



# Protection and security of information systems

## Safety design

prof. Ph.D. Krešimir Fertalj

University of Zagreb

Faculty of Electrical Engineering and Computing

Protected by license <http://creativecommons.org/licenses/by-nc-sa/2.5/hr/>



# Creative Commons

---



## □ you are free to:

- **share**—reproduce, distribute and communicate the work to the public
- **remix**—rework the work

## □ under the following conditions:

- **attribution.** You must acknowledge and attribute the authorship of the work in a manner specified by the author or licensor (but not in a manner that suggests that you or your use of their work has their direct endorsement).
- **non-commercial.** You may not use this work for commercial purposes.
- **shares under the same conditions.** If you modify, transform, or create using this work, you may distribute the adaptation only under a license that is the same or similar to this one.

In the case of further use or distribution, you must make clear to others the license terms of this work. The best way to do this is to link to this website.

Any of the above conditions may be waived with the permission of the copyright holder.

Nothing in this license infringes or limits the author's moral rights.

The text of the license was taken from <http://creativecommons.org/>.

---

# Threat modeling

Threat Modeling

---

# Threat modeling

---

## -threat modeling

- security analysis that helps uncover the biggest security threats
- the goal is to determine which threats should be removed and how
- assumption - the product is not safe if the threats are not assessed and the risk is reduced

## -uses:

- better understanding of the application
  - especially new members

## -finding errors

- estimate that MP finds 50% of errors, and the rest by testing and analyzing the code
- complex application errors, which are rarely found otherwise (design errors)

# Principles and process of threat modeling

---

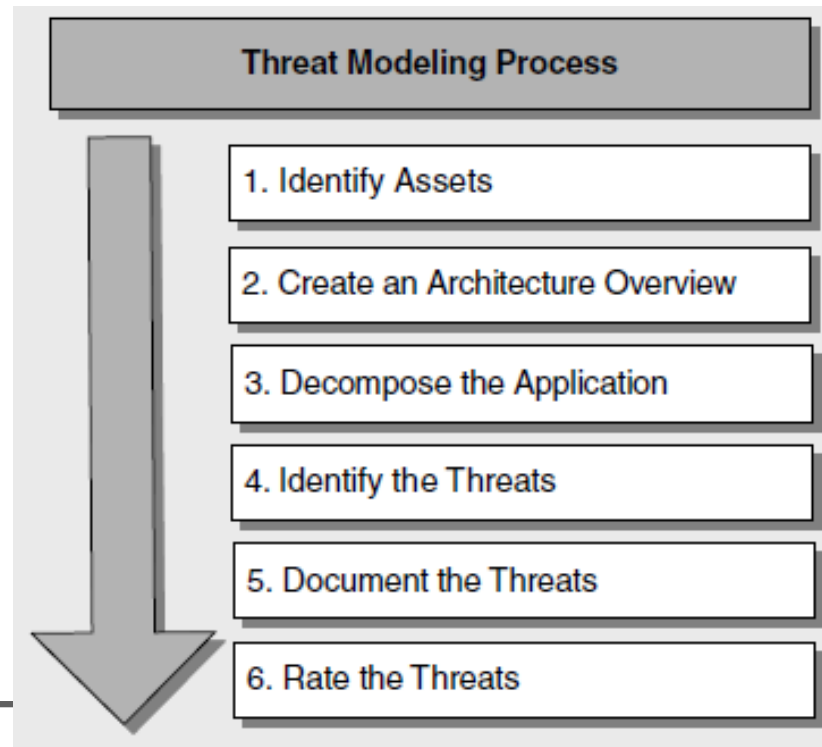
- Analyzing threats is a time-consuming job
  - it is important that it is done well
  - best iteratively

important:

- It's easier to find a security flaw in an application's design than to change it later
- The threat model should be current (up-to-date) - threats and ways to circumvent them

## -Threat modeling process

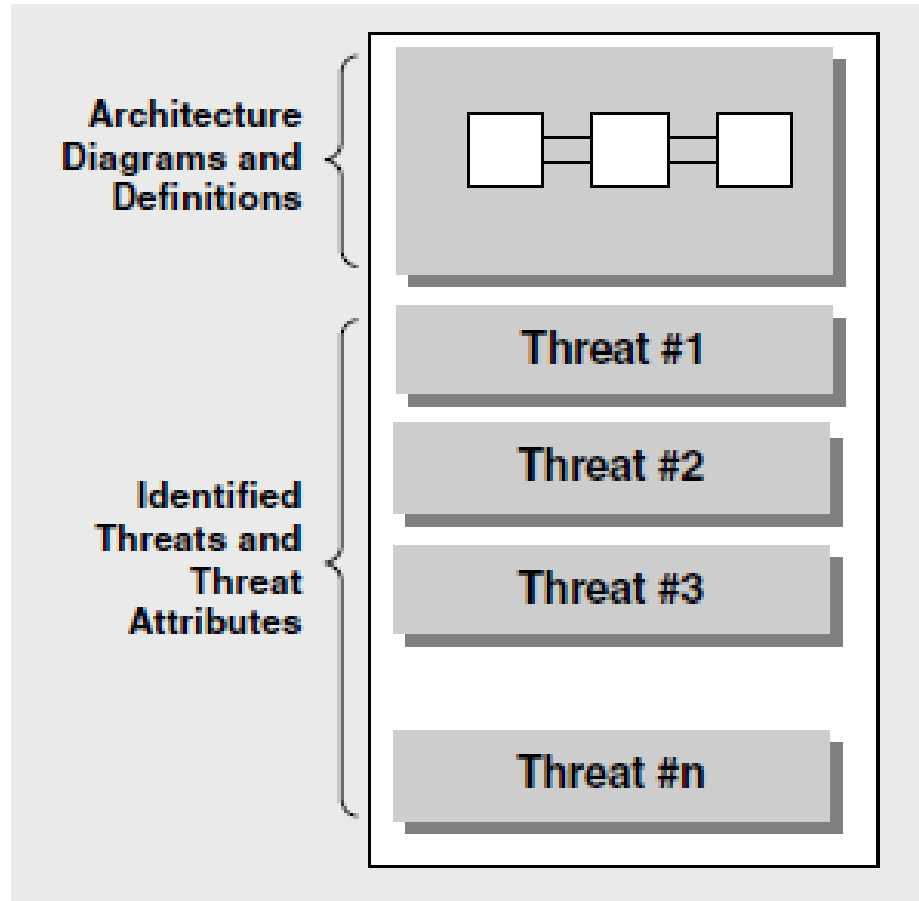
- Determination of protection objectives
- Application architecture
- Decomposition of the application
- Determining threats
- Documenting threats
- Threat ranking



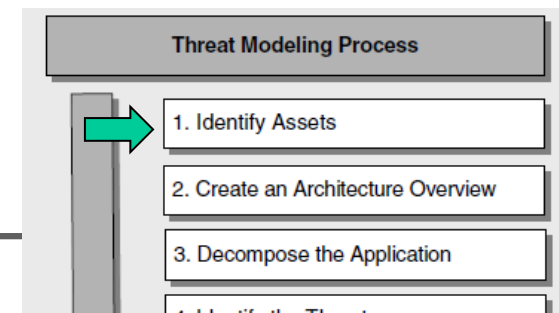
# Exit

## -Document with models

- by the definition of architecture and
- list of threats



- Step 1**–identification of resources to be protected
- from data repositories (file, BP), ..., to web pages



