The University of Sydney
School of Computer Science
Dr Suranga Seneviratne
Senior Lecturer - Security

THE UNIVERSITY OF SYDNEY

Introduction to Security Engineering

Recommended Reading

Security Engineering - 3^{rd} Edition by Ross Anderson

- **Chapter 1:** What is Security Engineering?
- **Chapter 2:** Who is the Opponent?

Cryptography and Network Security - 7^{th} Edition by William Stallings

- **Chapter 1:** Computer Network and Security Concepts

1 What is Engineering?

Engineering is the application of science and maths to solve problems. While scientists and inventors develop theories and innovations, engineers apply these discoveries to the real world. For example, the principles of thermodynamics, which are studied in physics, are applied by mechanical engineers to design efficient heating, ventilation, and air conditioning (HVAC) systems. These systems control the climate in buildings, ensuring comfort and air quality. For example, the principles of aerodynamics, which are studied in physics, are applied by aerospace engineers in the design and development of aircraft. By understanding how air flows over wings and other surfaces, engineers can optimize the shape and structure of aeroplanes to reduce drag, increase lift, and improve fuel efficiency.

Similar to the principles of thermodynamics and aerodynamics are applied to create practical solutions in HVAC systems and aircraft design, the principles of security engineering are employed to safeguard information and systems. In a similar fashion, security engineering uses scientific and mathematical principles to develop methods for protecting data, ensuring secure communication, and mitigating risks in both digital and physical environments.

2 What is Security Engineering?

The early definition of security engineering comes from an essay from Anderson and Needham titled “Programming Satan’s Computer” [1]. The key excerpt from the essay is:

Programming Satan's Computer

Cryptographic protocols are used in distributed systems to identify users and authenticate transactions. They may involve the exchange of about 2–5 messages, and one might think that a program of this size would be fairly easy to get right. However, this is absolutely not the case: bugs are routinely found in well-known protocols and years after they were first published. **The problem is the presence of a hostile opponent, who can alter messages at will. In effect, our task is to program a computer which gives answers which are subtly and maliciously wrong at the most inconvenient possible moment.** This is a fascinating problem; and we hope that the lessons learned from programming Satan's computer may be helpful in tackling the more common problem of programming Murphy's.

In simpler words, the excerpt says it is extremely difficult to build secure cryptographic protocols in the presence of an adversary who can tamper with messages. This hostile opponent can manipulate the communication in ways that are difficult to anticipate and defend against. The security engineer's job is to ensure that the protocol works even under attack.

A historical example that illustrates this challenge is the development of the Enigma machine during World War II. The Germans believed their cryptographic device was unbreakable, yet Allied cryptanalysts, including Alan Turing, managed to crack its codes. This breakthrough highlighted the importance of not only creating robust encryption but also anticipating and defending against the innovative strategies of adversaries. The lessons learned from securing communications during wartime continue to influence modern security engineering practices, emphasizing the need for vigilance and adaptability in protecting our digital and physical assets.

This concept can be extended beyond cryptographic protocols to encompass anything that requires security, such as hardware, software, or networks. In general, we will use the term **"distributed systems"** throughout this course to refer to any system where components located on different networked computers communicate and coordinate their actions by passing messages. Distributed systems often involve a client-server architecture but can also include peer-to-peer networks, cloud computing, and other decentralized models.

Security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, as well as to adapt existing systems as their environment evolves. The concept extends beyond mere fault tolerance, as in safety and reliability engineering.

Security Engineering

Security engineering means conceiving, designing, and implementing hardware and software that produces only the expected answers, even when confronted with malice, error, or mischance.

A secure system tolerates the activities of the opponent. However, there is no such thing as a perfectly secure system. Security is multi-faceted, and security engineering requires cross-disciplinary expertise, ranging from cryptography and computer security through hardware tamper-resistance to a knowledge of economics, applied psychology, organisations and the

law. System engineering skills, from business process analysis through software engineering to evaluation and testing, are also important, but they are not sufficient, as they deal only with error and mischance rather than malice. The security engineer also needs some skill at adversarial thinking, just like a chess player; you need to have studied lots of attacks that worked in the past, from their openings through their development to the outcomes.

3 Security Engineering vs. Safety Engineering

It is important to differentiate Security Engineering from **Safety Engineering** and **Reliability Engineering**.

- **Safety Engineering** - Safety engineering is primarily concerned with preventing accidents and ensuring that systems operate without causing harm to people, property, or the environment. Safety engineers identify potential hazards, assess risks, design safety features, and develop risk management procedures. They also investigate accidents to prevent recurrence.
- **Reliability Engineering** - Reliability engineering ensures that systems and components perform their intended functions without failure over a specified period. The primary goal is to enhance the dependability and availability of systems, minimizing downtime and maintenance costs.

Neither of these considers a hostile opponent, which differentiates them from security engineering. Follow the below example to get a better idea of the three disciplines.

Exercise

Consider the following scenarios and identify which type of engineering each one pertains to: Safety Engineering, Reliability Engineering, or Security Engineering.

1. A chemical plant is implementing measures to prevent the release of toxic gases that could harm workers and nearby residents. **Answer: Safety Engineering**
2. An organization is setting up a network of sensors to continuously monitor the operational status of its manufacturing equipment to predict and prevent unexpected breakdowns. **Answer: Reliability Engineering**
3. A bank is enhancing its IT infrastructure to protect customer data from cyberattacks, including the use of firewalls, encryption, and intrusion detection systems. **Answer: Security Engineering**
4. A commercial airline is conducting a thorough analysis of its aircraft components to identify and rectify potential failure modes before they can lead to in-flight malfunctions. **Answer: Reliability Engineering**
5. A hospital is installing backup power systems and fail-safe mechanisms to ensure that critical medical equipment remains operational during a power outage. **Answer: Reliability Engineering**
6. A factory is implementing emergency shutdown procedures and installing safety barriers around hazardous machinery to prevent worker injuries in case of an accident. **Answer: Safety Engineering**
7. A government agency is deploying a multi-factor authentication system to enhance the security of its sensitive databases and prevent unauthorized access. **Answer: Security Engineering**
8. A tech company is implementing secure coding practices and conducting regular penetration testing to identify and fix vulnerabilities in its software applications. **Answer: Security Engineering**

4 A Framework for Security Engineering

The framework we use in this class comes from the textbook we use by Ross Anderson [2]. Before going into the framework, let's try to understand why we need a framework for security engineering. We need some reference to design and assess the security of a distributed system. Also, a framework will help assist investigations after a security incident. Later, we will realise that any cybersecurity incident can be traced back to a failure in one of the components of this security framework.

