Security and Privacy

Inf2-SEPP

Adriana Sejfia

School of Informatics, University of Edinburgh

Previous lecture

- Usability
- Interaction design

This Lecture

- Security
  - Definition and relationship to other non-functional requirements
  - Why is it important?
  - Types of security attacks
  - Some attacks and defences
  - Authentication, Authorization, Encryption
  - Design guidelines for security
- Privacy
  - Definition and relationship to security
  - Why is it important?
  - Data protection laws
  - Privacy policy

Security

What is Security?

- "A system attribute that reflects the ability of the system to protect itself from malicious internal and external attacks." (Sommerville SE)

What is Security?

- "A system attribute that reflects the ability of the system to protect itself from malicious internal and external attacks." (Sommerville SE)
- Part of dependability – the ability to provide services that can defensibly be trusted within a time-period

What is Security?

- "A system attribute that reflects the ability of the system to protect itself from malicious internal and external attacks." (Sommerville SE)
- Part of dependability – the ability to provide services that can defensibly be trusted within a time-period
- Related attributes of dependability:
  - Reliability – likelihood to run without failure for a given period
  - Availability – proportion of time when available for use
  - Safety – ability to operate without catastrophic failure
  - Resilience – ability to recover

What is Security?

- "A system attribute that reflects the ability of the system to protect itself from malicious internal and external attacks." (Sommerville SE)
- Part of dependability – the ability to provide services that can defensibly be trusted within a time-period
- Related attributes of dependability:
  - Reliability – likelihood to run without failure for a given period
  - Availability – proportion of time when available for use
  - Safety – ability to operate without catastrophic failure
  - Resilience – ability to recover
- Also closely related to Privacy

Why is Security Important?

- In the 1990s, the Internet led to proliferation of attacks on online systems

Why is Security Important?

- In the 1990s, the Internet led to proliferation of attacks on online systems
- Effects of security breaches:
  - Unhappy customers
  - Legal action
  - Loss of business revenue – even getting out of business
  - Recovery taking a lot of time and effort

Why is Security Important?

- In the 1990s, the Internet led to proliferation of attacks on online systems
- Effects of security breaches:
  - Unhappy customers
  - Legal action
  - Loss of business revenue – even getting out of business
  - Recovery taking a lot of time and effort
- Human, social and organisational failings can cause vulnerabilities (e.g. easy-to-guess passwords)

Why is Security Important?

- In the 1990s, the Internet led to proliferation of attacks on online systems
- Effects of security breaches:
  - Unhappy customers
  - Legal action
  - Loss of business revenue – even getting out of business
  - Recovery taking a lot of time and effort
- Human, social and organisational failings can cause vulnerabilities (e.g. easy-to-guess passwords)
- Such failings are often a sign of poor system decisions and problems in SE activities

Types of Security Attacks

- Interception – attacker gaining access to an asset on the system

Types of Security Attacks

- Interception – attacker gaining access to an asset on the system
- Interruption – attacker making a system unavailable

Types of Security Attacks

- Interception – attacker gaining access to an asset on the system
- Interruption – attacker making a system unavailable
- Modification – attacker tampering with a system asset

Types of Security Attacks

- Interception – attacker gaining access to an asset on the system
- Interruption – attacker making a system unavailable
- Modification – attacker tampering with a system asset
- Fabrication – attacker inserting false information into the system

Attacks and Defences

Injection Attacks (Type: Fabrication)

- Attacker inputs malicious code or database commands using an input field

Injection Attacks (Type: Fabrication)

- Attacker inputs malicious code or database commands using an input field
- Examples:
  - Buffer overflow (C/C++) – inputting a string with executable instructions that overwrites memory and transfers control to malicious code

Buffer overflow example

```c
#include <stdio.h>
#include <string.h>

void secret() {
    printf("You've hijacked execution!\n");
}

void vulnerable(char *input) {
    char buf[16];       // Only 16 bytes allocated
    strcpy(buf, input); // No bounds check — dangerous!
    printf("Hello, %s\n", buf);
}

int main(int argc, char *argv[]) {
    vulnerable(argv[1]);
    return 0;
}
```

Injection Attacks (Type: Fabrication)

- Attacker inputs malicious code or database commands using an input field
- Examples:
  - Buffer overflow (C/C++) – inputting a string with executable instructions that overwrites memory and transfers control to malicious code
  - SQL poisoning attacks – inputting part of an SQL command to obtain confidential information

