# Requirements

### Stakeholder

Demands software(funder)

### Developer

Implements the software

### Analyst

Communicates with both parties to their respective level of understanding

# The forms of data

### Data in transit

The state where the data is mobile.

### Data in processing

Typcially the state where data is being worked on utilizing the computer components(ram cpu etc)

### Data at rest

The state where the data is stored/steady. (In a database, in a HDD)

### Structured Data

Database (in tables as rows or columns)

### Unstructured Data

A shared folder where you have documents, images and other files

### Data lifecycle

Create -> Store -> Use -> Share -> Archive -> Destroy

### Confidentiality Requirement

Prevent unauthorized disclosure of classified data

### Integrity requirement

### Data Integrity

Assuring that data is not altered by unauthorized entities(CRC checks hashing)

### System Integrity

Assuring that the software that maintains data is not altered by unauthorized entities

(Input validation against injections)

### Availability requirement

Maintain business continuity – ex. Downtime per year

### Authentication requirement

Validate entity identity

### Authorization requirement

Requirements that aim to ensure what can and can’t be done by an entitiy

### Accountability requirement

Provide historical records of entitiy interactions for auditing purposes

### General Requirements

Session management

Error & Exception Management (Try catch)

Configuration parameter management(Database)

### Operational Requirements

Deployment Environment Requirements

Archiving Requirements

Anti-Piracy Requirements

### Other Requirements

### Sequencing and timing

Software behavior at a lower level

TOCTOU

Race conditions

### International (or External) Requirement

Legal and technological sanctions and constraints

HIPAA, SOX, ISO

Character encoding, regional settings

### Procurement Requirements

Decisions regarding buying the software or a module of it rather than developing from scratch.

Does it make sense to develop card reader software for the authentication requirements of our software?

# It Security design

Skapa alltid ett säkerhetskrav per funktionellt krav.

### Security flaw

Design/Arkitektur defekter som kan leda till hål i säkerheten.(Death star)

### Security bug

Programmerings bugg som kan leda till hål i säkerheten.

# Confidentiality Design cryptography

### Symmetric algorithms

Samma nyckel för encryption/decryption

### Assymetric algorithms

Olika nycklar för encryption/decryption

One private and one public key, only the private can decrypt, allowing the data to be send around safely

# Integrity design

### Referential Integrity

Ser till att ingen data lämnas I ett öde läge när en förälder tas bort

### Resource Locking

Locks a record when it performs operations so that it cant be altered.

# Availability design

### Replication

Allows backup for the system

### Failover

Switch from active system/server etc to a standby system.

### Scalability

Ability for the system to grow without performance issues

**Vertical:**

Supports adding more disks/ram

**Horizontal:**

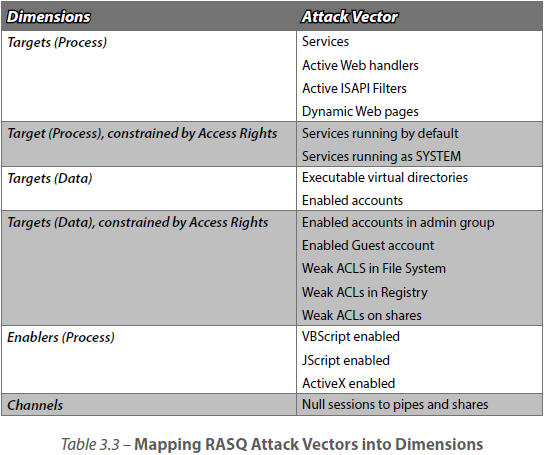
Supports adding more servers.

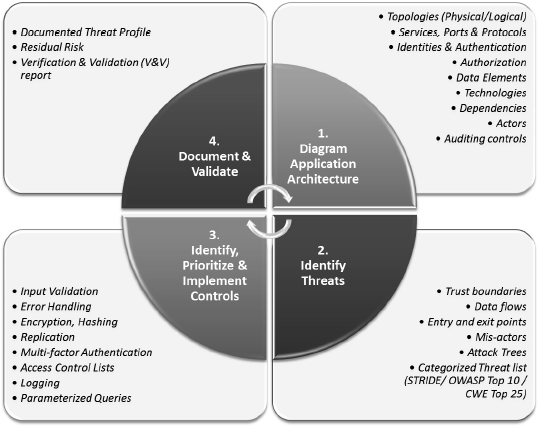
# Attack surface evaluation

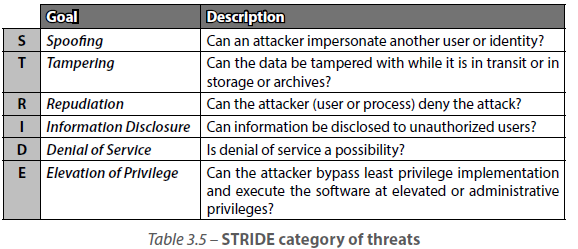
Evaluates entries / exits to the software that can lead to exploitation of weaknesses.

## Relative attack surface quotient (RASQ)

The likely opportunities for an attack compared to a baseline.







# Mainframe arhitecture

### Service Oriented Architecture (SOA)

Abstracted Business Functionality

Contract-Based Interfaces

# Rich Internet Applications (RIA)

Primary benefit – less workload on server side.

# Representational State Transfer (REST)

Supports media, JSON, xml etc.

Http methods (GET PUT POST DELETE).

# Cloud Computing

# Database Security

### Polyinstantiation

Depending on the grade of allowance the database returns different results.

-Encrypt Database

-Properly organized

# Programming language environment

### Unmanaged

Direcly compiled to the OS making is faster than managed.

Memory allocation not managed making it less secure against buffer overflow

= Programmer has to write all necessary routines to handle it.

### Managed

When compiling the code goes through a runtime enviornment (Common Intermediate Language (CIL)) into Just-in-Time (JIT) that transforms it into native code.

= not that many extra routines required but compiles at a slower rate.

### Compiler switches

/GS and StackGuard.

/GS creates a cookie if susceptible, if it’s an overflow the cookie gets destroyed and the argument won’t be written.

StackGuard creates a canary word if susceptible, if it’s an overflow the canary word gets overwritten and the argument won’t be written.

# Operating Systems

Address Space Layout Randomization (ASLR)

Randomizes the entry points (addresses) each time the program is run.