocols may be judged with three levels: (1) Who is he? – Biometrics is used, (2) What he possesses? –Some hardware gadgets can be used, (3) What he knows? – Secret keys or passwords are used. Some popular protocols are zero-knowledge protocol, Schnorr Protocol, Guillou-Quisquater protocol, witness hiding identification protocols, etc.

Using Password Authentication:

In absence of any digital signature scheme the two parties can share a password that is comparatively lesspowerful.

Issues in Cryptography

In symmetric cryptography, if the key is lost, communication cannot be completed. This creates an issue of secure key distribution with possibly involving either the sender and the receiver to communicate directly or via a trusted third party or communicating via an existing cryptographic medium (Sharma et al., 2021). The issue of key distribution is to be dealt with delicately: keys must be stored, used, as well as destroyed securely.

Cryptography only transforms plaintext but never hides it (Rahmani et al., 2014). One weakness of Cryptography is if somehow any third party detects the presence of an encrypted message, it can make attempts to break into it out of curiosity. Sometimes curiosity feeds the cat. As a consequence, it can reveal the secrecy, modify or misuse the information.

Legal Restrictions on Cryptography:

The legal landscape of cryptography is complex and constantly changing. In recent years the legal restrictions on cryptography in the United States have largely eased, while the restrictions in other countries have increased somewhat.

Cryptography and the Patent System:

Patents applied to computer programs, frequently called *software patents*, have been accepted by thecomputer industry over the past thirty years-some grudgingly, and some with great zeal.

The *doctrine of equivalence* holds that if a new device operates in substantially the same way as a patented device and produces substantially the same result, then the new device infringes the original patent. As a result of this doctrine, which is one of the foundation principles of patent law, a program that implements a patented encryption technique will violate that patent, even if the original patent was on a machine built from discrete resistors, transistors, and other components. Thus, the advent of computers that were fast enough to implement basic logic circuits in software, combined with the acceptance of patent law and patents on electronic devices, assured that computer programs would also be the subject of patent law.

The outlook for patents:

Although new encryption algorithms that are protected by patents will continue to be invented, a wide variety of unpatented encryption algorithms now exist that are secure, fast, and widely accepted. Furthermore, there is no cryptographic operation in the field of Internet commerce that requires the use of a patented algorithm. As a result, it appears that the overwhelming influence of patent law in the fields of cryptography and e-commerce have finally come to an end.

Cryptography and Trade Secret Law:

Until very recently, many nontechnical business leaders mistakenly believed that they could achieve additional security for their encrypted data by keeping the encryption algorithms themselves secret. Many companies boasted that their products featured *proprietary encryption algorithms.* These companies refused to publish their algorithms, saying that publication would weaken the security enjoyed by their users.

Today, most security professionals agree that this rationale for keeping an encryption algorithm secret is largely incorrect. That is, keeping an encryption algorithm secret does not significantly improve the security that the algorithm affords. Indeed, in many cases, secrecy actually decreases the overall security of an encryption algorithm.

There is a growing trend toward academic discourse on the topic of cryptographic algorithms. Significant algorithms that are published are routinely studied, analyzed, and occasionally found to be lacking. As a result of this process, many algorithms that were once trusted have been shown to have flaws. At the same time, a few algorithms have survived the rigorous process of academic analysis. Some companies think they can short-circuit this review process by keeping their algorithms secret. Other companies have used algorithms that were secret but widely licensed as an attempt to gain market share and control. But experience has shown that it is nearly impossible to keep the details of a successful encryption algorithm secret. If the algorithm is widely used, then it will ultimately be distributed in a form that can be analyzed and reverse-engineered.

Regulation of Cryptography by International and National Law:

In the past 50 years there has been a growing consensus among many governments of the world on the need to regulate cryptographic technology. The original motivation for regulation was military. During World War II, the ability to decipher the Nazi Enigma machine gave the Allied forces a tremendous advantage-Sir Harry Hinsley estimated that the Allied "ULTRA" project shortened the war in the Atlantic, Mediterranean, and Europe "by not less than two years and probably by four years."As a result of this experience,
military intelligence officials in the United States and the United Kingdom decided that they needed to control the spread of strong encryption technology-lest these countries find themselves in a future war in which they could not eavesdrop on the enemy's communications.

U.S. regulatory efforts and history:

Export controls in the United States are enforced through the Defense Trade Regulations (formerly known as the International Traffic in Arms Regulation-ITAR). In the 1980s, any company wishing to export a machine or program that included cryptography needed a license from the U.S. government. Obtaining a license could be a difficult, costly, and time consuming process.

Following the 1992 compromise, the Clinton Administration made a series of proposals designed to allow consumers the ability to use full-strength cryptography to secure their communications and stored data, while still providing government officials relatively easy access to the plaintext, or the unencrypted data. These proposals were all based on a technique called *key escrow*. The first of these proposals was the administration's Escrowed Encryption Standard (EES), more commonly known as the Clipper chip.

In 1997, an ad hoc group of technologists and cryptographers issued a report detailing a number of specific risks regarding all of the proposed key recovery, key escrow, and trusted third-party encryption schemes:

*The potential for insider abuse-*There is fundamentally no way to prevent the compromise of the system "by authorized individuals who abuse or misuse their positions. Users of a key recovery system must trust that the individuals designing, implementing, and running the key recovery operation are indeed trustworthy. An individual, or set of individuals, motivated by ideology, greed, or the threat of blackmail, may abuse the authority given to them.

The creation of new vulnerabilities and targets for attack: Securing a communications or data storage system is hard work; the key recovery systems proposed by the government would make the job of security significantly harder because more systems would need to be secured to provide the same level of security.

*Scaling might prevent the system from working at all-*The envisioned key recovery system would have to work with thousands of products from hundreds of vendors; it would have to work with key recovery agents all over the world; it would have to accommodate tens of thousands of law enforcement agencies, tens of millions of public-private key pairs, and hundreds of billions of recoverable session keys.

The difficulty of properly authenticating requests for keys- A functioning key recovery system would deal with hundreds of requests for keys every week coming from many difficult sources. How could all these requests be properly authenticated?

*The cost-*Operating a key recovery system would be incredibly expensive. These costs include the cost of designing products, engineering the key recovery center itself, actual operation costs of the center, and (we hope) government oversight costs. Invariably, these costs would be passed along to the end users, who would be further saddled with "both the expense of choosing, using, and managing key recovery systems and the losses from lessened security and mistaken or fraudulent disclosures of sensitive data."

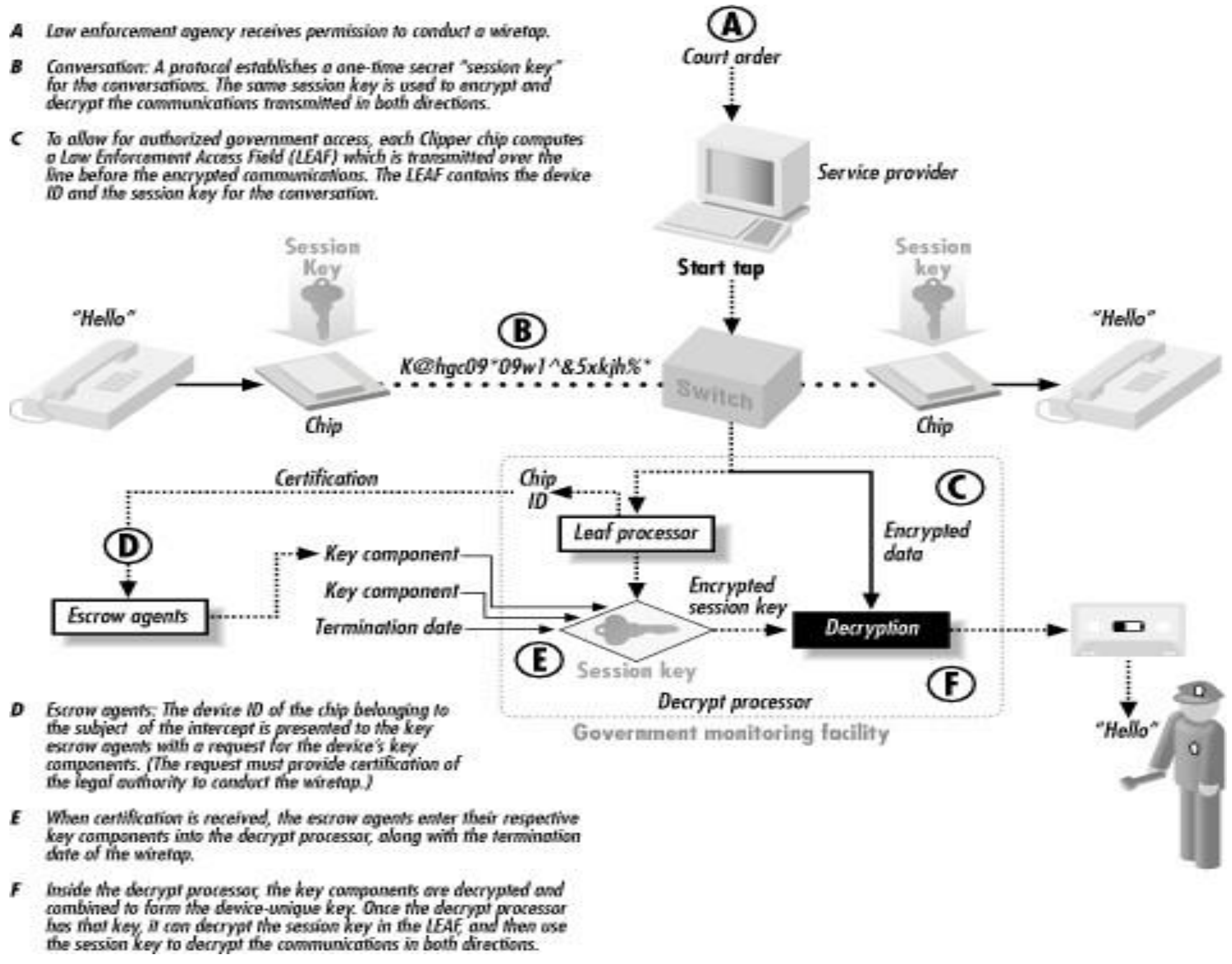
The Digital Millennium Copyright Act:

There is a huge and growing market in digital media. Pictures, e-books, music files, movies, and more represent great effort and great value. Markets for these items are projected to be in the billions of dollars per year. The Internet presents a great medium for the transmission, rental, sale, and display of this media. However, because the bits making up these items can be copied repeatedly, it is possible that fraud and theft can be committed easily by anyone with access to a copy of a digital item.

International agreements on cryptography:

International agreements on the control of cryptographic software date back to the days of COCOM (Coordinating Committee for Multilateral Export Controls), an international organization created to control the export and spread of military and dual-use products and technical data.

The Council of Europe is a 41-member intergovernmental organization that deals with policy matters in Europe not directly applicable to national law. On September 11, 1995, the Council of Europe adopted Recommendation R (95) 13 Concerning Problems of Criminal Procedure Law Connected with Information Technology, which stated, in part, that "measures should be considered to minimize the negative effects of the use of cryptography on the investigation of criminal offenses, without affecting its legitimate use more than is strictly necessary."



A *Law enforcement agency receives permission to conduct a wiretap.*

B *Conversation: A protocol establishes a one-time secret "session key" for the conversations. The same session key is used to encrypt and decrypt the communications transmitted in both directions.*

C *To allow for authorized government access, each Clipper chip computes a Law Enforcement Access Field (LEAF) which is transmitted over the line before the encrypted communications. The LEAF contains the device ID and the session key for the conversation.*

D *Escrow agents: The device ID of the chip belonging to the subject of the intercept is presented to the key escrow agents with a request for the device's key components. (The request must provide certification of the legal authority to conduct the wiretap.)*

E *When certification is received, the escrow agents enter their respective key components into the decrypt processor, along with the termination date of the wiretap.*

F *Inside the decrypt processor, the key components are decrypted and combined to form the device-unique key. Once the decrypt processor has that key, it can decrypt the session key in the LEAF, and then use the session key to decrypt the communications in both directions.*

\

UNIT-II

PRIVACY AND PRIVACY PROTECTING TECHNIQUES:

Choosing a Good Service Provider:

The first and most important technique for protecting your privacy is to pick service providers who respectyour privacy. Here are some things to consider when you choose an ISP:

- Unless you take special measures to obscure the content and destinations of your Internet usage, your ISPcan monitor every single web page that you visit, every email message that you send, every email message that you receive, and many other things about your Internet usage.
- If you have a dialup ISP, your ISP can also infer when you are at home, when you go on vacation, andother aspects of your schedule.
- If you check your email from work, your ISP can learn where you work.
- Many ISPs routinely monitor the actions of their subscribers for the purposes of testing equipment, learning about their user population, or collecting per-user demographics.

Picking a Great Password:

Passwords are the simplest form of authentication. Passwords are a secret that you share with the computer. When you log in, you type your password to prove to the computer that you are who you claim to be. The computer ensures that the password you type matches the account that you have specified. If they match, you are allowed to proceed. Using good passwords for your Internet services is a first line of defense foryour privacy. If you pick a password that is easy to guess, then somebody who is targeting you will find
it easier to gain access to your personal information. If you use the same password on a variety of different services, then a person who is able to discover the password for one of your services will be able to access other services.

Cleaning Up After Yourself:

When you use the Internet, you leave traces of the web sites that you visit and the information that you seeon your computer. Another person can learn a lot about the web sites that you have visited by examining your computer for these electronic footprints. This process of computer examination is called *computerforensics*, and it has become a hot area of research in recent years. Special-purpose programs can alsoexamine your computer and either prepare a report, or transmit the report over the Internet to someone else. Although it can be very hard to remove all traces of a web site that you have seen or an email message that you have downloaded, you can do a good job of cleaning up your computer with only a small amount ofwork. There are also a growing number of programs that can automatically clean up your computer atregular intervals.

Browser Cache:

Each time your web browser attempts to fetch a URL from the Internet, it first checks a special directory called the *browser cache* to see if it has a copy of the information that it is about to download. The browser cache can dramatically speed up the process of web browsing. For example, if you are clicking through a web site that has lots of little buttons, each of the buttons only needs to be downloaded once, even though the same buttons might appear on every page. But if another person can gain access to your computer, the browser cache will provide that person with copies of the web pages that you have visited. There are several ways to minimize the privacy impact of your browser's cache:

- You can usually configure your browser so that HTML pages and images downloaded by SSL are not cached on your hard disk. This way, confidential information such as bank statements and credit card numbers will not be locally stored. This is the default for many web browsers.
- You can configure your browser so that no HTML pages or images are cached. Although this approach will give you the maximum amount of privacy, it will also significantly decrease your browsing performance.
- You can manually delete the information in your browser's cache when you leave a particularly sensitive site. Alternatively, you can inspect the browser's cache and delete the specific material that you believe to be sensitive.

Avoiding Spam and Junk Email:

Unwanted electronic mail is the number one consumer complaint on the Internet today. A 1999 study by BrightMail, a company that develops antispam technology, found that 84 percent of Internet users had received spam; 42 percent loathed the time it takes to handle spam; 30 percent found it to be a "significant invasion of privacy;" 15 percent found it offensive; and ISPs suffered account churn rates as high as 7.2 percent as a direct result of spam.

Identity Theft:

In 1991, a car salesman from Orlando, Florida named Steven Shaw obtained the credit report for a journalist in Washington named, coincidently enough, Stephen Shaw. For Steven Shaw, getting Stephen Shaw's credit report was easier than you might think: for years, the consumer reporting firm Equifax had aggressively marketed its credit reporting service to car dealers. The service lets salespeople weed out the Sunday window-shoppers from the serious prospects by asking for a customer's name and then surreptitiously disappearing into the back room and running a quick credit check. In all likelihood, the Shaw in Florida had simply gone fishing for someone with a similar-sounding name and a good credit history.

Protecting Yourself From Identity Theft:

Identity theft is made possible because companies that grant credit-especially credit card companies-are always on the lookout for new customers, and they don't have a good way to verify the identity of a person who mails in an application or places an order over the telephone. So the credit-granting companies make a dangerous assumption: they take it for granted that if you know a person's name, address, telephone number, Social Security number, and mother's maiden name, *you must be that person*. And when the merchandise is bought and the bills aren't paid, that person is the one held responsible.

BACKUPS AND ANTITHEFT, WEB SERVER SECURITY:

Using Backups to Protect Your Data:

Backups are copies that you make of information that you hope you will never need. A backup can be a simple copy of a file that you put on a Zip disk and put away in the top drawer of your desk for safekeeping. If the original file is inadvertently deleted or corrupted, the backup can be retrieved after the damage is noticed. Backups can be very simple, like the Zip disk in your desk drawer, or they can be exceedingly complex. For example, many backup systems will let you copy every file on your computer onto a 30- gigabyte magnetic tape and create a special "restore floppy." In the event that your computer is lost or stolen, you can buy a new computer and a tape drive, put the tape into the tape drive, and boot the computer from the floppy disk; the backup system will automatically restore all of your files and applications to the newly purchased computer.

Make Backups:

Bugs, accidents, natural disasters, and attacks on your system cannot be predicted. Often, despite your best efforts, they can't be prevented. But if you have good backups, you at least won't lose your data-and in many cases, you'll be able to restore your system to a stable state. Even if you lose your entire computer-to fire, for instance-with a good set of backups you can restore the information after you purchase or borrow a replacement machine. Insurance can cover the cost of a new CPU and disk drive, but your data is something that in many cases can never be replaced.

WEB SERVER SECURITY:

PHYSICAL SECURITY OF THE SERVERS:

Surprisingly, many organizations do not consider physical security to be of the utmost concern. As an example, one New York investment house was spending tens of thousands of dollars on computer security measures to prevent break-ins during the day, only to discover that its cleaning staff was propping open the doors to the computer room at night while the floor was being mopped. A magazine in San Francisco had more than $100,000 worth of computers stolen over a holiday: an employee had used his electronic key card to unlock the building and disarm the alarm system; after getting inside, the person went to the supply closet where the alarm system was located and removed the paper log from the alarm system's printer.