## Step 2 – Architecture Review

### -Documentation

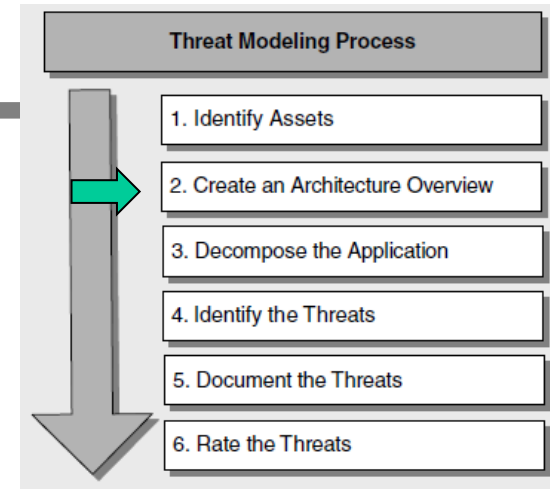
- application functions - what the application does
- application architecture and physical installation method (configuration)
- implementation technologies

### -Functionality modeling

- use cases (*use cases*)
- understanding how to use it
- the context of the application
- examples:
  - the employee sees business data, can update personal data,
  - the manager sees the employee's data.

### -Checking (violations) of business rules

- For example the user tries to change someone else's personal data
- He shouldn't do that if he doesn't have a sufficient level of permissions



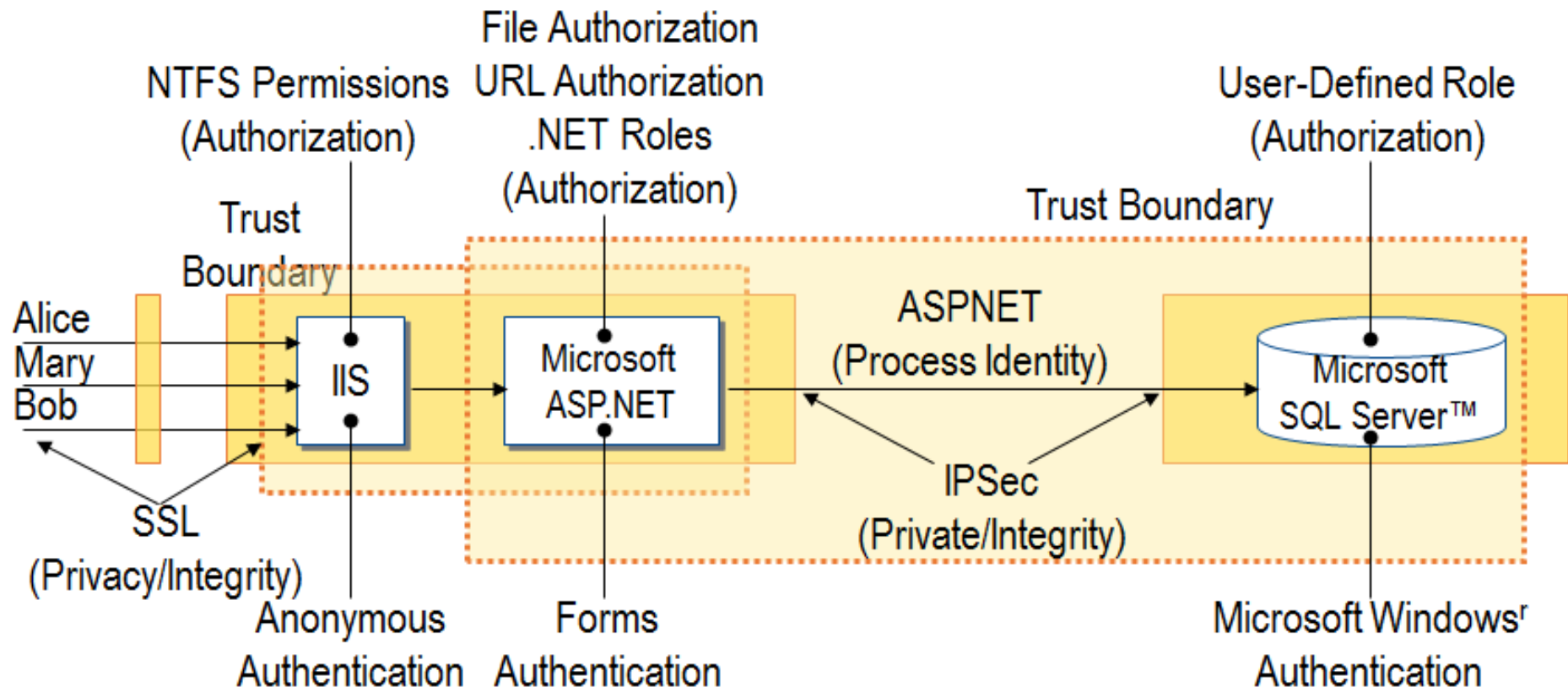
# Architecture and implementation technologies

-High-level diagram - describes the structure (components) of the system

-depending on the complexity of the application, more detailed diagrams of parts should be created

-eg diagrams of individual layers of a multi-layered application

-determination of implementation technologies to which aspects of protection are added, e.g.