To build really dependable distributed systems, we need four things to come together: policy, mechanisms, assurance, and incentives. All of these are interconnected, as shown in Figure 1.

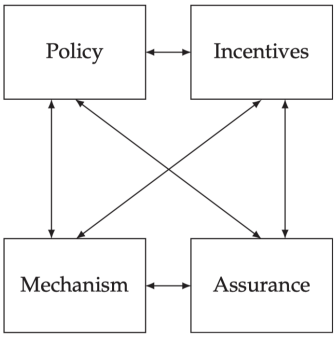

Figure 1: Security Engineering Analysis Framework (**Figure source** [2])

Policy defines what we mean to achieve. In other words, it defines what it means to keep the system secure. Often, higher management sets security policies that ensure business continuity and comply with various regulations and standards. Some example policies are *"Customer data must be stored only within our data centres."* and *"Only authorized employees must be able to access the personal information of clients."* Note that the policies do not define how to implement policies. Rather, they define what we want to achieve.

Mechanisms are the machinery we have to implement policies. E.g., encryption for data confidentiality, tamper-resistant hardware for physical security, and cryptographic hashes for data integrity. For most beginners in security engineering, it is all about machinery. However, security is a broader concept than only mechanisms. It is important to understand that it is only one aspect of our framework. Security engineers must be aware and familiar with various mechanisms that are required to implement policies.

Assurance refers to how much reliance you can place on each particular mechanism and how well they work together. For example, how much time is required to break RSA encryption based on current computing capacities? And how much time is required to brute-force a 10-character password? Security engineers must be able to conduct professional estimates of assurances given by commonly used security mechanisms. Note that some of the security mechanisms we consider secure under current hardware may not be secure in future. For instance, it would take trillions of years for a classical computer to break RSA-2048. However, breaking RSA-2048 will take around 10 seconds in a quantum computer.

Incentives refer to the motive that the people guarding and maintaining the system have to do their job and also the motive that the attackers have to try to defeat your policy. Depending on the business, we may get attacked by different attackers - State actors, Cybercriminals/Ransomware groups, Script kiddies, and Hacktivists. The methodical process of understanding the possible threats and prioritising defence mechanisms is called "Threat Modelling". Security costs money – Understanding incentives allows us to understand the degree of security we need and to decide against which attacks we need to prioritise our defences.

Security engineers need to understand all this; we need to be able to put risks and threats in context, make realistic assessments of what might go wrong, and give our clients good advice. That depends on a wide understanding of what has gone wrong over time with various systems; what sort of attacks have worked, what their consequences were, and how they were stopped.

4.1 Example: Bank Security

Banks operate numerous security-critical computer systems. A wide range of different applications run together and many people work together, presenting a huge attack surface and the potential for insider attacks.

Policy

- The overall goal could be: Maintaining correct account balances.
- Sub-goals could be: Safeguard the integrity, confidentiality and availability of ATMs or online banking systems; Ensure the accountability of bank-to-bank transfers etc.

Mechanisms

- Strong bookkeeping procedures and access control, especially in the backend.
- Anomaly detection systems (e.g., detecting unusual transactions).
- Four-eyes principle: large transfers typically need at least two people to authorise them.
- ATMs use strong cryptographic protocols for authentication.

Assurance

- Assurance refers to the confidence in the security measures protecting the bank's operations, heavily dependent on the mechanisms' effectiveness.
- The most significant perceived threat is internal, stemming from employees who might commit petty theft. This highlights that even the most robust bookkeeping or cryptographic defenses have vulnerabilities, especially against insiders familiar with the bank's systems.
- Anomaly detection plays a key role in identifying potential security breaches, yet it's challenging because not all unusual transactions are illegitimate; some may be exceptional but authentic customer activities. The bank must balance the sensitivity of its anomaly detection systems to flag suspicious activities without causing unnecessary disruption to legitimate banking operations.

Incentives

- For attackers: They aim to obtain customer's money without the risk of detection or conduct a series of transactions that obscure the money's trail, ultimately making the funds unrecoverable or dispersed beyond retrieval.
- For guards and admins: They may want to leverage their positions for personal advantage, a scenario that is increasingly recognized as a common vector for attacks. Therefore, they need to be carefully screened and vetted.

Case Study 2013 Target (US) Data Breach

In late 2013, Target, one of the largest retail chains in the United States, experienced a massive data breach. Attackers gained access to the credit and debit card information of approximately 40 million customers, as well as the personal information of up to 70 million individuals, by installing malware in Target POS (Point of Sales) systems.

You can read the details of what happened in the two links below.

- https://coverlink.com/cyber-liability-insurance/target-data-breach/
- https://www.sipa.columbia.edu/sites/default/files/2022-11/Target%20Final.pdf

Let's investigate this incident using the security engineering framework.

Policy - Target didn't have policies in place to ensure that their third-party suppliers had a proper security posture. Specifically, Target granted network access to a third-party HVAC contractor, Fazio Mechanical Services, without verifying their security practices. This lapse allowed attackers to exploit the contractor's credentials to gain entry into Target's network.

Since then, many organizations now have strengthened their third-party risk management policies. They now require vendors to comply with stringent security standards, conduct regular security audits, and implement continuous monitoring of third-party access. Organizations often use third-party risk management platforms to assess, monitor, and manage vendor risks effectively.

Mechanisms - While the technical mechanism worked, i.e., the FireEye malware detection tool detected the threats, Target didn't have proper incident response mechanisms to react, allowing the attackers to continue for weeks. Equally worked are the credit card fraud detection systems that start flagging fraudulent transactions.