The Physical Security Plan:

The first step to physically securing your installation is to formulate a written plan addressing your current physical security needs and your intended future direction. Ideally, your physical plan should be part of your site's written security policy. This plan should be reviewed by others for completeness, and it should be approved by your organization's senior management. Thus, the purpose of the plan is both planning and political buy-in. Your security plan should include:

- Descriptions of the physical assets that you are protecting
- Descriptions of the physical areas where the assets are located
- A description of your *security perimeter*- the boundary between the rest of the world and your securedarea-and the holes in the perimeter
- The threats (e.g., attacks, accidents, or natural disasters) that you are protecting against and theirlikelihood
- Your security defenses, and ways of improving them
- The estimated cost of specific improvements
- The value of the information that you are protecting

The Disaster Recovery Plan:

You should have a plan for immediately securing temporary computer equipment and for loading your backups onto new systems in case your computer is ever stolen or damaged. This plan is known as a *disasterrecovery plan*. We recommend that you do the following:

- Establish a plan for rapidly acquiring new equipment in the event of theft, fire, or equipment failure.
- Test this plan by renting (or borrowing) a computer system and trying to restore your backups.

If you ask, you may discover that your computer dealer is willing to lend you a system that is faster than the original system for the purpose of evaluation. There is probably no better way to evaluate a system than to load your backup tapes onto the system and see if they work.

Protecting Computer Hardware:

Physically protecting a computer presents many of the same problems that arise when protecting typewriters, jewelry, and file cabinets. As with a typewriter, an office computer is something that many people inside the office need to access on an ongoing basis. As with jewelry, computers are valuable and generally easy for a thief to sell. But the real danger in having a computer stolen isn't the loss of the system's hardware but the loss of the data that was stored on the computer's disks. As with legal files and financial records, if you don't have a backup-or if the backup is stolen or destroyed along with the computer-the data you have lost may well be irreplaceable. Even if you do have a backup, you will still need to spend valuable time setting up a replacement system. Finally, there is always the chance that the stolen information itself, or even the mere fact that information was stolen, will be used against you.

There are several measures that you can take to protect your computer system against physical threats. Many of them will simultaneously protect the system from dangers posed by nature, outsiders, and inside saboteurs.

- Environment
- Fire
- Smoke
- Dust
- Earthquake
- Explosion
- Temperature extremes
- Bugs (Biological)
- Electrical Noise
- Lightning
- Vibration
- Humidity
- Water
- Environmental monitoring

Host Security for Servers:

Host security is the security of the computer on which your web server is running. Traditionally, host security has been a computer security topic unto itself. Whole books (including a couple of our own) have been written on it. Many organizations that run servers on the Internet simply do not secure their servers against external attack. Other problems have gotten worse: people still pick easy-to-guess passwords, and many passwords are simply "sniffed" out of the Internet using a variety of readily available packet sniffers. And people still break into computers for the thrill, except that now many of them also steal information for financial gain or to make some ideological point.

Taxonomy of Attacks:

They are now simply one of many venues for an attacker to gain access and control over a targetcomputer system. Many of these techniques give the attacker the ability to run code on the targetmachine. These techniques include:

Remote Exploits: Vulnerabilities exist in many computers that make it possible for an attacker to compromise, penetrate, or simply disable the system over the network without actually logging into the system. For example, Microsoft Windows NT 4.0 was vulnerable to the *ping of death*, which allowed anybody on the Internet to crash a Windows NT 4.0 system by simply sending the computer a specially crafted "ping" packet.
Many remote exploits are based on the *buffer overflow* technique. This technique relies on the way that the C programming language lays out information inside the computer's memory. The remote system might try to store 100 bytes into a buffer that is only set up to hold 30 or 40 bytes. The resulting information overwrites the C program's stack frame and causes machine code specified by the attacker to be executed.

Malicious programs: Another way for an attacker to compromise a system is to provide the system's users with a hostile program and wait for them to run the program. Some programs when run will install hidden services that give attackers remote access capabilities to the compromised machine; these programs are called *back doors* because they offer attackers away into the system that bypasses conventional security measures. *Trojan horses* are programs that appear to have one function but actually have another purpose that ismalicious, similar to the great wooden horse that the Greeks allegedly used to trick theTrojans and end the siege of Troy.

Viruses and worms are self-replicating programs that can travel between computers as attachments on email or independently over a network. *Viruses* modify programs on your computer, adding to them their viral *payload*. *Worms* don't modify existing programs, but they can install back doors or drop viruses on the systems they visit.

Stolen usernames and passwords and social engineering: On many computer systems it is possible to exploit bugs or other vulnerabilities to parlay ordinary access granted to normal users into "super user" or "administrative" access that is granted to system operators. One of the most common ways for an attacker to get a username and password is *social engineering.*Social engineering is one of the simplest and most effective means of gaining unauthorized access to a computer system.

Phishing: Social engineering can also be automated. There are many so-called *phishing* Programs that will send social engineering emails to thousands or tens of thousands of users at a time. Some programs solicit usernames and passwords. Others try for valid credit cards.For example, one scam is to send email to users of an online service telling them that their credit cards have expired and that they need to enter a new one at the URL that is provided. Of course, the URL goes to the attacker's web server, not to the ISP's.

Understanding Your Adversaries:

Who is breaking into networked computers with the most sophisticated of attacks? It almost doesn'tmatter-no matter whom the attackers may be, they all need to be guarded against.

Script kiddies: As clichéd as it may sound, in many cases the attackers are children and teenagers-people who sadly have not (yet) developed the morals or sense of responsibility that is sufficient to keep their technical skills in check. It is common to refer to young people who use sophisticated attack tools as *script kiddies*. Script kiddies should be considered a serious threat and feared for the same reason that teenagers with guns should be respected and feared. We don't call gangbangers *gun kiddies* simply because youthful gang members don't have the technical acumen to design a Colt 45 revolver or cast the steel. Instead, most people realize that teenagers with handguns should be feared even more than adults, because a teenager is less likely to understand the consequences of his actions should he pull the trigger and thus more likely to pull it.

Industrial spies: There appears to be a growing black market for information stolen from computer systems. Some individuals have tried to ransom or extort the information from its rightful owners for example, by offering to help a company close its vulnerabilities in exchange for a large cash payment. There have also been reports of attackers who have tried to sell industrial secrets to competitors of the companies that they have penetrated. Suchtransactions are illegal in the United States and in many other countries, but not in all.

Ideologues and national agents: There is a small but growing population of "hacktivists" who break into sites for ideologic or political reasons. Often, the intent of these people is to deface web pages to make a statement of some kind. We have seen cases of defacement of law enforcement agencies, destruction of web sites by environmental groups, and destruction of research computing sites involving animal studies. Sometimes, the protesters are making a political statement; they may be advancing an ideologic cause, or they may merely be anarchists striking a blow against technology or business.

Organized crime: The apocryphal quote by Willie Sutton about why he robbed banks, "Because that's where the money is," also applies on the Net. Vast amounts of valuable information and financial data flow through the Internet. It would be naive to believe that thecriminal element is unaware of this, or is uninterested in expanding into the networked world. There have been incidents of fraud, information piracy, and money laundering conducted online that officials believe are related to organized crime. Communications on the Net have been used to advance and coordinate prostitution and pornography, gambling, trafficking in illegal substances, gun running, and other activities commonly involving organized crime.
Furthermore, law enforcement sites may be targeted by criminals to discover what isknown about them, or to discover identities of informants and witnesses.

Rogue employees and insurance fraud: Finally, there are many cases of tactically skilled employees who have turned against their employers out of revenge, malice, or boredom. In some cases, terminated employees have planted Trojan horses or logic bombs in their employer's computers. In other cases, computer systems have been destroyed by employeesas part of insurance scams.

Tools of the Attacker's Trade: Tools that are commonly used by attackers include:

Originally written by "Hobbit," *netcat* is the Swiss Army knife for IP-based networks. As such, it is avaluable diagnostic and administrative tool as well as useful to attackers. You can use *netcat* to send arbitrary data to arbitrary TCP/IP ports on remote computers, to set up local TCP/IP servers, and to perform rudimentary port scans.

Trinoo: *trinoo* is the attack server that was originally written by the DoS Project. *trinoo* waits for a message from a remote system and, upon receiving the message, launches a denial-of- service attack against a third party. Versions of *trinoo* are available for most Unix operating systems, including Solaris and Red Hat Linux. The presence of *trinoo* is usually hidden. A detailed analysis of *trinoo* can be found at http://staff.washington.edu/dittrich/misc/trinoo.analysis.

Back Orifice and Netbus: These Windows-based programs are Trojan horses that allow an attacker to remotely monitor keystrokes, access files, upload and download programs, and run programs on compromised systems.