## Step 3 – Application Decomposition

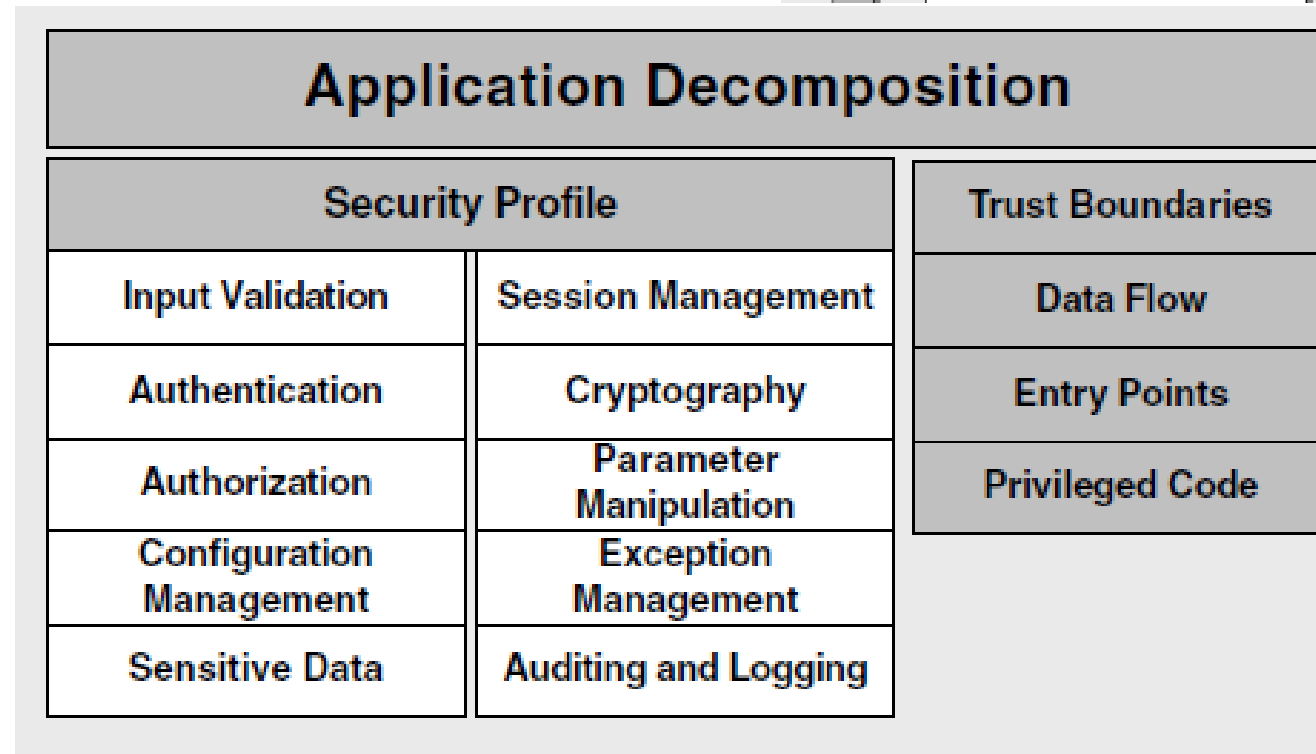
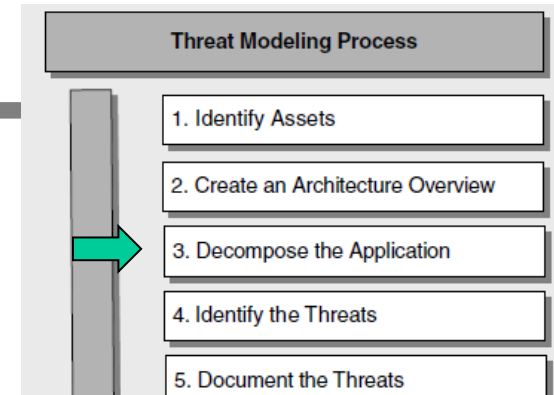
### -Creating a security profile

- Determination
  - trust boundaries
  - data flow
  - places of entry
  - privileged code

### -For every application

### -Decomposition techniques

- functional decomposition, activity diagram, data flow diagram, ...



# Confidence limits and data flows

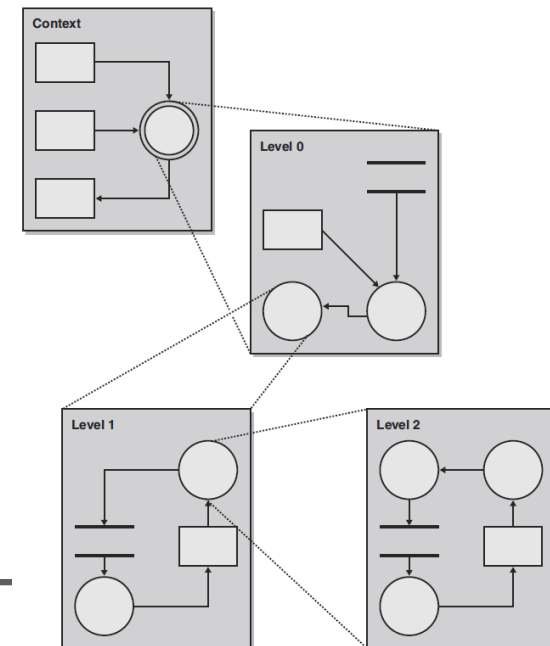
---

## -Determination of confidence limits

- analysis of the resource environment determined by the application design
- for each subsystem, an evaluation of whether the input stream or user input is confidential
  - if not - consider how to authenticate and authorize them
- assessment of whether the calling code is confidential
- checking server trust relationships (server trust relationships)

## -Determination of data flow (data flow)

- iterative decomposition
- by analyzing flows between subsystems, and in depth
  - Levels: 0-system, 1-main capabilities, 2-details



# Data flow diagram - notation

- A process, a multiple process
  - data processing, or action based on data
  - collection of subprocesses, can be decomposed
- Data storage
  - Any form of storage (file, BP, ...)
- Confidence limit
  - mark of privilege change (data rights levels)
- External entity, participant
  - everything that is outside the application, and interacting through the entry point
- Data flow
  - directed movement of data within the application



## **A Process**

Transforms or manipulates data.



## **Multiple Processes**

Transforms or manipulates data.



## **A Data Store**

A location that stores temporary or permanent data.



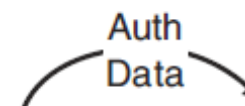
## **Boundary**

A machine, physical, address space or trust boundary.



## **Interactor**

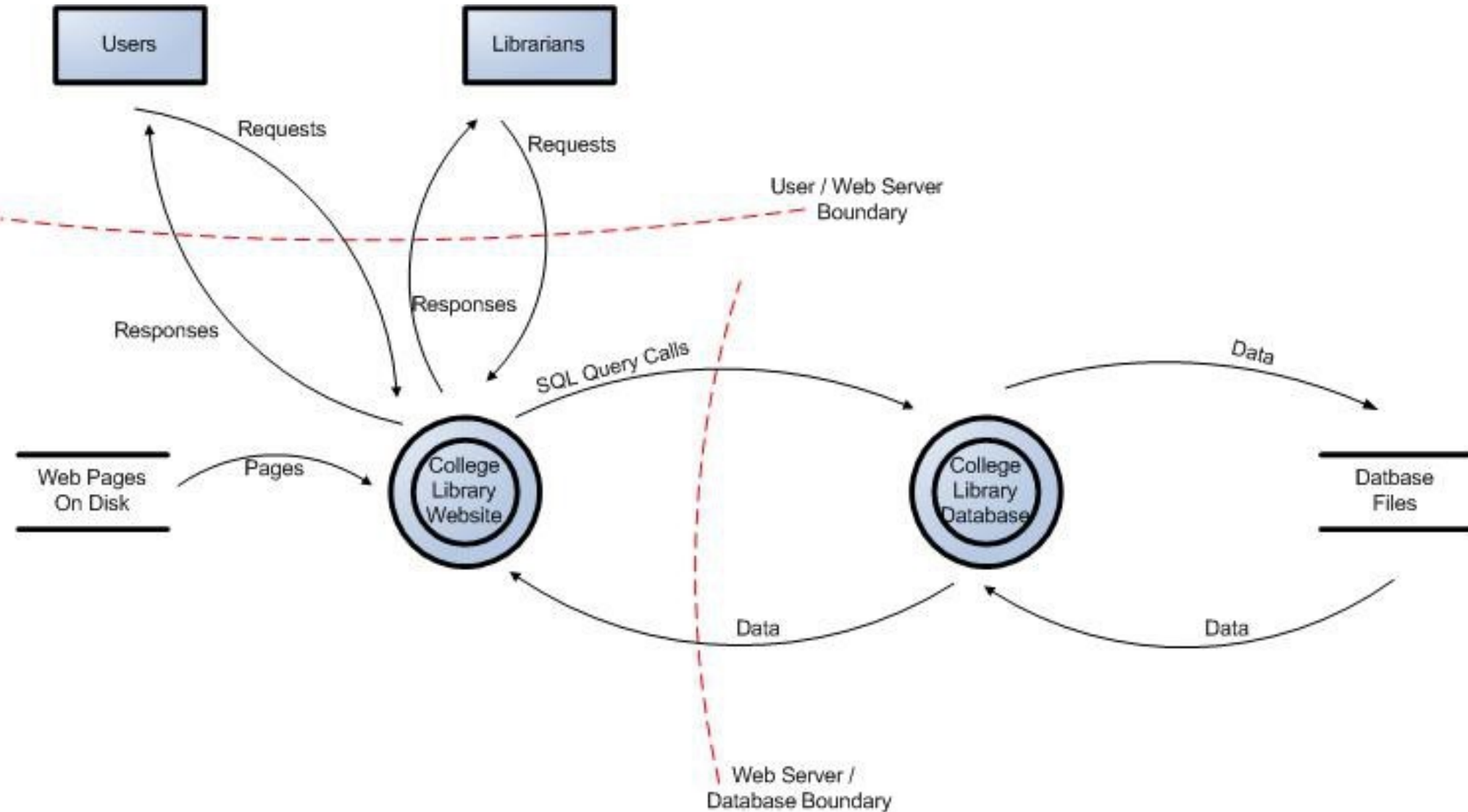
Input to the system.