Assurance - We have information that the FireEye tool was reliable in detecting the malware. Whether the other security mechanisms, such as Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), worked as expected, we don't know. Mainly because of the in-action regarding the incident from Target's side. The "Pass-the-hash" method the attackers used highlighted the limitations of the Single Sign On (SSO) mechanism without other supplementary mechanisms such as the principle of least privilege.

Incentives - Target, a larger company with a large customer base, was a lucrative target to the attackers. Not to mention, it was also the high-volume holiday shopping season. It is unclear whether Target's security team had the incentive to do their job properly as they failed to take timely action to the FireEye alarms. It also appears that Target's management had prioritised holiday season operations rather than timely notifying the impacted customers.

5 Security Goals

Security goals are used to define security policies. Note that sometimes there are overlaps between some of the goals.

Confidentiality and Secrecy

Confidentiality preserves authorised restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.

The most general understanding of confidentiality is the effect of mechanisms that limit who can access data in plaintext form. This implies that access control is a form of confidentiality or secrecy. Secrecy is an engineering term that refers to the effect of the mechanisms used to limit the number of principals who can access information, such as cryptography or computer access controls. Confidentiality involves an obligation to protect some other person's or organisation's secrets if you know them.

The other use of confidentiality is the goal of protecting the secret content of a message or data, commonly by encryption. We do not distinguish between secrecy and confidentiality and we rarely need to refer to the obligation aspect of confidentiality; hence we simplify our terminology.

Data/Message Integrity

Integrity refers to the effect of a mechanism to allow a verifier to check whether a message has been modified in transit or data has been altered since the last legitimate update. A loss of integrity is the unauthorized modification or destruction of information.

Generally, defending against data modification in transit or message alterations is challenging. The attacker may have a useful position in the network and can flip bits. The important effect of integrity is that we can detect such changes. Integrity is very close in meaning to tampering-evident. However, it's not the same as tampering resistance. Tampering resistance refers to the capability to withstand attempts at modification, which is more commonly used for hardware.

Authenticity

Authenticity refers to the effect of allowing a verifier to determine the originator of a message or data item. This involves verifying that users are who they claim to be and that each input arriving at the system came from a trusted source. Effectively implementing authenticity as a mechanism typically implies integrity and freshness as well. Note that authenticity is not the same as authentication (the process of identifying a party we are communicating with).

Authenticity

Authenticity: Authenticity ensures that the origin of the message or data item is genuine, and it often encompasses both integrity and freshness.

Integrity: Integrity is a crucial part of authenticity, as it ensures the message has not been altered in transit. Without integrity, we cannot trust the content of the message.

Freshness: Freshness ensures that the message is new and not a replay of an old message. Without freshness, even an authentic and unaltered message could be reused by an attacker in a harmful way.

Authorisation

Authorisation refers to the effect that a verifier can determine whether an entity is allowed to execute some action or access some data. Authorisation often utilises authentication but does not imply it. For example, when entering a cinema, you are required to show authorization (such as a ticket) rather than prove your identity. Authorisation is closely linked to access control and accountability.

Authorisation

Alice, as a Finance Manager, tries to access the financial reporting system:

Authorisation: The system checks her role and determines she is authorized to access financial reports.

Access Control: Based on this authorization, the access control system allows Alice to access the financial reporting application and view/edit financial data. Note that we don't consider access control to be a security goal, though some materials do. In this class we will consider it as a mechanism.

Accountability: Every action Alice performs within the financial reporting system, such as viewing, editing, or exporting reports, is logged with her user ID and timestamp. If there is an audit or an investigation, the logs provide a clear record of Alice's activities, ensuring she is held accountable for her actions.

Accountability

Accountability refers to the effect that a verifier can determine which entity (or which category of authorised entities) is responsible for a certain action. It allows for the tracing of an entity's actions within the system. It also implies authorisation and access control and may imply authentication. It is closely related to non-repudiation. Since truly secure systems are not yet an achievable goal, we must be able to trace a security breach to a responsible party. Systems must keep records of their activities to permit later forensic analysis to trace security breaches or to aid in transaction disputes. Thus, we need some form of logging, which needs to be auditable — meaning the verifier can access and trust the logs. The protection of logs is also crucial.

Non-repudiation

Non-repudiation means that the entity that has caused an action cannot successfully deny responsibility. At its core, this is a legal concept. It guarantees that one cannot deny sending or receiving a transaction. Note that in law, we do not need cryptographic proof; we only need plausible evidence to make a case. Non-repudiation is supported by accountability and, in cryptographic terms, requires authenticity and integrity, as well as secure timestamping of the action, which is challenging to achieve.

Deniability

Deniability is the capability to successfully reject the notion of being responsible for some action. In many ways, it's the opposite of non-repudiation and can be more useful, especially in scenarios involving highly confidential communications. Deniability can be partially achieved through cryptographic methods. It's important to note that the understanding of deniability in security contexts may differ from its interpretation by a court.

Availability

Availability refers to a system's capability of providing its services when needed, even during an attack, error, or mishap. It assures that systems work promptly and service is not denied to authorized users. A loss of availability is the disruption of access to or use of information or an information system. Well-known DDoS (Distributed Denial of Service) attacks are examples of attacks against availability.

Since availability is important in many applications, sometimes, a lower level of security may be exchanged for higher availability. For example, in warfare, the availability of communication channels might be prioritized over confidentiality, as the ability of commanders to communicate is often more crucial than safeguarding communications from eavesdroppers. In such cases, encryption might be disabled if necessary. Availability is hard to achieve, given that many systems are asymmetric by design (offering exploitable vulnerabilities), and attackers often have a huge attack surface to choose from.

Privacy

There is no straightforward definition due to its various aspects related to society, law, and technology. Privacy implies an entity's ability to control what information relates to themselves to release or hide. Technology can support the goal of privacy to a degree. Different aspects of privacy include: privacy of static data (released data sets); privacy of dynamic data (DB queries); privacy when moving through the network.

Note that anonymity is different from privacy. Anonymity refers to the state of being unidentified or unidentifiable within a particular context. It means an individual's identity is unknown and cannot be traced back to them.