Bots: Short for robots, *bots* are small programs that are typically planted by an attacker on a collection of computers scattered around the Internet. Bots are one of the primary tools for conducting distributed denial-of-service attacks and for maintaining control on Internet Relay Chat channels. Bots can be distributed by viruses or Trojan horses. They can remain dormant for days, weeks, or years until they are activated. Bots can even engage in autonomous actions.

*root kits:*A *root kit* is a program or collection of programs that simultaneously gives the attacker superuser privileges on a computer, plants back doors on the computer, anderases any trace that the attacker has been present. Originally, root kits were designed for Unix systems (hence the name "root"), but root kits have been developed for Windows systems as well. A typical root kit might attempt a dozen or so different exploits to obtain superuser privileges. Once superuser privileges are achieved, the root kit might patch the *login* programto add a back door, then modify the computer's kernel so that any attempt to read the *login* program returns the original, unmodified program, rather than the modified one. The *netstat* command might be modified so that network connections from the attacker's machine are notdisplayed. Finally, the root kit might then erase the last five minutes of the computer's log files.

Securing Web Applications:

Web servers are fine programs for displaying static information such as brochures, FAQs, and product catalogs. But applications that are customized for the user or that implement business logic (such as shopping carts) require that servers be extended with specialized code that executes each time the web page is fetched. This code most often takes the form of *scripts* or *programs* that are run when a particular URL is accessed. There is no limit to what a good programming team can do with a web server, a programming language, and enough time. Unfortunately, programs that provideadditional functionality over the Web can have flaws that allow attackers to compromise the system on which the web server is running. These flaws are rarely evident when the program is run as intended.

A Legacy of Extensibility and Risk: There are four primary techniques that web developers can useto create web-based applications:

CGI:
The Common Gateway Interface (CGI) was the first means of extending web servers. When a URL referencing a CGI program is requested from the web server, the web server runs the CGI program in a separate process,
Captures the program's output, and sends the results to the requesting web browser. Parameters to the CGI programs are encoded as environment variables and also provided to the program on standard input.

Plug-ins, loadable modules, and Application Programmer Interfaces (APIs):
The second technique developed to extend web servers involved modifying the web server with extension modules, usually written in C or C++. The extension module was then loaded into the web server at runtime.
Plug-ins, modules, and APIs are a faster way to interface custom programs to web servers because they do not require that a new process be started for each web interaction. Instead, the web server process itself runs applicationcode within its own address space that is invoked through a documented interface. But these techniques have a distinct disadvantage: the plug-in code can be very difficult to write, and a single bug can cause the entire webserver to crash.

Embedded scripting languages:
Web-based scripting languages were the third technique developed for adding programmatic functionality to web pages. These systems allow developers to place small programs, usually called scripts, directly into the web
page. An interpreter embedded in the web server runs the program contained on the web page before the resulting code is sent to the web browser. Embedded scripts tend to be quite fast. Microsoft's ASP, PHP, serverside JavaScript, and *mod_perl* are all examples of embedded scripting languages.

Embedded web server:
Finally, some systems do away with the web server completely and embed their own HTTP server into the webapplication itself.

Rules to Code By:

Most security-related bugs in computer programs are simply that: bugs. For whatever reason, thesefaults keep your program from operating properly.

General Principles for Writing Secure Scripts: Over the years, we have developed a list of generalprinciples by which to code. What follows is an excerpt from that list, edited for its particular relevance to CGI and API programs:

1. Carefully design the program before you start. Be certain that you understand what you are trying to build. Carefully consider the environment in which it will run, the input and output behavior, files used, arguments recognized, signals caught, and other aspects of behavior. List all of the errors that might occur, and how your program will deal with them. Write a code specification in English (or your native language) before writing the code in the computer language of your choice.

2. Show the specification to another person. Before you start writing code, show the

specification that you have written to another programmer. Make sure they can understand the specification and that they think it will work. If you can't convince another programmer that your paper design will work, you should go back to the design phase and make your specification clearer. The time you spend now will be repaid many times over in the future.

3. Write and test small sections at a time. As you start to write your program, start small and test frequently. When you test your sections, test them with both expected data and unexpected data. Where practical, functions should validate their arguments and perform reasonable actions (such as exiting with an error message or returning an error code) when presented with unreasonable data. A large number of security-related programs are simply bugs that have exploitable consequences. By writing code that is more reliable, you will also be writing code that is more secure.

4. Check all values provided by the user. An astonishing number of security-related bugs arise because an attacker sends an unexpected value or an unanticipated format to a program or a function within a program. A simple way to avoid these types of problems is by having your scripts always check and validate all of their arguments. Argument checking will not noticeably slow your scripts, but it will make them less susceptible to hostile users. As an added benefit, argument checking and error reporting will make the process of catching non security-relatedbugs easier.

Securely Using Fields, Hidden Fields, and Cookies: One of the reasons that it can be difficult to develop secure web applications has to do with the very architecture of web applications. When you develop an application, you generally write a body of code that runs locally on the web server and a much smaller body of code that is downloaded and run remotely on the user's web browser. You might spend a lot of time making sure that these two code bases work properly together.

For example, it's very important to make sure that the field names downloaded in web forms exactly match the field names that server-side scripts are expecting. And you will probably spend time making sure that the HTML forms, JavaScript, and other codes that are downloaded to the browser work properly on a wide range of different browser programs.

Using Fields Securely: When checking arguments in your program, pay special attention to thefollowing:

● Filter the contents of every field, selecting the characters that are appropriate for each response. For example, if a field is supposed to be a credit card number, select out the characters 0-9 and leaveall other characters behind. This will also allow people to enter their credit card numbers with spacesor dashes.
● After you filter, check the length of every argument. If the length is incorrect, do not proceed, butinstead generate an error.
● If you use a selection list, make certain that the value provided by the user was one of the legal values. Attackers can provide any value that they wish: they are not constrained by the allowablevalues in the selection list.
● Even if your forms use JavaScript to validate the contents of a form before it is submitted, be surethat you revalidate the contents on the server. An attacker can easily turn off JavaScript or bypass itentirely.

Hidden Fields and Compound URLs: A *hidden field* is a field that the web server sends to the web browser that is not displayed on the user's web page. Instead, the field merely sits in the browser's memory. When the form on the page is sent back to the server, the field and its contents are sent back. Some web developers use hidden fields to store information that is used for session tracking on e-commerce systems. For example, instead of using HTTP Basic Authentication, developers sometimes embed the username and password provided by the user as hidden fields in all future formentries:

```
<INPUT TYPE="hidden" NAME="username" VALUE="simsong">
<INPUT TYPE="hidden" NAME="password"
VALUE="myauth11">
```

Hidden fields can also be used to implement a shopping cart:

```
<INPUT TYPE="hidden" NAME="items" VALUE="3">
<INPUT TYPE="hidden" NAME="item1" VALUE="Book of Secrets:$4.99">
<INPUT TYPE="hidden" NAME="item2" VALUE="Nasty Software:$45.32">
<INPUT TYPE="hidden" NAME="item3" VALUE="Helping Hand:$32.23">
```

Instead of embedding this information in hidden fields, it can be placed directly in the URL. These URLs will then be interpreted as if they were forms that were posted using the HTTP GET protocol.For example, this URL embeds a username and password:

```
http://www.vineyard.net/cgi-
bin/password_tester?username=simsong&password=myauth11
```

Using Cookies: One attractive alternative to using hidden fields or URLs is to store information suchas usernames, passwords, shopping cart contents, and so on, in HTTP cookies.

Users can modify their cookies, so cookies used for user tracking, shopping carts, and other types ofe-commerce applications have all of the same problems described for hidden fields or compound URLs. But cookies also have problems all their own, including:

● Old cookies may continue to be used, even after they have "expired."
● Users may make long-term copies of cookies that are supposed to remain ephemeral and not everbe copied onto a hard drive.
● Some users are suspicious of cookies and simply turn off the feature.

Using Cryptography to Strengthen Hidden Fields, Compound URLs, and Cookies:

Many of the problems discussed in this section can be solved by using cryptography to protect the information in hidden fields, compound URLs, and cookies. Cryptography can:

- Prevent users from understanding the information stored on their computer.
- Allow web server applications to detect unauthorized or accidental changes to this information.

Here are the three examples from the previous sections, recoded to use cryptography.Username and password authentication:

```
<INPUT TYPE="hidden" NAME="auth"
VALUE="p6e6J6FwQOk0tqLFTFYq5EXR03GQ1wYWG0ZsVnk09yv7ItIHG17ymls4UM%2F1bw
HygRhp7ECawzUm
%0AKl3Q%2BKRYhlmGILFtbde8%0A:">
```

A secure shopping cart:

```
<INPUT TYPE="hidden" NAME="cart"
VALUE="fLkrNxpQ9GKv9%2FrAvnLhuLnNDAV50KhNPjPhqG6fMJoJ5kCQ5u1gh0ij8JBqphBx
dGVNOdja41XJ%0APLsT%2Bt1kydWN4Q%2BO9pW0yR9eIPLrzaDsZxauNPEe7cymPmXwd%2 B6c1L49uTwdNTKoS0XAThDzow%3D%3D%0A:">
```

A compound URL:

```
http://www.vineyard.net/cgi-bin/password_
tester?p6e6J6FwQOk0tqLFTFYq5EXR03GQ1wYWG0ZsVnk09yv7ItIHG17ymls4UM%2F1bwHyg Rhp7ECawzUm%0AKl3Q%2BKRYhlmGILFtbde8%0A:
```

Rules for Programming Languages: This section provides rules for making programs writtenin specific programming languages more secure.