## **Data Flow**

Depicts data flow from data stores, processes or interactors.

# Data flow diagram - example



## Other decomposition activities

---

### -Determining the entry point

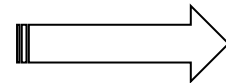
- parts of the user interface, eg web application pages
- data transfer connection points, e.g. web service interfaces, remoting components, physical ports and sockets

### -Specifying a privileged code

- that accesses certain types of secure resources or performs privileged operations
- ex. non/secure resources: DNS servers, *registry*, *event log*, ..., printers, web services, ...
- ex. non/safe operations: *unmanaged code calls*, reflection, serialization, ...

### -Documenting the security profile

- determining access to design and installation for input validation, authentication, authorization, configuration management, ...
- examples of questions to answer when creating a profile



Category	Considerations
Input validation	<p>Is all input data validated?</p> <p>Could an attacker inject commands or malicious data into the application?</p> <p>Is data validated as it is passed between separate trust boundaries (by the recipient entry point)?</p> <p>Can data in the database be trusted?</p>
Authentication	<p>Are credentials secured if they are passed over the network?</p> <p>Are strong account policies used?</p> <p>Are strong passwords enforced?</p> <p>Are you using certificates?</p> <p>Are password verifiers (using one-way hashes) used for user passwords?</p>
Authorization	<p>What gatekeepers are used at the entry points of the application?</p> <p>How is authorization enforced at the database?</p> <p>Is a defense in depth strategy used?</p> <p>Do you fail securely and only allow access upon successful confirmation of credentials?</p>
Configuration management	<p>What administration interfaces does the application support?</p> <p>How are they secured?</p> <p>How is remote administration secured?</p> <p>What configuration stores are used and how are they secured?</p>
Sensitive data	<p>What sensitive data is handled by the application?</p> <p>How is it secured over the network and in persistent stores?</p> <p>What type of encryption is used and how are encryption keys secured?</p>

Category	Considerations
Session management	<p>How are session cookies generated?</p> <p>How are they secured to prevent session hijacking?</p> <p>How is persistent session state secured?</p> <p>How is session state secured as it crosses the network?</p> <p>How does the application authenticate with the session store?</p> <p>Are credentials passed over the wire and are they maintained by the application? If so, how are they secured?</p>
Cryptography	<p>What algorithms and cryptographic techniques are used?</p> <p>How long are encryption keys and how are they secured?</p> <p>Does the application put its own encryption into action?</p> <p>How often are keys recycled?</p>
Parameter manipulation	<p>Does the application detect tampered parameters?</p> <p>Does it validate all parameters in form fields, view state, cookie data, and HTTP headers?</p>
Exception management	<p>How does the application handle error conditions?</p> <p>Are exceptions ever allowed to propagate back to the client?</p> <p>Are generic error messages that do not contain exploitable information used?</p>
Auditing and logging	<p>Does your application audit activity across all tiers on all servers?</p> <p>How are log files secured?</p>

# Step 4 - Threat Determination

- They are done by the development team and the testing team
  - architects, security guards, developers, testers and system administrators

## -Basic approaches

### -**OYSTERS** modeling practice defined by SDL

- acronym (*Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege*)

### -Categorized threat lists

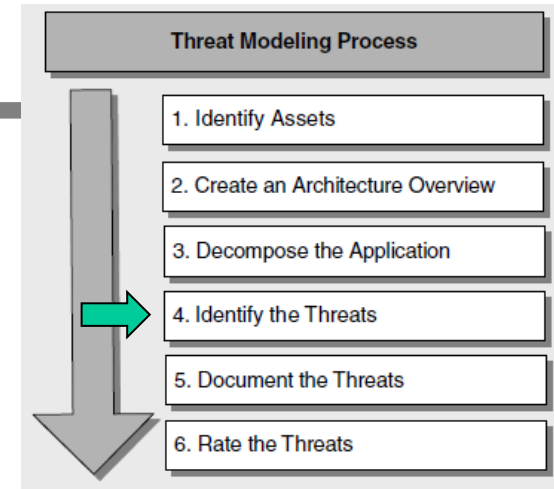
- list of commonly "suspicious" threats (*laundry list*)
- grouped by categories: network, server, application
- applying the list to your own architecture

## -Other useful techniques

### -Threat Trees (*threat trees*)

- describe what decisions an attacker must make when attacking a component

### -Attack Patterns (*attack patterns*)







# STRIDE - assessment by threat categories

---

- **WITH**poofing – deceiving, faking
  - assuming someone else's identity in order to access resources in the network
  - eg illegal retrieval of other people's data during authentication
- **T**ampering [with Data] – malicious modification of data
  - unauthorized modification, for example, in the database or during network transmission
- **R**epudiation – non-recognition, denial
  - the ability of the user to deny the action without being able to prove it to him
  - eg "I did not delete", "I did not order", ...
- **AND**nformation disclosure - disclosure of information
  - unwanted exposure of private data
  - for example, the user sees the content of someone else's file to which he has no right
- **D**enial of service - denial of service
  - prevents the normal operation of the system, relatively simply and anonymously
  - for example *flooding, amplification, protocol vulnerability, malformed packets*
- **E**elevation of privilege - elevation of authority
  - a user with limited authority assumes the identity of a user with higher authority

# STRIDE - procedure

---

- The system is broken down into relevant components
  - the threat sensitivity of each component is assessed
  - threats are reduced (mitigation) by appropriate security features
  - it is repeated (recursively) until a satisfactory result
- 
- 
- The impact of threats on individual parts of the system
  - ... by analyzing data flow diagrams