Privacy vs. Anonymity

Privacy: Alice joins a health forum where her real name and home address are required for registration, but she can control what information is visible to others. The forum ensures her data is protected and only accessible by authorized personnel, demonstrating privacy by protecting her identifiable information while allowing her to participate.

Anonymity: Bob joins the same forum but uses a pseudonym like "HealthSeeker123" and does not provide any real personal information. The forum does not log his IP address or any identifiable data, ensuring that his identity remains completely unknown, exemplifying anonymity.

6 Fundamental Security Design Principles

Despite years of research and development, it has not been possible to develop security design and implementation techniques that systematically exclude security flaws and prevent all unauthorized actions. However, we have a set of widely agreed design principles that can guide the development of protection mechanisms.

Economy of mechanism

The design of security measures embodied in both hardware and software should be as simple and small as possible. Relatively simple, small design is easier to test and verify thoroughly. With a complex design, there are many more opportunities for an adversary to discover subtle weaknesses to exploit that may be difficult to spot ahead of time.

Example: UNIX is modularly designed with a small codebase. The UNIX operating system's design principles have had a lasting impact on the development of secure and efficient software systems. Many modern operating systems, including Linux and macOS, trace their heritage back to UNIX and its design philosophy. Windows Vista, on the other hand, had many complex features and a significantly larger codebase compared to its predecessors. The large number of features and the extensive codebase increased the attack surface, providing more opportunities for adversaries to find and exploit vulnerabilities.

Fail-safe defaults

Access decisions should be based on permission rather than exclusion. That is, the default situation is lack of access, and the protection scheme identifies conditions under which access is permitted. This approach exhibits a better failure mode than the alternative approach, where the default is to permit access. An example could be file permissions in operating systems: Systems default to denying file access, requiring explicit user permissions to grant access. This ensures that errors default to safe denials rather than unintended access. However, the idea os also applicable beyond access control as well.

Example: The POODLE attack exploits the fallback mechanism in SSL/TLS protocols, where connections downgrade to the older, less secure SSL 3.0 if negotiation for a more secure protocol fails.

Complete mediation

Every access must be checked against the access control mechanism. Systems should not rely on access decisions retrieved from a cache. In a system designed to operate continuously, this principle requires that, if access decisions are remembered for future use, careful consideration be given to how changes in authority are propagated into such local memories. File access systems appear to provide an example of a system that complies with this principle. However, typically, once a user has opened a file, no check is made to see if permissions change. To fully implement complete mediation, every time a user reads a field or record in a file, or a data item in a database, the system must exercise access control. This resource-intensive approach is rarely used.

Example: Zero Trust Networks (ZTN) serve as a contemporary example of the principle of complete mediation in access control mechanisms. In a Zero Trust model, every access request is thoroughly validated against security policies and access controls, irrespective of the user's previous authorization status. This approach ensures that no trust is granted based on network location or previous authentications.

Open design

The design of a security mechanism should be open rather than secret. For example, although encryption keys must be secret, encryption algorithms should be open to public scrutiny. The algorithms can then be reviewed by many experts, and users can, therefore, have high confidence in them. This is the philosophy behind the National Institute of Standards and Technology (NIST) program of standardizing encryption and hash algorithms and has led to the widespread adoption of NIST-approved algorithms.

Example: The Content Scramble System (CSS) used in DVDs violated the principle of open design by keeping its encryption algorithm secret. This lack of transparency prevented public scrutiny and expert analysis, which eventually led to significant security flaws. In November 1999, Frank A. Stevenson published three exploits that rendered the CSS cipher practically ineffective. Subsequently, a group of hackers reverse-engineered CSS, creating the DeCSS program to bypass the encryption. The secrecy of the design ultimately resulted in a compromised security mechanism, illustrating that security through obscurity is not a reliable strategy.

Kerkhoff's principle: A related concept from cryptography, Kerkhoff's principle, states that a cryptographic system should be secure even if everything about the system except the key is public knowledge, aligns closely with the principle of open design. Both principles emphasize that security should rely on the secrecy of the keys, not the obscurity of the algorithms, allowing for public scrutiny and expert analysis to ensure robust and reliable security mechanisms.

Separation of privilege

A practice in which multiple privilege attributes are required to achieve access to a restricted resource. A good example of this is multifactor user authentication, which requires the use of multiple techniques, such as a password and a smart card, to authorize a user. The term is also now applied to any technique in which a program is divided into parts limited to the specific privileges it requires to perform a specific task. This is used to mitigate the potential damage

of a computer security attack. One example of this latter interpretation of the principle is removing high privilege operations to another process and running that process with the higher privileges required to perform its tasks. Day-to-day interfaces are executed in a lower-privileged process.

Example: On Berkeley-based versions of the UNIX operating system, users are not allowed to change from their accounts to the root account unless two conditions are met. The first condition is that the user knows the root password. The second condition is that the user is in the wheel group (the group with GID 0). Meeting either condition is not sufficient to acquire root access; meeting both conditions is required.

Least privilege

Every process and every user of the system should operate using the least set of privileges necessary to perform the task. A good example of the use of this principle is role-based access control. The system security policy can identify and define the various roles of users or processes. Each role is assigned only those permissions needed to perform its functions. Each permission specifies a permitted access to a particular resource (such as read and write access to a specified file or directory, connect access to a given host and port). Unless a permission is granted explicitly, the user or process should not be able to access the protected resource. More generally, any access control system should allow each user only the privileges that are authorised for that user. There is also a temporal aspect to the least privilege principle. For example, system programs or administrators who have special privileges should have those privileges only when necessary; when they are doing ordinary activities, the privileges should be withdrawn. Leaving them in place just opens the door to accidents.

Example: In the Apache web server environment, the "www-data" user account is designated specifically for running the Apache web server and associated processes. The "www-data" user is granted only the minimal permissions necessary to serve web content, such as reading files from the web directory and writing to specific log files. It does not have administrative privileges or access to other parts of the operating system. This restricted access ensures that even if the web server is compromised by an attacker, the potential damage is limited to the web server context, preventing the attacker from modifying system files, accessing sensitive data elsewhere on the server, or escalating privileges to perform broader system attacks.