Rules for Perl: Here are some rules to follow to make your Perl scripts more secure:

1. Use Perl's tainting features for all CGI programs. These features are invoked by placing the"-T" option at the beginning of your Perl script. Perl's tainting features make it more suited than C to CGI programming. When enabled, tainting marks all variables that are supplied by users as tainted." Variables whose values are dependent on tainted variables are themselves tainted as well. Tainted values cannot be used to open files or for system calls. Untainted information can only be extracted from a tainted variable by the use of Perl's string match operations.

2. The tainting feature also requires that you set the PATH environment variable to aknown "safe value" before allowing your program to invoke the *system( )* call.

3. Remember that Perl ignores tainting for filenames that are opened read-only. Nevertheless, be sure that you untaint all filenames, and not simply filenames that are used for writing.

Rules for C: It is substantially harder to write secure programs in C than it is in the Perl programming language; Perl has automatic memory management whereas C does not. Furthermore, because of the lack of facilities for dealing with large programs, Perl program sources tend to be smaller and more modular than their C counterparts.Nevertheless, one very important reason to write CGI programs in C remains: speed. Each time a CGI program is run, the program must be loaded into memory and executed. If your CGI program is writtenin Perl, the entire Perl interpreter must be loaded and the Perl program compiled before the CGI program can run. The overhead from these two operations dwarfs the time required bythe CGI program itself. The overhead of loading Perl can be eliminated by using the Apache Perl module or Microsoft's Internet Information Server *perl* script.

Rules for the Unix Shell: Don't write CGI scripts with the Unix shells (*sh*, *csh*, *ksh*, *bash*, or *tcsh*) for anything but the most trivial script. It's too easy to make a mistake, and there are many lurking security problems with these languages.

Using PHP Securely: PHP is a widely used and loved server-side scripting language for building web pages. Originally called Personal Home Page, and then PHP3, PHP now stands for PHP Hypertext Preprocessor. The web site for PHP development is http://www.php.org/. PHP is an official project of The Apache Foundation.

● It is easy to use and very fast. Even though PHP scripts are interpreted at runtime, the interpreter isbuilt into the web server. As a result, PHP pages can run significantly faster (more than 10 times faster is not uncommon) than the equivalent Perl/CGI web pages. Unlike CGI scripts, PHP pages do not need to be made "executable" or placed in special directoriesto run: if PHP is enabled in the web server, all you need to do is to give an HTML file a *.php* or
.php3 extension and the PHP system will automatically run.
● The PHP interpreter shows errors directly on the web page, not in a log file.
● PHP can cache connections to the MySQL database system. As a result, PHP pages that are fed from information in a database can display dramatically faster than database-driven pages using othersystems.
● PHP is extremely powerful. Scripts written in PHP can open files, open network connections, andexecute other programs.

Writing Scripts That Run with Additional Privileges: Many scripts need to run with user permissions different from those of the web server itself. On a Unix computer, the easiest way to do this is to make the script SUID or SGID. By doing this, the script runs with the permissions of the owner of the file, rather than those of the web server itself. On Macintosh, DOS, and Windows 95 systems, there is no such choice-scripts run with the same privileges and can access everything on the system.

Unfortunately, programs that run with additional privileges traditionally have been a source of security problems. This list of suggestions is based on those problems and specially tailored for the problems faced bythe web developer:
1. Avoid using the superuser (SUID root or SGID *wheel*) unless it is vital that your program perform actions that can only be performed by the superuser. For example, you will need to use SUID root if you want your program to modify system databases such as */etc/passwd*. But if you merely wish to have the program access a restricted database of your own creation, create a special Unix user for that application and have your scripts SUID to that user.

2. If your program needs to perform some functions as superuser but generally does not require SUID permissions, consider putting the SUID part in a different program and constructing a carefully controlled andmonitored interface between the two.

3. If you need SUID or SGID permissions, use them for their intended purpose as early in the program as possible and then revoke them by returning the effective and real UIDs and GIDs to those of the process thatinvoked the program.

4. Avoid writing SUID scripts in shell languages, especially in *csh* or its derivatives.

5. Consider creating a different username or group for each application to prevent unexpected interactions andamplification of abuse.

Connecting to Databases: It is common for a CGI program or script to connect to databases that are external to the web server. External databases can be used for many purposes, such as storing user preferences, implementing shopping carts, and even order processing. When the script runs, it opensa connection to the database, issues a query, gets the result, and then uses the result to formulate a response to the user. On some systems, a new database connection is created each time a new script is run. Other systems maintain a small number of persistent connections which are cached.

Database-backed web sites give a tremendous amount of power and flexibility to the web designer. Unfortunately, this approach can also reduce the overall security of the system: many of the security incidents mentioned earlier in this book happened because an attacker was able to execute arbitrarySQL commands on the database server and view the results. (For example, recall the story of theattacker who was able to obtain tens of thousands of credit card numbers from an e-commerce site.) If you decide to deploy a database server to supplement your web site, it is important to be sure thatthe server will be deployed and used securely.

Protect Account Information: Before the database server provides results to the script running on the web server, the server needs to authenticate the script to make sure it is authorized to access the information. Most databases use a simple username/password for account authentication, which means the script needs to have a valid username/password and present this information to the database server each time a request is issued.

Use Filtering and Quoting to Screen Out Raw SQL": As we mentioned earlier in this chapter, it is extremely important to filter all data from the user to make sure that it contains only allowable characters. When working with SQL servers, it is further important to properly quote data provided by the user before sending the data to the server. These procedures are used to prevent users from constructing their own SQL commands and sending that data to the SQL server.

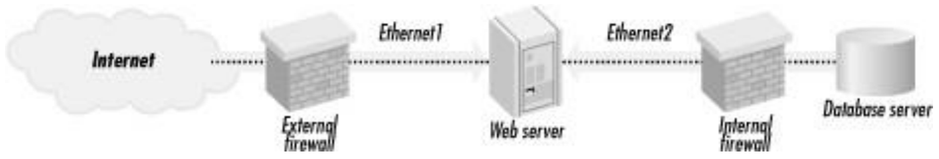
Protect the Database Itself: Finally, it is important that you protect the database serveritself:

- Configure your firewall or network topology so that it is impossible for people outside yourorganization to access your database server. For example, you may wish to set up your web server with two Ethernet adapters-one that connects to the Internet, and one that connects to asmall firewall appliance that, in turn, connects to your database server. The firewall should beset up so that only database queries can pass between the web server and the database server.

- Make sure logins on the database server are limited. The individuals with login capabilitiesshould be the system administrator and the database administrator.

- Make sure that the database server is backed up, physically protected, and maintained in thesame way as your other secure servers.

Connecting a database server and a web server to the Internet and your internal networkwith multiple firewalls:



UNIT – III
DATABASE SECURITY

Recent Advances in Access Control:

Information plays an important role in any organization and its protection against unauthorized disclosure (secrecy) and unauthorized or improper modifications (integrity), while ensuring its availability to legitimateusers (no denials-of-service) is becoming of paramount importance. An important service in guaranteeing information protection is the access control service.

Access control is the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied. An access control system can be considered at three different abstractions of control: access control policy, access control model, and access control mechanism. A policy defines the high level rules used to verify whether an access request is to be granted or denied. A policy is then formalized through a security model and is enforced by an access control mechanism.

An example of access matrix

	Document 1	Document 2	Program1	Program2
Ann	read, write	read	execute	
Bob	read	read	read, execute	
Carol		read, write		read, execute
David			read, write, execute	read, write, execute

The variety and complexity of the protection requirements that may need to be imposed in today‘s systems makes the definition of access control policies a far from trivial process. An access control system should besimple and expressive. It should be simple to make easy the management task of specifying and maintainingthe security specifications. It should be expressive to make it possible to specify in a flexible way different protection requirements that may need to be imposed on different resources and data. Moreover, an access control system should include support for the following features.

- Policy combination. Since information may not be under the control of a single authority, access control policies information may take into consideration the protection requirements of the owner, but also the requirements of the collector and of other parties. These multiple authorities' scenario should be supported from the administration point of view providing solutions for modular, large- scale, scalable policy composition and interaction.

- Anonymity. Many services do not need to know the real identity of a user. It is then necessary to make access control decisions dependent on the requester's attributes, which are usually proved by digital certificates.
- Data outsourcing. A recent trend in the information technology area is represented by data outsourcing, according to which companies shifted from fully local management to outsourcing the administration of their data by using externally service providers. Here, an interesting research challenge consists in developing an efficient mechanism for implementing selective access to the remote data.

Classical Access Control Models:

Classical access control models can be grouped into three main classes: discretionary access control (DAC), which bases access decisions on users' identity; mandatory access control (MAC), which bases access decisions on mandated regulations defined by a central authority; and role-based access control (RBAC), which bases access decisions on the roles played by users in the models. We now briefly present the main characteristics of these classical access control models.