Element	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privileges
Data Flows		X		X	X	
Data Stores		X	?	X	X	
Processes	X	X	X	X	X	X
Interactors	X		X			

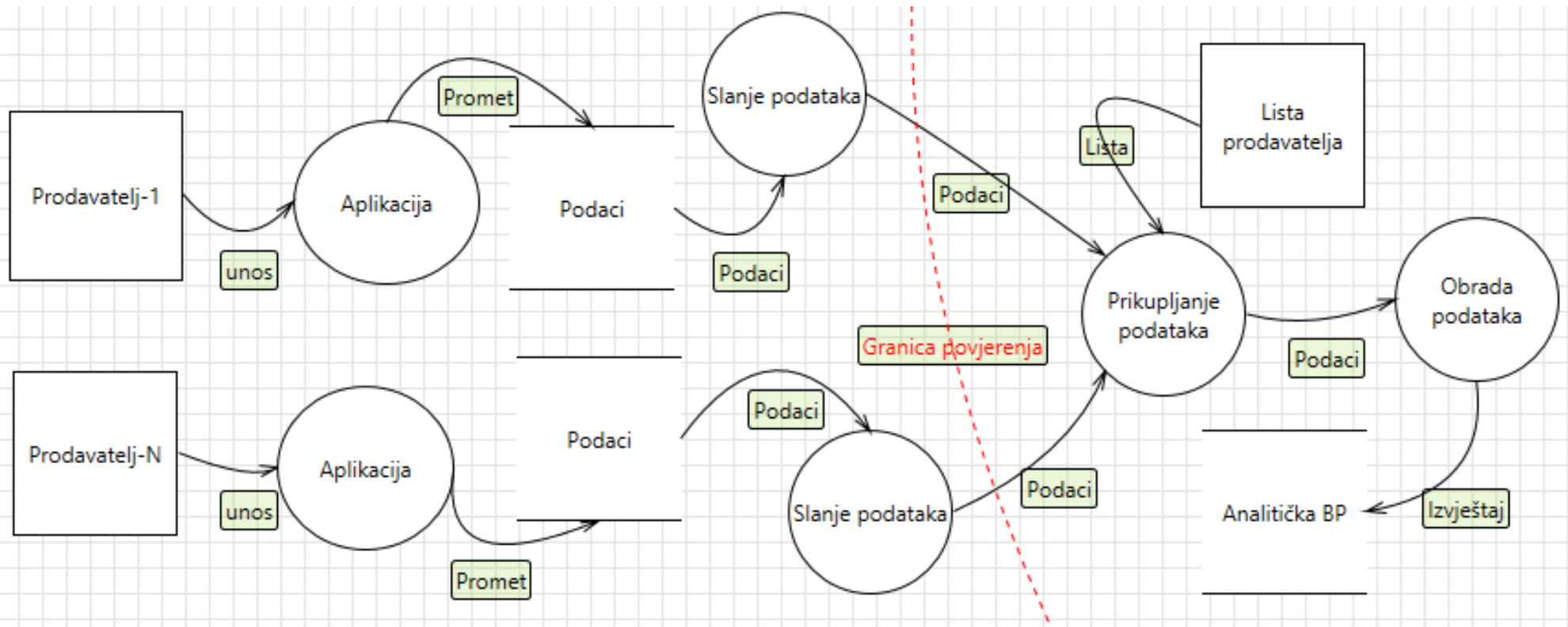
Spoofing	Authentication	<ul style="list-style-type: none"> <li>To authenticate principals: <ul style="list-style-type: none"> <li>Basic &amp; Digest authentication</li> <li>LiveID authentication</li> <li>Cookie authentication</li> <li>Windows authentication (NTLM)</li> <li>Kerberos authentication</li> <li>PKI systems such as SSL/TLS and certificates</li> <li>IPSec</li> <li>Digitally signed packets To authenticate code or data: <ul style="list-style-type: none"> <li>Digital signatures</li> <li>Message authentication codes</li> <li>Hashes</li> </ul> </li> </ul> </li> </ul>
Tampering	Integrity	<ul style="list-style-type: none"> <li>Windows Mandatory Integrity Controls</li> <li>ACLs</li> <li>Digital signatures</li> <li>Message Authentication Codes</li> </ul>
Repudiation	Non Repudiation	<ul style="list-style-type: none"> <li>Strong Authentication</li> <li>Secure logging and auditing</li> <li>Digital Signatures</li> <li>Secure time stamps</li> <li>Trusted third parties</li> </ul>
Information Disclosure	Confidentiality	<ul style="list-style-type: none"> <li>Encryption</li> <li>ACLS</li> </ul>
Denial of Service	Availability	<ul style="list-style-type: none"> <li>ACLs</li> <li>Filtering</li> <li>Quotas</li> <li>Authorization</li> <li>High availability designs</li> </ul>
Elevation of Privilege	Authorization	<ul style="list-style-type: none"> <li>ACLs</li> <li>Group or role membership</li> <li>Privilege ownership</li> <li>Permissions</li> <li>Input validation</li> </ul>

## STRIDE - example

---

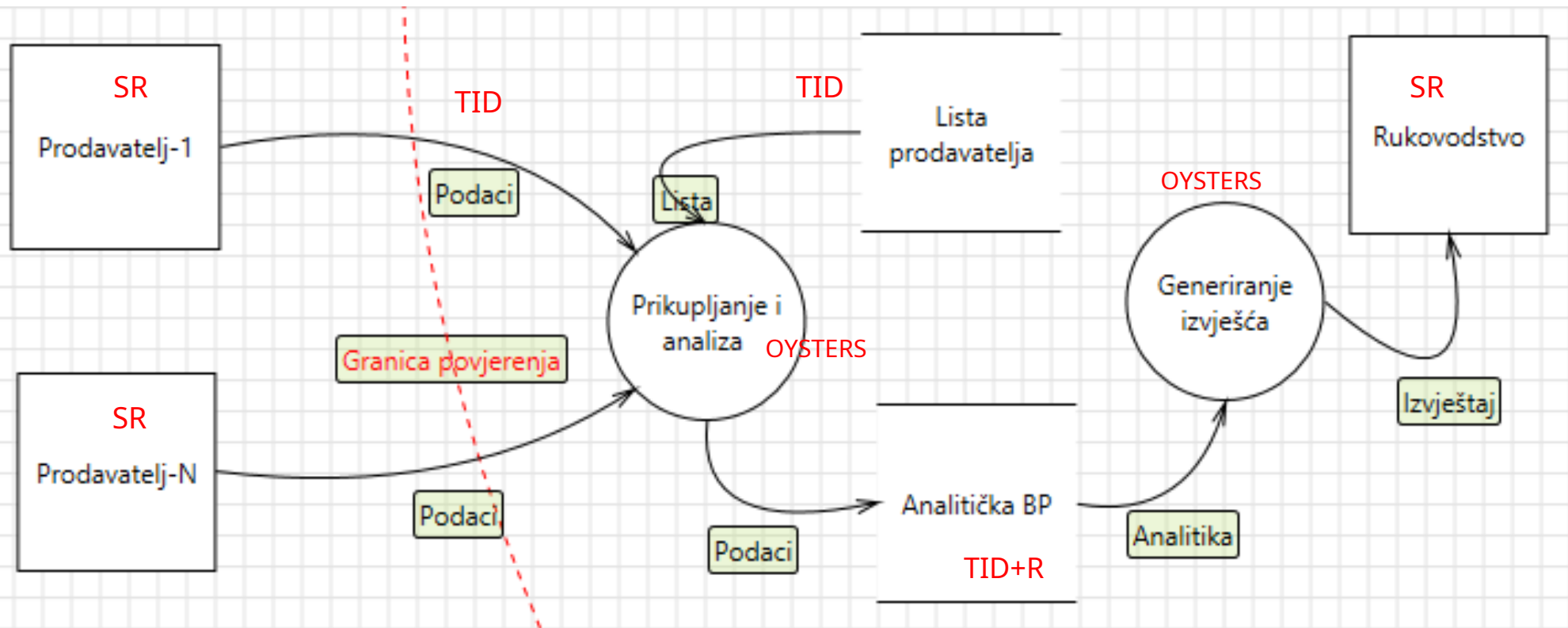
- Example: Uncover Security Design Flaws Using The STRIDE Approach
  - Sellers collect sales data in local records
  - Sales files should be collected on the server
  - And generate weekly reports
  - For previously registered sellers
- Security requirements
  - Protect data during transmission and storage
  - Authenticate and authorize sellers
  - Application resistant to attacks (intakes, injections, overflows)
  - ...
- the user does not need to pronounce them all - they need to be invented in view of the problem