Least common mechanism

The design should minimize the functions shared by different users, providing mutual security. This principle helps reduce the number of unintended communication paths and reduces the amount of hardware and software on which all users depend, thus making it easier to verify if there are any undesirable security implications. For example, operating systems provide distinct user environments or workspaces to minimize shared functions and resources, enhancing security by isolating user activities and reducing unintended interactions.

Example: Passwords must not be re-used between different systems/services.

Psychological acceptability

Security mechanisms should not interfere unduly with the work of users, while at the same time meeting the needs of those who authorize access.

If security mechanisms hinder the usability or accessibility of resources, then users may opt to turn off those mechanisms. Where possible, security mechanisms should be transparent to the users of the system or at most introduce minimal obstruction. In addition to not being intrusive or burdensome, security procedures must reflect the user's mental model of protection. If the protection procedures do not make sense to the user or if the user must translate his image of protection into a substantially different protocol, the user is likely to make errors.

Example: A good example is incorporating fingerprint or facial recognition for device access combines high security with ease of use, making security seamless and aligned with user expectations, minimizing disruptions to their workflow.

Isolation

Public access systems should be isolated from critical resources (data, processes, etc.) to prevent disclosure or tampering. In cases where the sensitivity or criticality of the information is high, organizations may want to limit the number of systems on which that data is stored and isolate them, either physically or logically. Physical isolation may include ensuring that no physical connection exists between an organization's public access information resources and an organization's critical information. When implementing logical isolation solutions, layers of security services and mechanisms should be established between public systems and secure systems responsible for protecting critical resources.

The processes and files of individual users should be isolated from one another except where it is explicitly desired. All modern operating systems provide facilities for such isolation so that individual users have separate, isolated process space, memory space, and file space, with protections for preventing unauthorized access.

Example: Apple devices utilize the Secure Enclave, a hardware-based security mechanism, to isolate sensitive cryptographic operations and data. The Secure Enclave is a coprocessor integrated into Apple's system on a chip (SoC) that handles encryption, decryption, and key management independently from the main processor. This isolation ensures that even if the main operating system is compromised, the cryptographic keys and operations remain secure within the Secure Enclave. This physical and logical separation protects sensitive data, such as biometric information used for Face ID and Touch ID, from unauthorized access or tampering, adhering to the principle of isolation for critical security mechanisms.

Encapsulation

It can be viewed as a specific form of isolation based on object-oriented functionality. Protection is provided by encapsulating a collection of procedures and data objects in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the protected subsystem, and the procedures may be called only at designated domain entry points.

Example: In a healthcare database, encapsulation is used to enhance security by restricting direct access to sensitive patient data. Instead of allowing users to query the tables directly, access is controlled through stored procedures and views. For example, a stored procedure GetPatientRecord retrieves patient records based on specific criteria, ensuring that only authorized information is returned, while a view PublicPatientInfo provides access to non-sensitive data like names and dates of birth. This encapsulation ensures that all interactions with sensitive data go through well-defined, controlled entry points, enforcing access control policies and business rules, maintaining data integrity, and enabling effective auditing and logging. By using these methods, the database protects the internal structure and sensitive information from unauthorized access or tampering.

Modularity

It refers both to the development of security functions as separate, protected modules and to the use of a modular architecture for mechanism design and implementation.

With respect to the use of separate security modules, the design goal here is to provide common security functions and services, such as cryptographic functions, as common modules. For example, numerous protocols and applications make use of cryptographic functions. Rather than implementing such functions in each protocol or application, a more secure design is provided by developing a common cryptographic module that can be invoked by numerous protocols and applications. The design and implementation effort can then focus on the secure design and implementation of a single cryptographic module and including mechanisms to protect the module from tampering. With respect to the use of a modular architecture, each security mechanism should be able to support migration to new technology or upgrade of new features without requiring an entire system redesign. The security design should be modular so that individual parts of the security design can be upgraded without the requirement to modify the entire system.

Example: Amazon Web Services (AWS) Key Management Service (KMS) exemplifies modularity in security design by providing a centralized, managed cryptographic module that various AWS services and applications can use for encryption and key management. Instead of each AWS service implementing its own cryptographic functions, they rely on KMS to perform encryption, decryption, and key management tasks. This centralization ensures consistent and secure encryption across services such as S3, EBS, RDS, and Lambda, while robust security measures protect the module from tampering and unauthorized access. The modular architecture allows AWS to update or enhance KMS's cryptographic capabilities without disrupting the dependent services, ensuring that all services benefit from the latest security standards and features.

Layering

It refers to the use of multiple, overlapping protection approaches addressing the people, technology, and operational aspects of information systems. By using multiple, overlapping protection approaches, the failure or circumvention of any individual protection approach will not leave the system unprotected. A layering approach is often used to provide multiple barriers between an adversary and protected information or services. This technique is often referred to as defense in depth.

Example: Most of today's corporate network design follows this principle. The key terms here are "multiple overlapping protection" and "people, technology, and operational aspects of information system". Here is a hypothetical example.

Layering (Defence in Depth) Example

Consider a company using a multi-layered defense strategy to protect its computers:

User Authentication: Multi-factor authentication (MFA) is required to log into the OS.

Access Control: Users are granted access based on their roles, ensuring they only access necessary files and applications.

Patch Management: The OS is regularly updated with security patches to fix vulnerabilities.

Antivirus Software: The OS runs antivirus software to detect and remove malware.

Firewalls: Both network and host-based firewalls are in place to monitor and control incoming and outgoing traffic.

Encryption: Sensitive data is encrypted to protect it from unauthorised access.

Security Training: Employees are trained on recognising phishing attempts and safe usage.

For instance, even if a phishing attack manages to steal a user's password, the attacker would still need to bypass MFA and other security layers like access controls and antivirus protection to compromise the OS.

Least astonishment

A program or user interface should always respond in the way that is least likely to astonish the user. For example, the mechanism for authorization should be transparent enough to a user that the user has a good intuitive understanding of how the security goals map to the provided security mechanism.