Discretionary Access Control:

Discretionary access control is based on the identity of the user requesting access and on a set of rules, called authorizations, explicitly stating which user can perform which action on which resource. In the most basic form, an authorization is a triple (s, o, a), stating that user s can execute action a on object o. The first discretionary access control model proposed in the literature is the access matrix model [4, 5, 6]. Let S, O, and A be a set of subjects, objects, and actions, respectively.The access matrix model represents the set of authorizations through a $|S|\times|O|$ matrix A. Each entry A[s, o] contains the list of actions that subject s can execute over object o.

The access matrix model can be implemented through different mechanisms. The straightforward solution exploiting a two-dimensional array is not viable, since A is usually sparse. The mechanisms typically adopted are:

- Authorization table. The non empty entries of A are stored in a table with three attributes: user,action, and object.
- Access control list (ACL). The access matrix is stored by column, that is, each object is associated with a list of subjects together with a set of actions they can perform on the object.
- Capability. The access matrix is stored by row, that is, each

subject is associated with a listindicating, for each object, the set of actions the subject can perform on it.

User	Action	Object
Ann	read	Document1
Ann	write	Document1
Ann	read	Document2
Ann	Execute	Program1
Bob	read	Document1
Bob	read	Document2
Bob	read	Program1
Bob	execute	Program1
Carol	read	Document2
Carol	write	Document2
Carol	execute	Program2
David	read	Program1
David	write	Program1
David	execute	Program1
David	read	Program2
David	write	Program2
David	execute	Program2

(a)

From the access matrix model, discretionary access control systems have evolved and they includesupport for the following features.

- Conditions. To make authorization validity depend on the satisfaction of some specific constraints, today's access control systems typically support conditions associated with authorizations. For instance, conditions impose restrictions on the basis of: object content (content-dependent conditions), system predicates (system-dependent conditions), or accesses previously executed(history-dependent conditions)

- Abstractions. To simplify the authorization definition process, discretionary access control supportsalso user groups and classes of objects, which may also be hierarchically organized. Typically, authorizations specified on an abstraction propagate to all its members according to different propagation policies. Here, for example, an authorization specified for the Nurse group applies also to Bob and Carol.

- Exceptions. The definition of abstractions naturally leads to the need of supporting exceptions in authorization definition. Suppose, for example, that all users belonging to a group but u can access resource r. If exceptions were not supported, it would be necessary to associate an authorization witheach user in the group but u, therefore not exploiting the possibility of specifying the authorization of the group. This situation can be easily solved by supporting both positive and negative authorizations: the system would have a positive authorization for the group and a negative authorization for u.

The introduction of both positive and negative authorizations brings to two problems: inconsistency, when conflicting authorizations are associated with the same element in a hierarchy; and incompleteness, when some accesses are neither authorized nor denied.

Incompleteness is usually easily solved by assuming a default policy, open or closed (this latter being more common), where no authorization applies. In this case, an open policy approach allows the access, while the closed policy approach denies it.

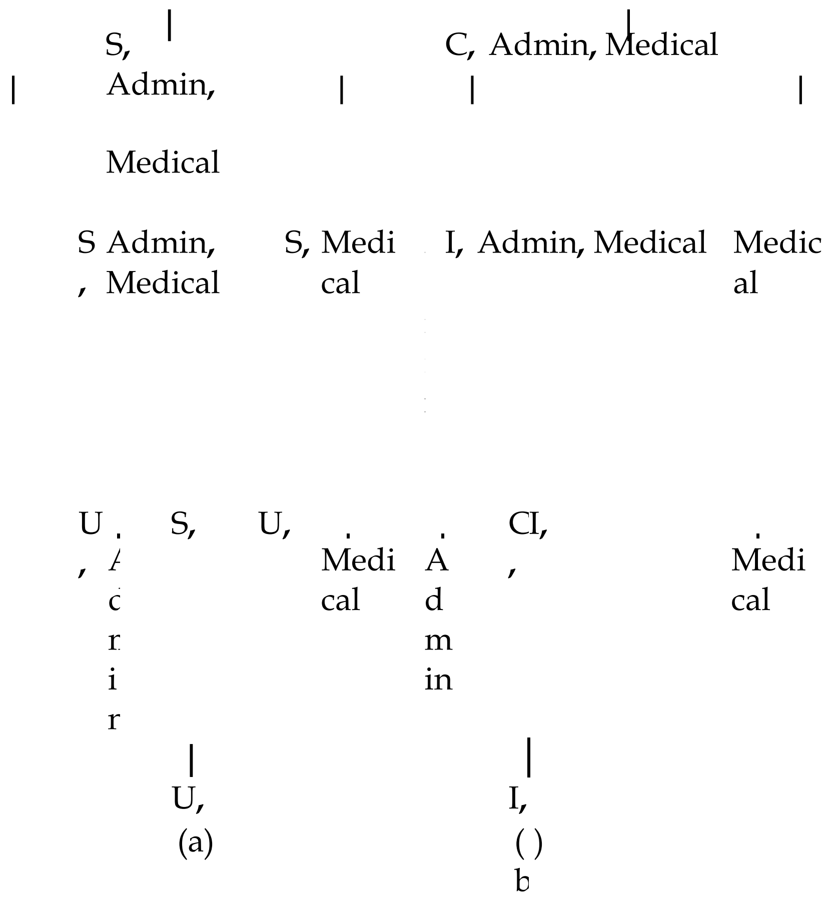
To solve the inconsistency problem, different conflict resolution policies have been proposed suchas:
– *No conflict*. The presence of a conflict is considered an error.
– *Denials take precedence*. Negative authorizations take precedence.
– *Permissions take precedence*. Positive authorizations take precedence.
– *Nothing takes precedence*. Conflicts remain unsolved.
– *Most specific takes precedence*. An authorization associated with an element n overrides a contradicting authorization (i.e., an authorization with the same subject, object, and action but with a different sign) associated with an ancestor of n for all the descendants of n. For instance, consider theuser-group hierarchy.

Mandatory Access Control:

Mandatory security policies enforce access control on the basis of regulations mandated by a central authority. The most common form of mandatory policy is the multilevel security policy, based on theclassifications of subjects and objects in the system. Each subject and object in the system is associated with an access class, usually composed of a security level and a set of categories. Security levels in the system are characterized by a total order relation, while categories form an unordered set. As a consequence, the set of access classes is characterized by a partial order relation, denoted ≥ and called dominance.

Given two access classes c1 and c2, c1 dominates c2, denoted c1 ≥ c2, iff the security level of c1 is greater than or equal to the security level of c2 and the set of categories of c1 includes the set of categories of c2. Access classes together with their partial order dominance relationship form a lattice. Mandatory policies can be classified as secrecy-based and integrity-based, operating in a dualmanner.

Mandatory policies can be classified as secrecy-based and integrity-based, operating in a dualmanner.



Secrecy-Based Mandatory Policy. The main goal of secrecy based mandatory policies is to protect data confidentiality. As a consequence, the security level of the access class associated with an objectreflects the sensitivity of its content, while the security level of the access class associated with a subject, called clearance, reflects the degree of trust placed in the subject not to reveal sensitive information. The set of categories associated with both subjects and objects defines the area of competence of users and data. A user can connect to the system using her clearance or any access class dominated by her clearance. A process generated by a user connected with a specific access class has the same access class as the user.

The access requests submitted by a subject are evaluated by applying the following two principles. No-Read-Up: A subject s can read an object o if and only if the access class of the subject dominatesthe access class of the object.

No-Write-Down: A subject s can write an object o if and only if the access class of the objectdominates the access class of the subject.

Role-Based Access Control:

A third approach for access control is represented by Role-Based Access Control (RBAC) models. A role is defined as a set of privileges that any user playing that role is associated with. When accessing the system, each user has to specify the role she wishes to play and, if she is granted to play that role, she can exploit the corresponding privileges. The access control policy is then defined through two different steps: first the administrator defines roles and the privileges related to each of them; second, each user is assigned with the set of roles she can play. Roles can be hierarchically organized to exploit the propagation of access control privileges along the hierarchy.

A user may be allowed to simultaneously play more than one role and more users may simultaneously play the same role, even if restrictions on their number may be imposed by the security administrator. It is important to note that roles and groups of users are two different concepts. A group is a named collection of users and possibly other groups, and a role is a named collection of privileges, and possibly other roles. Furthermore, while roles can be activated and deactivated directly by users at their discretion, the membership in a group cannot be deactivated.

The main advantage of RBAC, with respect to DAC and MAC, is that it better suits to commercial environments. In fact, in a company, it is not important the identity of a person for her access to the system, but her responsibilities. Also, the role-based policy tries to organize privileges mapping the organization's structure on the roles hierarchy used for access control.

