MSc - Cybersecurity

CMT310: Developing Secure Systems and Applications

Security and Privacy Goals
Poor and Secure Coding Practices

Dr Neetesh Saxena

saxenan4@cardiff.ac.uk

Security Goals and Properties

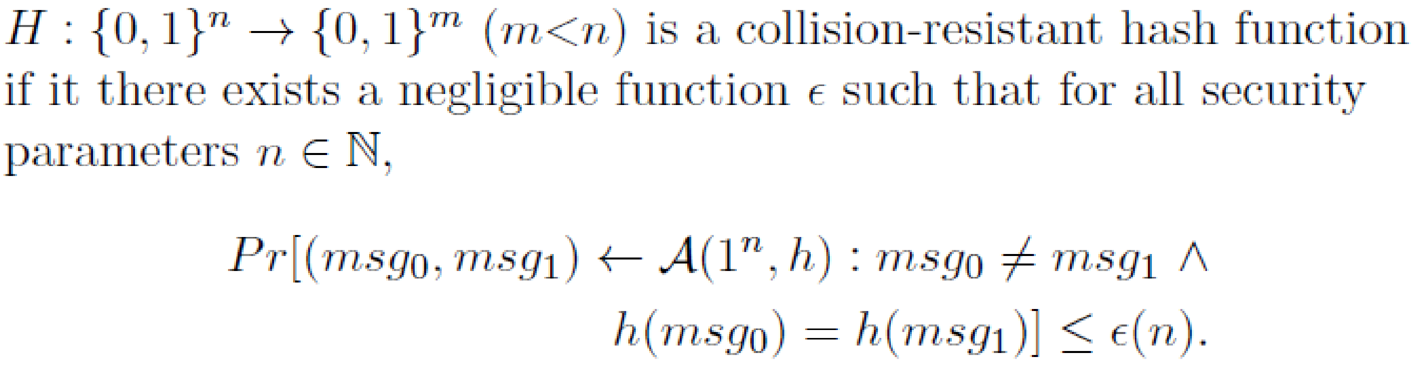
- Mutual Authentication
  - The server must authenticate the user and the user should be able to verify that it is connected to the legitimate server.
  - Defeats the redirection and impersonation attacks.

- Perfect Forward Secrecy (PFS)
  - A scheme maintains PFS if no adversary A can retrieve the past session keys, even the long term keys LTK (i.e., the private key of the user or a session key) are compromised.

Security Goals and Properties

- Information Confidentiality
  - Encrypted message sent by the user must be indistinguishable from a randomly generated messages, and supports Indistinguishability under Chosen Plaintext Attack (IND-CPA).

- Message Integrity
  - Integrity of each message can be achieved using a well known Collision-Resistant Hash Functions (CRHF).



$H : \{0,1\}^n \to \{0,1\}^m$ $(m{<}n)$ is a collision-resistant hash function if it there exists a negligible function ϵ such that for all security parameters $n \in \mathbb{N}$,

$$Pr[(msg_0, msg_1) \leftarrow \mathcal{A}(1^n, h) : msg_0 \neq msg_1 \wedge h(msg_0) = h(msg_1)] \leq \epsilon(n).$$

Privacy Goals and Properties

- Untraceability
  - Untraceability is maintained if A cannot distinguish whether two generated messages correspond to the same or two different identities of the users.
    - Forward untraceability
    - Backward untraceability

- Forward Privacy
  - Similar to untraceability with additional capability.
  - One of two messages is given to adversary A. Clearly, now A can trace the user's identity and/or other information.
  - Forward privacy is maintained if A is still unable to trace previous sessions (without giving a secret or session key).

- Anonymity is maintained if only the sender and the intended receiver can know the actual identity of the user.

Poor Coding Practices - Software Development

- Typos in your code
- Failing to Indent or format your code
- Failing to modularize your code
- Write a function that one thing only
- IDE – put you not into a false sense of security
- Hard-coding passwords and other codes
- Failing to use good encryption to protect data
- Failing to think scalability
- Not knowing how to optimize
- Ignoring error messages and not catching errors/exceptions
- Writing tests to pass
- Disregarding performance testing for critical cases
- Not commenting your code
- Not using version control
- Allowing unused code to remain in a system
- Naming functions, variables, files and classes poorly
- Working on your own all the time
- Coding before design

*this topic is under secure programming practices

OWASP Secure Coding Principles

OWASP Secure Coding Principles

- Minimize attack surface area
- Establish secure defaults
- Principle of Least privilege
- Principle of Defence in depth
- Fail securely
- Don’t trust services
- Separation of duties
- Avoid security by obscurity
- Keep security simple
- Fix security issues correctly

Security Principles

- Establish a security process
  - Define the product security goals
  - Consider security as a product feature
  - Learn from mistakes
  - Assume external systems are insecure
  - Remember that security features != secure features

*this topic is under secure programming practices