# Initial diagram – client/server



- Data sinks - someone needs to read them, by process
- Chained processes - indicate dependence, separate
- Redundancy – generalize and normalize behavior (functionality)
- Wrong data sources - check, change

# Improved diagram – server side analysis



- Client kicked out
- Added *Generation*, consequently *iManagement*
- The list became a repository
- Integrated processing

-OYSTERS?

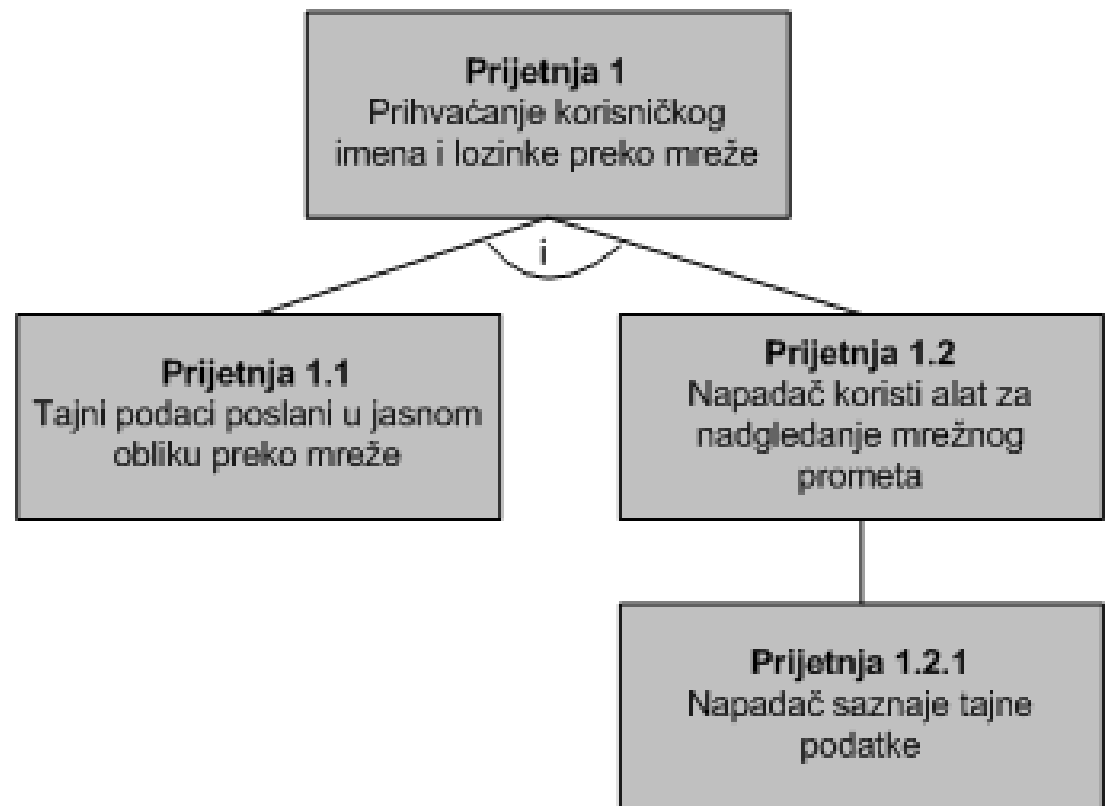
# Threat trees

---

- For each component obtained by decomposition
  - possible threats are determined
  - the way in which threats are reflected on the system is determined

## -Example

- the root is a threat
- children represent the steps an attacker must take to execute a threat



# Threat Trees (continued)

---

## -Alternative view

1.0 Threat 1 :

Accepting username and password over the network

1.1 Confidential data sent in clear form over the network**AND**

1.2 The attacker uses a tool to monitor network traffic

1.2.1 The attacker learns secret information

## -Using STRIDE over threat trees is easy

-for each part of the system, it is checked whether it is subject to one of the STRIDE categories

-for example, can an attacker deny the operation of a process, view data, etc.

-What threats does the above example illustrate?



# Attack patterns

---

## -General representation of common attacks

-defines the aim, conditions, technique and result of the attack

-The emphasis is on attack technique (with STRIDE on the attacker's goals)

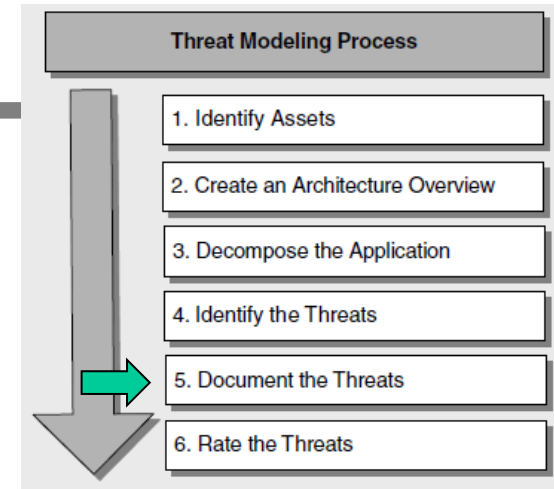
## -Example form:

Pattern	Code injection attacks
Attack goals	Command or code execution
Required conditions	Weak input validation Code from the attacker has sufficient privileges on the server.
Attack technique	1. Identify program on target system with an input validation vulnerability. 2. Create code to inject and run using the security context of the target application. 3. Construct input value to insert code into the address space of the target application and force a stack corruption that causes application execution to jump to the injected code.
Attack results	Code from the attacker runs and performs malicious action.