Credential-Based Access Control:

In an open and dynamic scenario, parties may be unknown to each other and the traditional separation between authentication and access control cannot be applied anymore. Such parties can also play the role of both client, when requesting access to a resource, and server for the resources it makes available for other users in the system. Advanced access control solutions should then allow to decide, on one hand, which requester (client) is to be granted access to the resource, and, on the other hand, which server is qualified for providing the same resource. Trust management has been developed as a solution for supporting access control in open environments [19]. The first approaches proposing a trust management solution for access control are PolicyMaker and KeyNote.

The key idea of these proposals is to bind public keys to authorizations and to use credentials to describe specific delegations of trust among keys. The great disadvantage of these early solutions is that they assign authorizations directly to users' keys. The authorization specification is then difficult to manage and, moreover, the public key of a user may act as a pseudonym of her, thus reducing theadvantages of trust management, where the identity of the users should not be considered. The problem of assigning authorizations directly to keys has been solved by the introduction of digital certificates.A digital certificate is the on-line counterpart of paper credentials (e.g., a driver license).

A digital certificate is a statement, certified by a trusted entity (the certificate authority), declaring a set of properties of the certificate's holder (e.g., identity, accreditation, or authorizations). Access control models, by exploiting digital certificates for granting or denying access to resources, make access decisions on the basis of a set of properties that the requester should have. The final user can prove tohave such properties by providing one or more digital certificates**.**

The development and effective use of credential-based access control models require however tackling several problems related to credential management and disclosure strategies, delegation andrevocation of credentials, and establishment of credential chains. In particular, when developing an access control system based on credentials, the following issues need to be carefully considered.

- Ontologies. Since there is a variety of security attributes and requirements that may need tobe considered, it is important to guarantee that different parties will be able to understand each other, by defining a set of common languages, dictionaries, and Ontologies.
- Client-side and server-side restrictions. Since parties may act as either a client or a server,access control rules need to be defined both client-side and server-side.
- Credential-based access control rules. New access control languages supporting credentials need to be developed. These languages should be both expressive (to define different kinds ofpolicies) and simple (to facilitate policy definition).
- Access control evaluation outcome. The resource requester may not be aware of the attributesshe needs to gain access to the requested resource. As a consequence, access control mechanisms should not simply return a permit or deny answer, but should be able to ask the final user for the needed credentials to access the resource.
- Trust negotiation strategies. Due to the large number of possible alternative credentials that would enable an access request, a server cannot formulate a request for all these credentials, since the client may not be willing tore lease the whole set of her credentials.

On the other hand, the server should not disclose too much of the underlying security policy, since it may contain sensitive information. In the following, we briefly describe some proposals that havebeen developed for trust negotiation and for regulating service access in open environments.

Policy Composition:

In many real word scenarios, access control enforcement needs to take into consideration different policies independently stated by different administrative subjects, which must be enforced as if they were a single policy. As an example of policy composition, consider an hospital, where the global policy may be obtained by combining together the policies of its different wards and the externally imposed constraints (e.g., privacy regulations). Policy composition is becoming of paramount importance in all those contexts in which administrative tasks are managed by different, non collaborating, entities.

Policy composition is an orthogonal aspect with respect to policy models, mechanisms, and languages. As a matter of fact, the entities expressing the policies to be composed may even not be aware of the access control system adopted by the other entities specifying access control rules. The main desiderata for a policy composition framework can be summarized as follows.

Heterogeneous policy support: The framework should support policies expressed in differentlanguages and enforced by different mechanisms.

- Support of unknown policies. The framework should support policies that are not fully defined or are not fully known when the composition strategy is defined. Consequently, policies are to be treated as black-boxes and are supposed to return a correct and completeresponse when queried at access control time.
- Controlled interference. The framework cannot simply merge the sets of rules defined by thedifferent administrative entities, since this behavior may cause side effects. For instance, the accesses granted/denied might not correctly reflect the specifications anymore.
- Expressiveness. The framework should support a number of different ways for combining the input policies, without changing the input set of rules or introducing ad-hoc extensions to authorizations.
- Support of different abstraction levels. The composition should highlight the different components and their interplay at different levels of abstraction.
- Formal semantics. The language for policy composition adopted by the framework should be declarative, implementation independent, and based on a formal semantic to avoid ambiguity.

Access Control Models for XML:

Introduction:

The amount of information that is made available and exchanged on the Web sites is continuously increasing. A large portion of this information (e.g., data exchanged during EC transactions) is sensitive and needs to be protected. However, granting security requirements through HTML-based information processing turns out to be rather awkward, due to HTML's inherent limitations. HTML provides no clean separation between the structure and the layout of a document and some of its content is only used to specify the document layout. Moreover, site designers often prepare HTML pages according to the needs of a particular browser. Therefore, HTML markup has generally little todo with data semantics.

To the aim of separating data that need to be represented from how they are displayed, the World Wide Web Consortium (W3C) has standardized a new markup language: the eXtensible Markup Language (XML) [1]. XML is a markup meta-language providing semantics-aware markup without losing the formatting and rendering capabilities of HTML. XML's tags' capability of self-descriptionis shifting the focus of Web communication from conventional hypertext to data interchange. Although HTML was defined using only a small and basic part of SGML (Standard Generalized Markup Language: ISO 8879), XML is a sophisticated subset of SGML, designed to describe data using arbitrary tags. As its name implies, extensibility is a key feature of XML; users and applications are free to declare and use their own tags and attributes.

Therefore, XML ensures that both the logical structure and content of semantically rich informationis retained. XML focuses on the description of information structure and content as opposed to its presentation. Presentation issues are addressed by a separate language, XSL [2] (XML Style sheet Language), which is also a W3C standard for expressing how XML-based data should be rendered.

Since XML documents can be used instead of traditional relational databases for data storage and organization, it is necessary to think of a security system for XML documents protection. In this chapter, we will focus on access control enforcement. Specifically, in the literature, different access control models have been proposed for protecting data stored in XML documents, exploiting the flexibility offered by the markup language.

Preliminary Concepts:

XML (eXtensible Markup Language) is a markup language developed by the World Wide Web Consortium (W3C) and used for describing semi structured information. We introduce some of the most important concepts related to XML, which are useful to define an access control system for protecting XML documents.

Well-Formed and Valid XML Documents: XML document is composed of a sequence of (possibly nested) **elements** and **attributes** associated with them. Basically, elements are delimited by a pair of start and end tags (e.g., <request> and </request>) or, if they have no content, are composed of an empty tag (e.g., <request/>). Attributes represent properties of elements and are included in the start tag of the el-ement with which they are associated (e.g., <request number=―10‖>). An XML document is said to be **well-formed**

if its syntax complies with the rules defined by the W3C consortium [1], which can be summarized as follows:

• the document must start with the **prologue** <?xml version=–1.0‖?>;
• the document must have a **root** element, containing all other elements in the document;
• all open tags must have a corresponding closed tag, provided it is not an empty tag;
• elements must be properly nested;
• tags are case-sensitive;
• attribute values must be quoted.

An **XML language** is a set of XML documents that are characterized by a syntax, which describes the markup tags that the language uses and how they can be combined, together with its semantics. A **schema** is a formal definition of the syntax of an XML language, and is usually expressed through a schema language. The most common schema languages, and on which we focus our attention, are **DTD** and **XML Schema**, both originating from W3C.

Document Type Definition.

A DTD document may be either internal or external to an XML document and it is not itself written in the XML notation.

A DTD schema consists of definition of elements, attributes, and other constructs. An element declaration is of the form <!ELEMENT **element name content** >, where **element name** is an element name and **content** is the description of the content of an element and can assume one of the following alternatives:

o the element contains parsable character data #PCDATA); p the element has no content (Empty);
q the element may have any content (Any);
r the element contains a group of one or more subelements, which in turn may be composed of othersubelements;
s the element contains parsable character data, interleaved with subele-ments.

When an element contains other elements (i.e., subelements or mixed con-tent), it is necessary to declare the subelements composing it and their organi-zation. Specifically, sequences of elements are separated by a comma –,‖ and alternative elements are separated by a vertical bar – ‖. Declarations of se-quence and choices of subelements need to describe subelements' cardinality. With a notation inspired by extended BNF grammars, –*‖ indicates zero or more occurrences, –+‖ indicates one or more occurrences, –?‖ indicates zero or one occurrence, and no label indicates exactly one occurrence.

An attribute declaration is of the form <!ATTLIST **element name at-tribute def** >, where **element name** is the name of an element, and **attribute def** is a list of attribute definitions that, for each attribute, specify the at-tribute name, type, and possibly default value. Attributes can be marked as #REQUIRED, meaning that they must have an explicit value for each occur-rence of the elements with which they are associated; #IMPLIED, meaning that they are optional; #FIXED, meaning that they have a fixed value, indicated in the definition itself.

An XML document is said to be **valid** with respect to a DTD if it is syntactically correct according to the DTD. Note that, since elements and attributes defined in a DTD may appear in an XML document zero (optional elements), one, or multiple times, depending on their cardinality constraints, the structure specified by the DTD is not rigid; two distinct XML documents of the same schema may differ in the number and structure of elements.