Example: A programming language's standard library usually provides a function similar to the pseudocode `ParseInteger(string, radix)`, which creates a machine-readable integer from a string of human-readable digits. The radix conventionally defaults to 10, meaning the string is interpreted as decimal (base 10). This function usually supports other bases, like binary (base 2) and octal (base 8), but only when they are specified explicitly. In a departure from this convention, JavaScript originally defaulted to base 8 for strings beginning with "0", causing developer confusion and software bugs. This was discouraged in ECMAScript 3 and dropped in ECMAScript 5 [3].

7 Security Attacks

A security attack is any action that compromises the security of information owned by an organization. Equally, you can say a security attack is any event or action that compromises the security goals of information or information systems owned by an organisation. A useful means of classifying security attacks, used both in X.800 and RFC 4949, is in terms of **passive**

attacks and **active attacks**.

- **Passive attack** attempts to learn or make use of information from the system but does not affect system resources. Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two examples of passive attacks are the release of message contents and traffic analysis.
- **Active attack** attempts to alter system resources or affect their operation. Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into sub-categories such as masquerade, replay, modification of messages, and denial of service.

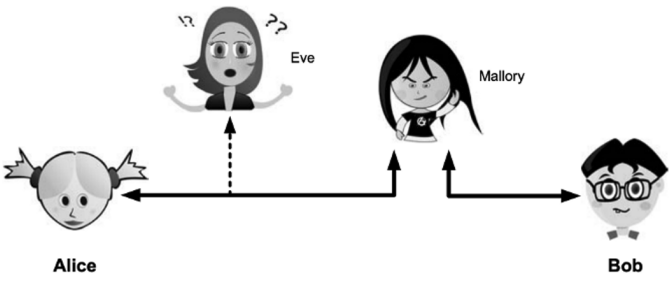

Diagram by TCP/IP Illustrated Volume 1 Second Edition The Protocols

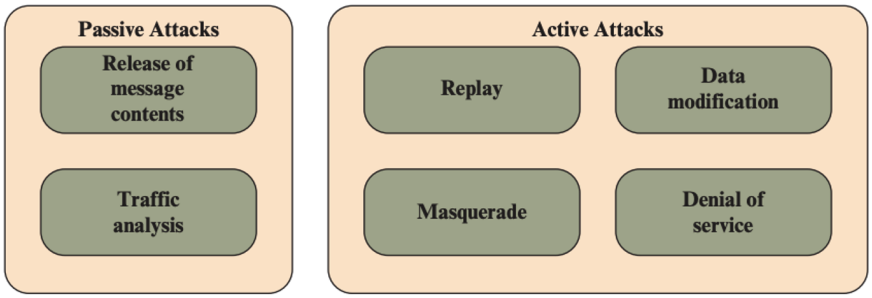

Figure 2: Security Attacks (**Figure sources:** Top [4], Bottom [5])

Common Actors in Cryptography

These characters help to personify the various roles and participants in cryptographic operations, making it easier to explain and understand complex concepts.

- **Alice (A):** The sender or originator of a message.
- **Bob (B):** The recipient or intended receiver of the message.
- **Eve (E):** An eavesdropper who tries to intercept and possibly decipher the communication between Alice and Bob.
- **Mallory (M):** A malicious attacker who attempts to intercept, alter, or disrupt communications.
- **Trent (T):** A trusted third party who might facilitate communication, such as a certification authority.

There are other personas such as Carol (C), Dave (D), Charlie (C), Peggy (P) and Victor (V), which we will not require in this unit.

7.1 Attack Vector

Attack Vector is a specific method or pathway through which a hacker or malicious actor gains unauthorized access to a computer or network system.

Examples:

1. **Phishing:** Using fraudulent emails or messages that appear to be from a trusted source to trick individuals into revealing sensitive information or clicking on malicious links.
2. **Malware:** Software designed to harm or exploit any programmable device, service, or network. Attackers can introduce malware through email attachments, software downloads, or operating system vulnerabilities.
3. **Person-in-the-Middle (PitM) Attacks:** Interception of the communication between two parties to steal or manipulate the data being exchanged.

7.2 Attack Surfaces

An attack surface consists of the reachable and exploitable vulnerabilities in a system.

Examples of attack surfaces are the following:

- Open ports on outward-facing web and other servers and code listening on those ports.
- Services available on the inside of a firewall
- Code that processes incoming data, email, XML, office documents, and industry-specific custom data exchange formats.
- Interfaces, SQL, and Web forms.

- An employee with access to sensitive information is vulnerable to a social engineering attack.

Attack surfaces can be categorised as follows:

- **Network attack surface:** This category refers to vulnerabilities over an enterprise network, wide-area network, or the Internet. Included in this category are network protocol vulnerabilities, such as those used for a denial-of-service attack, disruption of communications links, and various forms of intruder attacks.
- **Software attack surface:** This refers to vulnerabilities in application, utility, or operating system code. A particular focus in this category is Web server software.
- **Human attack surface:** This category refers to vulnerabilities created by personnel or outsiders, such as social engineering, human error, and trusted insiders.

An attack surface analysis is a useful technique for assessing the scale and severity of threats to a system. A systematic analysis of points of vulnerability makes developers and security analysts aware of where security mechanisms are required. Once an attack surface is defined, designers may be able to find ways to make the surface smaller, thus making the task of the adversary more difficult. The attack surface also provides guidance on setting priorities for testing, strengthening security measures, and modifying the service or application.

7.3 Attack Trees

An attack tree is a branching, hierarchical data structure that represents a set of potential techniques for exploiting security vulnerabilities.

The motivation for the use of attack trees is to effectively exploit the information available on attack patterns. Security analysts can use the attack tree to document security attacks in a structured form that reveals key vulnerabilities. The attack tree can guide both the design of systems and applications, and the choice and strength of countermeasures.

Figure 3 is an example of an attack tree analysis for an Internet banking authentication application. The root of the tree is the objective of the attacker, which is to compromise a user's account. The shaded boxes on the tree are the leaf nodes, which represent events that comprise the attacks.