SQL injection example

query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'"

SELECT * FROM users WHERE username = 'alice' AND password = 'secret'

SQL injection example

query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'"

SELECT * FROM users WHERE username = 'alice' AND password = 'secret'

Attacker input for the username: admin' --

SQL injection example

query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'"

SELECT * FROM users WHERE username = 'alice' AND password = 'secret'

Attacker input for the username: admin ' --

SELECT * FROM users WHERE username = ' admin ' --
(anything after "--" does not get executed)

Injection Attacks (Type: Fabrication)

- Attacker inputs malicious code or database commands using an input field
- Examples:
  - Buffer overflow (C/C++) – inputting a string with executable instructions that overwrites memory and transfers control to malicious code
  - SQL poisoning attacks – inputting part of an SQL command to obtain confidential information
- Defence: input validation

Cross-Site Scripting Attacks (Type: Fabrication)

- Another form of injection attack
- Attacker adds malicious JavaScript to a webpage (e.g. via a web form)

Cross-Site Scripting Attacks (Type: Fabrication)

- Another form of injection attack
- Attacker adds malicious JavaScript to a webpage (e.g. via a web form)
- When loaded by a client may:
  - Steal user information
  - Direct user to another website to capture personal information or display ads
  - Facilitate session hijacking attacks

Cross-Site Scripting Attacks (Type: Fabrication)

- Another form of injection attack
- Attacker adds malicious JavaScript to a webpage (e.g. via a web form)
- When loaded by a client may:
  - Steal user information
  - Direct user to another website to capture personal information or display ads
  - Facilitate session hijacking attacks
- Defences: input validation, check database inputs before rendering in webpage, HTML ‘encode’ command to make content non-executable

Session Hijacking Attacks (Type: Interception)

- Attacker steals a session cookie to impersonate a user
  - Active: carry out user actions
  - Passive: monitor user to detect valuable information

Session Hijacking Attacks (Type: Interception)

- Attacker steals a session cookie to impersonate a user
  - Active: carry out user actions
  - Passive: monitor user to detect valuable information
- Can be achieved through:
  - Cross-site scripting
  - Traffic monitoring (server–client)

Session Hijacking Attacks (Type: Interception)

- Attacker steals a session cookie to impersonate a user
  - Active: carry out user actions
  - Passive: monitor user to detect valuable information
- Can be achieved through:
  - Cross-site scripting
  - Traffic monitoring (server–client)
- Likelihood reduced by:
  - Traffic encryption (HTTPS)
  - Multi-factor authentication
  - Short session timeouts

Denial-of-Service Attacks (Type: Interruption)

- Attacker makes the system unavailable to users

Denial-of-Service Attacks (Type: Interruption)

- Attacker makes the system unavailable to users
- Types:
  - Distributed DoS (DDoS) – sends large volume of requests to a web application, denying access to legitimate users → Defence: specialist software

Denial-of-Service Attacks (Type: Interruption)

- Attacker makes the system unavailable to users
- Types:
  - Distributed DoS (DDoS) – sends large volume of requests to a web application, denying access to legitimate users → Defence: specialist software
  - User lockout attacks – exploits account-lockout policy using the user's email as username → Defence: temporary lockouts, IP address tracking and locking

Brute Force Attacks (Type: Interception)

- Attacker repeatedly generates passwords using known username to gain access

Brute Force Attacks (Type: Interception)

- Attacker repeatedly generates passwords using known username to gain access
- Vulnerability factors:
  - No account lockout after failed validation
  - Users choosing weak passwords

Brute Force Attacks (Type: Interception)

- Attacker repeatedly generates passwords using known username to gain access
- Vulnerability factors:
  - No account lockout after failed validation
  - Users choosing weak passwords
- Solutions:
  - Lock accounts after failed validation (but may attract DoS attacks)
  - Insist on strong passwords
  - Two-factor authentication

Defences in Detail

Authentication

- Process of verifying that a system user is who they claim to be

Authentication

- Process of verifying that a system user is who they claim to be
- Approaches:
  - Knowledge-based – secret personal information (e.g. passwords)