XML Schema:

An XML Schema is an XML document that, with respect to DTD, has a number of advantages. First, an XML Schema is itself an XML document, consequently it can be easily extended for future needs. Furthermore, XML Schemas are richer and more powerful than DTDs, since they provide support for data types and namespaces, which are two of the most significant issues with DTD.

An element declaration specifies an element name together with a simple or complex type. A simple type isa set of Unicode strings (e.g., decimal, string, float, and so on) and a complex type is a collection of requirements for attributes, subelements, and character data that apply to the elements assigned to that type. Such requirements specify, for example, the order in which subelements must appear, and the cardinality of each subelement (in terms of maxOccurs and minOccurs, with 1 as default value). Attribute declarations specify the attributes associated with each element and indicate attribute name and type. Attribute declarations may also specify either a default value or a fixed value. Attributes can be marked as: required, meaning that they must have an explicit value for each occurrence of the elements with which they are associated; optional, meaning that they are not necessary.

Example 1. Suppose that we need to define an XML-based language for describing bank account operations. Figure 1(a) illustrates a DTD stating that each account operation contains a request element and one or more operation elements. Each account operation is also characterized by two mandatory attributes: bankAccN, indicating the number of the bank account of the requester; and id, identifying the single update. Each request element is composed of date, means, and notes elements, where only date is required. Element operation is instead composed of: type, amount, recipient, and possibly one between notes and value.

DTDs and XML documents can be graphically represented as trees. A DTD is represented as a labeled tree containing a node for each element, attribute, and value associated with fixed attributes. To distinguish elements and attributes in the graphical representation, elements are represented as ovals, while attributes as rectangles. There is an arc in the tree connecting each element with all the elements/attributes belonging to it, and between each #FIXED attribute and its value. Arcs connecting an element with its subelements are labeled with the cardinality of the relationship. Arcs labeled or and with multiple branching are used to represent a choice in an element declaration (|).

An arc with multiple branching is also used to represent a sequence with a cardinality constraint associated with the whole sequence (?, +, *). To preserve the order between elements in a sequence, for any two elements ei and ej, if ej follows ei in the element declaration, node ej appears below node ei in the tree. Each XML document is represented by a tree with a node for each element, attribute, and value in the document.

There is an arc between each element and each of its subelements/attributes/values and between each attribute and each of its value(s). Each arc in the DTD tree may correspond to zero, one, or multiple arcs in the XML document, depending on the cardinality of the corresponding containment relationship. Note that arcs in XML documents are not labeled, as there is no further information that needs representation.

Elements and Attributes Identification:

The majority of the access control models for XML documents identify the objects under protection (i.e., elements and attributes) through the XPath language [3]. XPath is an expression language, where the basicbuilding block is the path expression. A path expression on a document tree is a sequence of element names or predefined functions separated bycharacter / (slash): l1/l2/ . . . /ln. Path expressions may terminate with an attribute name as the last term of the sequence. Attribute names are syntactically distinguished by preceding them with special character @.

XPath allows the association of conditions with nodes in a path; in this case the path expression identifies the set of nodes that satisfy all the conditions. Conditional expressions in XPath may operate on the -text‖ of elements (i.e., character data in elements) or on names and values of attributes. A condition is represented by enclosing it within square brackets, following a label li in a path expression l1/l2/ . . . /ln.

The condition is composed of one or more predicates, which may be combined via and and or boolean operators. Each predicate compares the result of the evaluation of the relative path expression (evaluated at li) with a constant or with another expression. Multiple conditional expressions appearing in the same path expression are considered to be anded (i.e., all the conditions must be satisfied). In addition, conditional expressions may include functions last() and position() that permit the extraction of the children of a node that are in given positions. Function last() evaluates to true on the last child of the current node. Function position () evaluates to true on the node in the evaluation context whose position is equal to the context position.

Path expressions are also the building blocks of other languages, such as XQuery [4] that allows to make queries on XML documents through FLWOR expressions. A FLOWR expression is composed of the following clauses:

• FOR declares variables that are iteratively associated with elements in the XML documents, which areidentified via path expressions;
• LET declares variables associated with the result of a path expression; • WHERE imposes conditions ontuples;
• ORDER BY orders the result obtained by FOR and LET clauses;
• RETURN generates the final result returned to the requester.
Example 2. Consider the DTD and the XML document in Example 1. Some examples of path expressionsare the following.
• /account operation/operation: returns the content of the operation element, child of account operation;
• /account operation/@bankAccN: returns attribute bankAccN of element account operation;
• /account operation//notes: returns the content of the notes elements, anywhere in the sub tree rooted at account operation; in this case, it returns both /account operation/request/notes and /account operation/operation/notes;
• /account operation/operation[./type=-bank transfer‖]: returns the content of the operation element, child of account operation, only if the type element, child of operation, has value equal to -bank transfer‖.

The following XQuery extracts from the XML document in Fig. 1(b) all the account operation elements with operation type equal to -bank transfer‖. For the selected elements, the amount and the recipient of the operation are returned, along with all notes appearing in the selected account operation element.

```
<BankTransf>
{ FOR $r in document(-update
account‖)/account operation WHERE
$r/operation/type=-bank transfer‖
RETURN $r/operation/amount, $r/operation/recipient, $r//notes
}
</BankTransf>
```

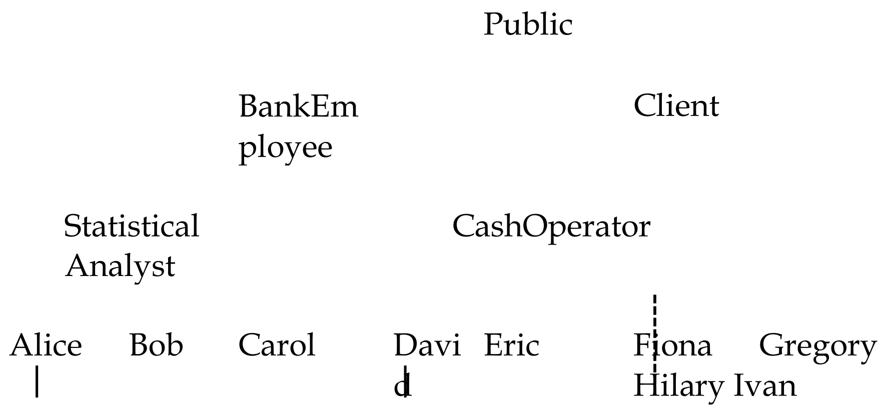
XML Access Control Requirements:

Due to the peculiar characteristics of the XML documents, they cannot be protected by simply adopting traditional access control models, and specific models need to be defined. By analyzing the existing proposals, it is easy to see that they are all based on the definition of a set of authorizations that at least specify the subjects on which they apply the objects to be protected, and the action to be executed. The existing XML-based access control models differentiate on the basis of the subjects, objects, and actions they can support for access control specification and enforcement.

Subject: Subjects are usually referred to on the basis of their identities or of the network location from which requests originate. Locations can be expressed with reference to either the numeric IP address (e.g., 150.100.30.8) or the symbolic name (e.g., bank.com) from which the request comes. It often happens that the same privilege should be granted to sets of subjects, which share common characteristics, such as the department where they work, or the role played in the company where they work. To the aim of simplifying the authorizations definition, some access control models allow the specification of authorizations having as subject:

• a group of users, which is a statically defined set of users; groups can be nested and overlapping;
• a location pattern, which is an expression identifying a set of physical locations, obtained by using the wild character * in physical or symbolic addresses;
• a role, which is a set of privileges that can be exploited by any user while playing the specific role; users can dynamically decide which role to play, among the ones they are authorized to play. Also, subjects are often organized in hierarchies, where an authorization defined for a general subject propagates to its descendants.

An example of user-group hierarchy:



A hierarchy can be pictured as a directed acyclic graph containing a node for each element in the hierarchy and an arc from element x to element y, if x directly dominates y. Dominance relationships holding in the hierarchy correspond to paths in the graph. Figure 3 shows an example of user-group hierarchy.

Object Granularity: The identification of the object involved in a specific authorization can exploit the possibility given by XML of identifying elements and attributes within a document through path expressions as defined by the XPath language.

Action: Most of the proposed XML access control models support only read operations, since there is not a standard language for XML update. Furthermore, the management of write privileges is a difficult task, which needs to take into account both the access control policy and the DTD (or XML Schema) defined for the document. In fact, the DTD may be partially hidden to the user accessing the document, as some elements/ attributes may be denied by the access control policy. For instance, when adding an element to the document, the user may even not be aware of the existence of a required attribute associated with it, as she is not entitled to access the attribute itself.

Support for Fine and Coarse Authorizations. The different protection re-quirements that different documents may have call for the support of access restrictions at the level of each specific document. However, re-quiring the specification of authorizations for each single document would make the authorization specification task too heavy. The system may then support, beside authorizations on single documents (or portions of doc-uments), authorizations on collections of documents [9]. The concept of DTD can be naturally exploited to this end, by allowing protection re-quirements to refer to DTDs or XML documents, where requirements specified at the level of DTD apply to all those documents instance of the considered DTD. Authorizations specified at DTD level are called **schema level** authorizations, while those specified at XML document level are called **instance level** authorizations.