## Step 5 - Documenting the threats

### -Threat log template

- Be sure to fill in the description and goal
- the risk is left for the next step
- other attributes may be optional



### -Examples

Threat Description	Attacker obtains authentication credentials by monitoring the network
Threat target	Web application user authentication process
Risk	
Attack techniques	Use of network monitoring software
Countermeasures	Use SSL to provide encrypted channel
Threat Description	Injection of SQL commands
Threat target	Data access component
Risk	
Attack techniques	Attacker appends SQL commands to user name, which is used to form a SQL query
Countermeasures	Use a regular expression to validate the user name, and use a stored procedure that uses parameters to access the database.

# Step 6 - Threat Ranking

-Ranking - determination of importance (rate the threats)

-techniques are often used to determine risk

-**risk**=probability of event \* potential damage

-**probability**eg in the range 1-10

-**too bad**eg in the range 1-10

-risk in the range 1-100

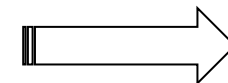
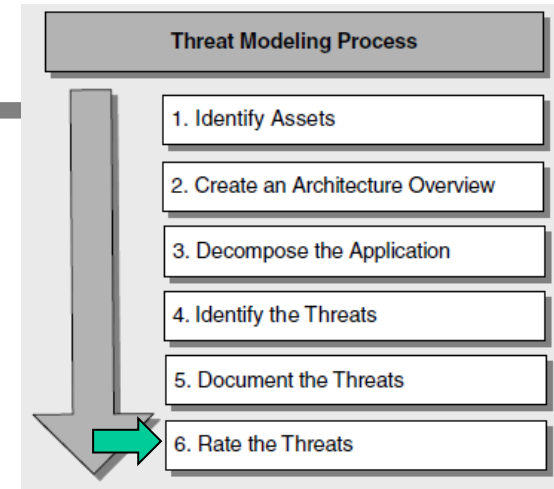
-distribution into three groups (high, medium, low) representing priorities

-An issue:

-team members cannot agree on values

-Solution:

-DREAD model, risk ranking for a given threat



# DREAD (risk assessment model)

---

## -DREAD– classification of computer threats

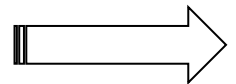
- **D***harm potential*–possible damage, amount of damage if the attack is successful
- **R***eproducibility*–reproducibility, how easy it is to repeat the attack
- **E***xploitability*–exploitability, effort and knowledge required for a successful attack
- **A***ND**affected users*–affected users, possibly by a successful attack, percentage
- **D***iscoverability*–detectability, difficult to measure

## -Assessment of each threat according to the specified parameters

- individual value from 1 to 10 (least bad - worst)
- total risk - average of 5 individual DREAD values

## -Better – (simple)grading scheme

- Low, medium, high – mapped to interval 1 to 3



# Example of a simple grading scheme

Rating		High (3)	Medium (2)	Low (1)
D	Damage potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.	Leaking sensitive information	Leaking trivial information
R	Reproducibility	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced, but only with a timing window and a particular race situation.	The attack is very difficult to reproduce, even with knowledge of the security hole.
E	Exploitability	A novice programmer could make the attack in a short time.	A skilled programmer could make the attack, then repeat the steps.	The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
A	Affected users	All users, default configuration, key customers	Some users, non-default configuration	Very small percentage of users, obscure feature; affects anonymous users
D	Discoverability	Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.	The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use.	The bug is obscure, and it is unlikely that users will work out damage potential.

## Example of a simple grading scheme (continued)

---

- The values (1-3) for the given threat are added up
  - the score is in the range 5-15
  - risk is assigned, eg 5-7 low, 8-11 medium, 12-15 high
- For example for two documented threats from the beginning of the story

Threat	D	R	E	A	D	Total	Rating
Attacker obtains authentication credentials by monitoring the network.	3	3	2	2	2	12	High
SQL commands injected into application.	3	3	3	3	2	14	High

- ... threat documentation templates are updated (step 5)

# Threat resolution (after modeling)

---

- Fix (reduction, risk reduction)

  - reduce the consequence

- Do nothing (accept the risk)

  - bad, if the problem is real, it will happen at some point and will have to be fixed

- Notify the user and leave the decision on use to him (transfer)

  - problematic

  - many users do not know what the right decision is, and the notifications are incomprehensible

  - use only if there is a great need to use the (risky) service

- Removal of the risky property (avoidance)

  - when the problem cannot be corrected immediately (e.g. there is no time, etc.),

  - to be corrected in the next version

## Threats and countermeasures - another time

---

Threats	Countermeasures (security techniques)
Deception	Appropriate authentication Protection of private data Private data must not be stored in a clear form
Modification of data	Appropriate authorization Use of compression functions Use of digital signatures
Denial	Use of digital signatures
Disclosure of information	Appropriate authorization Private data must not be stored in a clear form Provide a communication channel
Denial of Service	Validate and filter input data
Increase in authority	Grant only the necessary powers

---

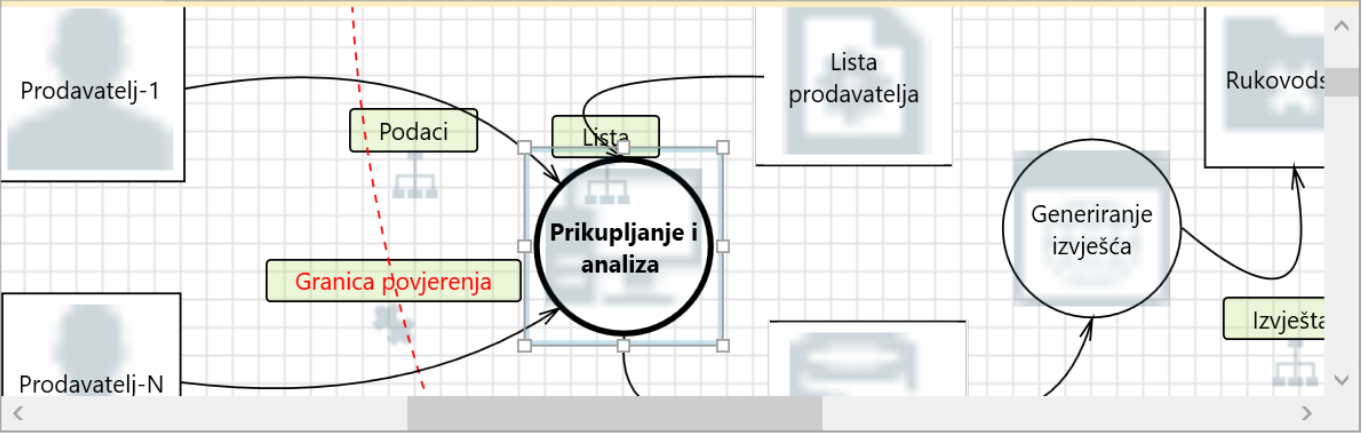


# Example, Microsoft Threat Modeling Tool

Firma02 - Microsoft Threat Modeling Tool

File Edit View Settings Diagram Reports Help DiagramReader

Analitika



Threat List

ID	Title	Category	Description	Short Descrip
9	Persistent Cross Site Scripting	Tampering	The web server 'Prikupljanje i an	Tampering is th
10	Cross Site Scripting	Tampering	The web server 'Prikupljanje i anali	Tampering is the