Authentication

- Process of verifying that a system user is who they claim to be
- Approaches:
  - Knowledge-based – secret personal information (e.g. passwords)
  - Possession-based – physical device (e.g. mobile phone)

Authentication

- Process of verifying that a system user is who they claim to be
- Approaches:
  - Knowledge-based – secret personal information (e.g. passwords)
  - Possession-based – physical device (e.g. mobile phone)
  - Attribute-based – biometric (e.g. fingerprint, face recognition, retina scan)

Authentication

- Process of verifying that a system user is who they claim to be
- Approaches:
  - Knowledge-based – secret personal information (e.g. passwords)
  - Possession-based – physical device (e.g. mobile phone)
  - Attribute-based – biometric (e.g. fingerprint, face recognition, retina scan)
- Multi-factor authentication combines two or more of the above for increased security

Authentication via Federated Identity

- "Login with Google", "Login with Facebook", etc. – using an external system for authentication

Authentication via Federated Identity

- "Login with Google", "Login with Facebook", etc. – using an external system for authentication
- Advantages for users:
  - Single set of credentials
  - Credentials stored in a single place (more secure)

Authentication via Federated Identity

- "Login with Google", "Login with Facebook", etc. – using an external system for authentication
- Advantages for users:
  - Single set of credentials
  - Credentials stored in a single place (more secure)
- Some people have privacy concerns

Authentication via Federated Identity

- "Login with Google", "Login with Facebook", etc. – using an external system for authentication
- Advantages for users:
  - Single set of credentials
  - Credentials stored in a single place (more secure)
- Some people have privacy concerns
- Advantages for businesses:
  - No need to maintain databases with user information
  - Potentially access additional user information from the identity provider (with users' approval)

Authorization (for Multiuser Systems)

- Using users' identity to control access to system resources

Authorization (for Multiuser Systems)

- Using users' identity to control access to system resources
- Business systems define user rights through an access control policy

Authorization (for Multiuser Systems)

- Using users' identity to control access to system resources
- Business systems define user rights through an access control policy
- Policy is important:
  - Legal perspective – conforming to data protection rights
  - Technical perspective – defines the access control scheme for the system

Authorization (for Multiuser Systems)

- Using users' identity to control access to system resources
- Business systems define user rights through an access control policy
- Policy is important:
  - Legal perspective – conforming to data protection rights
  - Technical perspective – defines the access control scheme for the system
- Most file and database systems use access control lists – tables linking users with resources and permitted actions

Encryption

- "Process of making a document unreadable by applying an algorithm transformation including a secret key to it" (Sommerville ESP)

Encryption

- "Process of making a document unreadable by applying an algorithm transformation including a secret key to it" (Sommerville ESP)
- Practically uncrackable – but evolves with technology

Encryption

- "Process of making a document unreadable by applying an algorithm transformation including a secret key to it" (Sommerville ESP)
- Practically uncrackable – but evolves with technology
- Types:
  - Symmetric – same key used for encoding and decoding; problem: securely sharing the key
  - Asymmetric – each user has a public and a private key; encrypt with one, decrypt with the other

Encryption

- "Process of making a document unreadable by applying an algorithm transformation including a secret key to it" (Sommerville ESP)
- Practically uncrackable – but evolves with technology
- Types:
  - Symmetric – same key used for encoding and decoding; problem: securely sharing the key
  - Asymmetric – each user has a public and a private key; encrypt with one, decrypt with the other
- Key management is essential for businesses – use key management systems, change keys regularly, timestamp them

Design Guidelines for Security

Design Guidelines for Security (1–5)

1. Base security decisions on an explicit security policy – high-level statement of 'what', not 'how'

Design Guidelines for Security (1–5)

1. Base security decisions on an explicit security policy – high-level statement of 'what', not 'how'
2. Use defence in depth – combine multiple security techniques in critical systems

Design Guidelines for Security (1–5)

1. Base security decisions on an explicit security policy – high-level statement of 'what', not 'how'
2. Use defence in depth – combine multiple security techniques in critical systems
3. Fail securely – when failure occurs, do not fall back to less secure procedures

Design Guidelines for Security (1–5)

1. Base security decisions on an explicit security policy – high-level statement of 'what', not 'how'
2. Use defence in depth – combine multiple security techniques in critical systems
3. Fail securely – when failure occurs, do not fall back to less secure procedures
4. Balance security and usability – poor usability may cause users to undermine security