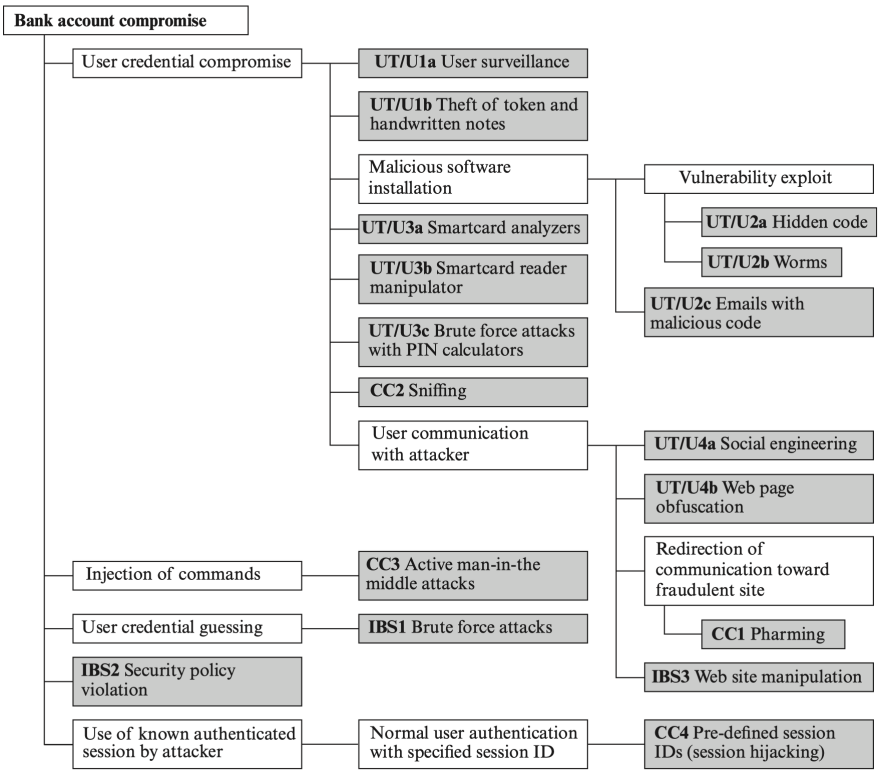

Figure 3: An Attack Tree for Internet Banking Authentication (**Figure Source** [5])

7.4 Who Is the Opponent

As we discussed under incentives in our security framework, understanding the nature and motivations of these attackers is crucial for developing effective defences. Here are some of the main categories of attackers:

- **Criminals:** These attackers include botnet herders, malware developers, spam senders, those involved in bulk account compromises, targeted attackers, cash-out gangs, and those who deploy ransomware. Their primary motivation is often financial gain.
- **Internal attacks:** Fraud and malicious activities carried out by insiders, such as employees or contractors, who misuse their access to an organization's resources.
- **Whistle-blowers:** Individuals within intelligence agencies and secretive firms who might reveal sensitive information, posing what is often referred to as the 'insider threat'.
- **Security Researchers:** This category includes researchers and academics who investigate vulnerabilities and report them to improve security. They often seek to advance knowledge out of curiosity and gain professional recognition, which can lead to career advancements.
- **Hacktivists:** Individuals or groups engaged in hacktivism, using hacking techniques to promote political ends, social change, or other ideological goals.
- **Script Kiddies:** Inexperienced individuals who use existing tools and scripts to launch attacks without a deep understanding of the underlying technology. Their actions can still cause significant harm despite their lack of expertise.
- And more!

8 Security is a Process

Security is a dynamic, continuous process, not merely a static product one can simply purchase. Historical evidence suggests that all products have inherent imperfections. Therefore, security cannot be achieved solely by building or buying a product; it requires a holistic approach that considers various factors, including the roles of users, administrators, and potential internal threats. System linkages add complexity to security management. Moreover, the ever-changing nature of business environments and the corresponding shift in attacker motivations call for frequent system updates and collaborations. The profound complexity of systems intended to be secured by "security products" further underscores that security is an evolving process, demanding constant vigilance and adaptation.

9 Practice Questions

Question 1: You are designing a small network with a storage server using safety/reliability engineering practices. Which one of the following is not part of safety/reliability engineering?

a) Taking measures to prevent someone from erasing data in the storage server.

b) Implement RAID-1 to ensure data is not lost because of disk failures.

c) Installing proper surge protection equipment for electrical safety.

d) Dimension the storage server so it can handle increased requests at the end of the financial year.

Explanation: *The correct answer is a). The key difference between safety/reliability engineering and security engineering is that, in security engineering, we take measures to circumvent actions by an adversary. In contrast to safety and reliability engineering, we take measures to prevent disruptions caused by known and predictable factors such as hardware failures. Answers b), c) and d) deal with possible hardware failures. On the other hand, a) plans for measure against an adversary.*

Question 2: Which of the following is FALSE about traditional security goals?

a) Privacy is my ability to control the visibility of my personal information.

b) Authentication is verifying an endpoint's identity to ensure we are talking to the right endpoint.

c) Accountability is the ability to trace some action or an event back to an actor or an originator.

d) Authorization ensures that an endpoint proves its identity before communicating with someone.

Explanation: *The correct answer is d). Authorisation refers to the effect that a verifier can determine whether an entity is allowed to execute some action or access some data. This can be done without Authentication, for example, by using authorisation tokens. Therefore d) is*

FALSE. What it describes is the definition of authentication.

Question 3: You are a security engineer for a healthcare company. The CISO of the company asks you to ensure that "we do not keep our customer information for more than three years, and whenever a customer requests us to delete their personal data, we must comply with that within three working days". What part of the conceptual framework we studied was the CISO referring to?

a) Assurance

b) Policy

c) Mechanisms

d) Incentives

Explanation: *This question is related to the security framework we discussed. The definition of policy is we define at a high level what it means by security to us (i.e., the company). In this case, the CISO (Chief Information Security Officer) is defining a policy. Note that when we define policy, we don't stipulate how to implement the policy. That comes under mechanisms. Therefore the correct answer is b) Policy.*

Question 4: Mary is helping a computer user who sees the following message appear on their computer. What security goal has been breached here?