Design Guidelines for Security (1–5)

1. Base security decisions on an explicit security policy – high-level statement of 'what', not 'how'
2. Use defence in depth – combine multiple security techniques in critical systems
3. Fail securely – when failure occurs, do not fall back to less secure procedures
4. Balance security and usability – poor usability may cause users to undermine security
5. Log user actions – aids failure recovery, attack detection and acts as a deterrent

Design Guidelines for Security (6–10)

6. Use redundancy and diversity to reduce risk – multiple versions of system/data, avoid single-platform dependency

Design Guidelines for Security (6–10)

6. Use redundancy and diversity to reduce risk – multiple versions of system/data, avoid single-platform dependency
7. Specify the format of system inputs

Design Guidelines for Security (6–10)

6. Use redundancy and diversity to reduce risk – multiple versions of system/data, avoid single-platform dependency
7. Specify the format of system inputs
8. Compartmentalise your assets – restrict access to information using compartments

Design Guidelines for Security (6–10)

6. Use redundancy and diversity to reduce risk – multiple versions of system/data, avoid single-platform dependency
7. Specify the format of system inputs
8. Compartmentalise your assets – restrict access to information using compartments
9. Design for deployment – support deployment to avoid human error introducing vulnerabilities

Design Guidelines for Security (6–10)

6. Use redundancy and diversity to reduce risk – multiple versions of system/data, avoid single-platform dependency
7. Specify the format of system inputs
8. Compartmentalise your assets – restrict access to information using compartments
9. Design for deployment – support deployment to avoid human error introducing vulnerabilities
10. Design for recovery – assume failure can always occur; plan to restore to secure operational state

Privacy

What is Privacy?

- "Social concept that relates to the collection, dissemination and appropriate use of personal information held by a third party" (Sommerville ESP)

What is Privacy?

- "Social concept that relates to the collection, dissemination and appropriate use of personal information held by a third party" (Sommerville ESP)
- Relationship with security:
  - Privacy requires security, but security does not guarantee privacy (e.g. Facebook)
  - Not all security attacks breach privacy (e.g. denial-of-service attacks)

Why is Privacy Important?

- Legal action from customers or data regulators if privacy regulations are not met
- Business customers require privacy guarantees
- Leaking personal information can ruin a company's reputation and lose its customers

Data Protection Laws

- Many countries have laws limiting the collection, dissemination and use of personal data (e.g. GDPR for companies holding data on EU citizens)

Data Protection Laws

- Many countries have laws limiting the collection, dissemination and use of personal data (e.g. GDPR for companies holding data on EU citizens)
- Central notions:
  - Data subjects (individuals) have the right to: access data, correct errors, consent to its use, require deletion
  - Data controllers (data managers) are responsible for: storing data securely in a location covered by legislation, providing subjects access, using data only for consented purposes

Privacy Policy

- Legal document outlining how personal and sensitive information is collected, stored and managed

Privacy Policy

- Legal document outlining how personal and sensitive information is collected, stored and managed
- Should:
  - Not be buried in long 'terms and conditions' – must be accessible to users
  - Be auditable to check conformance with data protection laws in each market
  - Be reviewable by users, with the ability to opt out of storing different types of information

Reading

- Essential:
  - Sommerville ESP – Chapter 7: Security and Privacy
  - Sommerville SE – Chapter 13: Security Engineering (Introduction, §13.1, §13.4)
- Recommended:
  - Remainder of Sommerville SE Chapter 13: Security Engineering

Takeaways from the course

- Remember that your system will evolve!

Takeaways from the course

- Remember that your system will evolve!
- Try to always reduce your and your team member's cognitive load.

Takeaways from the course

- Remember that your system will evolve!
- Try to always reduce your and your team member's cognitive load.
- Spend as much effort in understanding what your users want as in how to build and verify.

Takeaways from the course

- Remember that your system will evolve!
- Try to always reduce your and your team member's cognitive load.
- Spend as much effort in understanding what your users want as in how to build and verify.
- Trust your code, but always, always verify it!

Takeaways from the course

- Remember that your system will evolve!
- Try to always reduce your and your team member's cognitive load.
- Spend as much effort in understanding what your users want as in how to build and verify.
- Trust your code, but always, always verify it!
- Personal: stay in touch! asejfia@ed.ac.uk