Figure 4: Mary's screen

a) Malleability

b) Accountability

c) Availability

d) Non-repudiation

Explanation: *The screenshot is a typical message that is displayed under ransomware attacks. In case it appears that all the files in the computer have been encrypted and, as a result, are not accessible. Also, means the user will not be able to do their usual work on the computer. Therefore the correct answer is c) Availability. Also, note that a) Malleability is not even a security goal.*

Question 5: A shipping company is implementing new controls for its accounting department. Management is concerned that a rogue accountant may be able to create a new false vendor and then issue checks to that vendor as payment for services that were never rendered. What "security design principle" can best help to prevent this situation?

a) Economy of mechanism

b) Open design

c) Psychological acceptability

d) Separation of privilege

Explanation: *The correct answer is d) Separation of privilege. When following the Separation of Privilege principle, organisations divide critical tasks into discrete components and ensure that no individual can perform both actions. This prevents a single rogue accounting from creating the vendor and making a payment.*

"Open design" refers to the fact that a security mechanism's design should be open rather than secret.

"Economy of mechanism" means that the design of security measures embodied in both hardware and software should be as simple and small as possible.

"Psychological acceptability" refers to implies that the security mechanisms should not interfere unduly with the work of users while at the same time meeting the needs of those who authorise access.

None of them is directly related to the given scenario.

Question 6: Which of the following are examples of passive attacks? Select all that apply.

a) The attacker is recording the keyboard sounds of a target user and tries to infer what keys they might be pressing.

b) The attacker is conducting a person-in-the-middle attack and attempts to steal the bank login credential of a target user.

c) The attacker uses a WiFi jammer to disturb the WiFi communication in a building.

d) The attacker monitors the power consumption of a CPU during cryptographic operations and tries to infer cryptographic keys.

Explanation: *The correct answers are a) and d). In both cases, the attacker is not interfering with what the user/device is doing. Rather they monitor some artefacts of the main task and try to infer what is happening.*

In "person-in-the-middle attacks", the attacker acts as a proxy, accepts the source's messages, modifies them, and forwards them to the destination and vice versa. As the attacker interferes with the message flow between the source and the destination, this is an active attack. We will learn more about person-in-the-middle attacks later in this course.

Question 7: ..... is a path or a technique by which someone gains illegal access to a computer system.

a) Attack vector

b) Attack surface

c) Attack point

d) Attack arena

Explanation: *The correct answer is a) Attack vector. An attack vector is one means of breaking security goals associated with a system. For example, if you think about breaking into a computer, password brute force is an attack vector. The sum of all such attack vectors is called the attack surface.*

The other two, "attack point" and "attack arena", don't have any particular meaning. They are just made-up terminology.

Question 8: If you consider a web application, which one of the following can be considered an attack vector? Select all that apply.

a) Compromised credentials

b) Misconfigurations

c) Phishing

d) Zero-day vulnerabilities

Explanation: All of them are correct.

"Compromised credentials" describe a case where user credentials, such as usernames and passwords, are exposed to unauthorised entities. Without any multi-factor authentication,

compromised credentials can be used to obtain unauthorised access.

"Misconfigurations" are when there is an error in system configuration. For example, if set-up pages are enabled or a user uses default usernames and passwords, this can lead to breaches.

"Phishing" is one of the most commonly exploited attack vectors. It is a vector in which the targets are contacted by email, telephone or text message by someone posing as a legitimate institution to lure individuals into providing sensitive data such as personally identifiable information, banking and credit card details, and passwords. In this particular case, the user or admin of the web application can be the possible target.

"Zero-day vulnerability" is a vulnerability that nobody is aware of until the breach happens (hence the name zero-day, as there is no time elapsed between when the attack happens and the vulnerability is made public). If a developer has not released a patch for the zero-day vulnerability before a hacker exploits that vulnerability, then the following attack is known as a zero-day attack.

Question 9: Jerry is investigating an attack where the attacker stole an authentication token from a user's web session and used it to impersonate the user on the site. What term best describes this attack?

a) Masquerading

b) Denial of Service

c) Traffic Analysis

d) Modification

Explanation: *Correct answer is a) Masquerading. A masquerade occurs when one entity pretends to be a different entity (path 2 of Figure 1.2b is active). A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorised entity with few privileges to obtain extra privileges by impersonating an entity with those privileges.*

Denial of Service usually means flooding a service with a barrage of requests so that it is overloaded to handle legitimate requests.

Traffic analysis is a passive attack where the attacker observes encrypted network traffic and tries to infer what kind of activities are happening or what content is being transferred using statistical properties of traffic flows. The attacker does not try to decrypt the traffic.

Modification simply means that some portion of a legitimate message is altered or that messages are delayed or reordered to produce an unauthorised effect.

Question 10: ..... is the sum of all the possible points in the software or the system where unauthorised users can enter as well as extract data from the system.

a) Attack vector

b) Attack surface

c) Attack point

d) Attack arena

Explanation: *The correct answer is b). The attack surface consists of all a system's reachable and exploitable vulnerabilities. In other words, it is the sum of all the possible points that can be used to exploit a system.*

One such means is called an attack vector. For example, if you think about breaking into a computer, password brute force is an attack vector. The attack surface, in this case, is highly complex and involves many aspects, such as password brute-forcing, physical security of the location where the computer is hosted, whether the computer downloads files from the internet etc.

The other two, "attack point" and "attack arena", don't have any particular meaning. They are just made-up terminology.

References

[1] Ross Anderson and Roger Needham. Programming satan's computer. *Computer Science Today: Recent Trends and Developments*, pages 426–440, 2005.

[2] Ross Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2020.

[3] Wikipedia contributors. Principle of least astonishment — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Principle_of_least_astonishment, 2024. [Online; accessed 24-June-2024].

[4] Kevin R Fall and W Richard Stevens. *Tcp/ip illustrated*, volume 1. Addison-Wesley Professional, 2012.

[5] William Stallings. *Cryptography and Network Security: Principles and Practice, Global Edition*. Pearson, Upper Saddle River, NJ, 8th edition, 2022.