Export Csv Clear Filters 4 Threats Displayed, 31 Total

Threat Properties

ID: 11 Diagram: Analitika Status: Not Started Last Modified: 27. 11. 14. 15:46:46

Title: Spoofing of Source Data Store Lista prodavatelja

Category: Spoofing

Threat Properties Notes - no entries

Element Properties

Web Server

Name Prikupljanje i analiza

Out Of Scope ☐

Reason For Out Of Scope

Configurable Attributes

Code Type Managed

Sanitizes Input Not Selected

Sanitizes Output Not Selected

As Generic Process

Running As Not Selected

Isolation Level Not Selected

Accepts Input From Not Selected

Implements or Uses an Authentication Mechanism No

Implements or Uses an Authorization Mechanism No

Implements or Uses a Communication Protocol No

Add New Custom Attribute

---

## **Reducing the attack surface**

Attack Surface Reduction

---

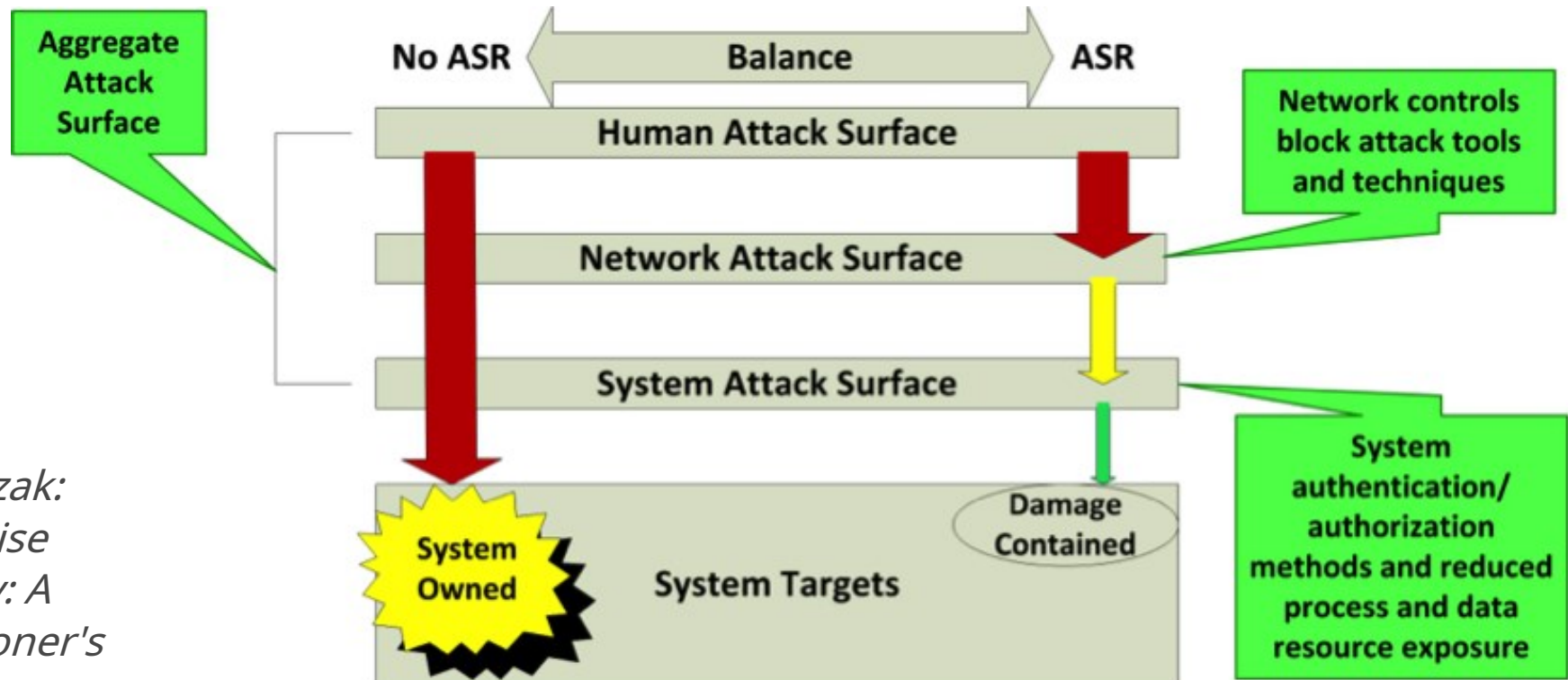
# attack surface

---

- **Attack surface** - a collection of program product entry points
  - User interfaces, web services, direct access to BP, network channels, API, ...
  - resource communication channels = attack vectors
  - **attackability measure**
- Larger attack surface = more protection work = more potential damage
- Surface area determines attack risk – a measure of potential access and impact
  - Experience shows that certain vectors are more risky
  - For example privileged (*root*) services, files with full access (*rw xrwxrwx*), scripts (JScript, VBScript) and active controls (ActiveX)

# Combined attack surface model

- access controls reduce
  - possibility to reach the system
  - the number of elements that are visible or usable



*Tom Olzak:  
Enterprise  
Security: A  
practitioner's  
guide*

# Attack surface reduction

---

- Main objectives

- Reducing the amount of code that is executed "by default"
- Reducing the amount of code that can be accessed by untrusted users, "on sight"
- Closing access points (access points, entry points) - doors that are easily opened/used
- Limiting the damage in case the access point is exploited

- The ultimate goal – repelling future attacks

# A common software security metric

---

## -Code level - bug count

- It does not count bugs that have not (yet) been found
- All bugs are of equal severity, although some are easier to exploit
- Some bugs can cause more damage than others

## -Product/system level

- Counting how many times the system version is mentioned in CERT, MITER CVE, ... bulletins
  - Computer emergency response teams (CERT), <http://www.cert.hr/>
  - Common Vulnerabilities and Exposures (CVE) dictionary, <https://cve.mitre.org/>
- Ignores specific configurations
  - Installed patches
  - Defaults on or off
  - Work in *admin* fashion

### -Measuring avenues of attack

- "more likely to be attacked" features

### -Measuring relative safety

- Delta measurement – differences between versions of the same product (eg v1 vs v2)
- Unusable for comparing different applications

### -Procedure

- Baseline + weekly measurements
- Determining the minimum area at the beginning
- If the area increases – determine how to reduce it

# Attack surface and access points

---

-Example of comparison of measurements of different versions

Baseline	Baseline + 1 month	Comment
3 x TCP ports	2 x TCP ports	Good; one fewer port to worry about.
1 x UDP port	2 x UDP port	Which functionality opened the new UDP port? Why is it open by default? Is it authenticated? Is it restricted to a subnet?
2 x Services (both SYSTEM)	3 x Services (2 x SYSTEM, 1 x LocalService)	Why is another service running by default? Why are any running as SYSTEM?
3 x ActiveX controls	4 x ActiveX controls	Why is the new control installed? Is it safe for scripting?
No additional user accounts	1 x application account	Turns out this is a member of the administrators group too! Why? What's the password?

*Fending Off Attacks by Reducing an Application's Attack Surface*  
Jason Taylor CTO, Security Innovationan SDL Pro Network member company



# ASR process

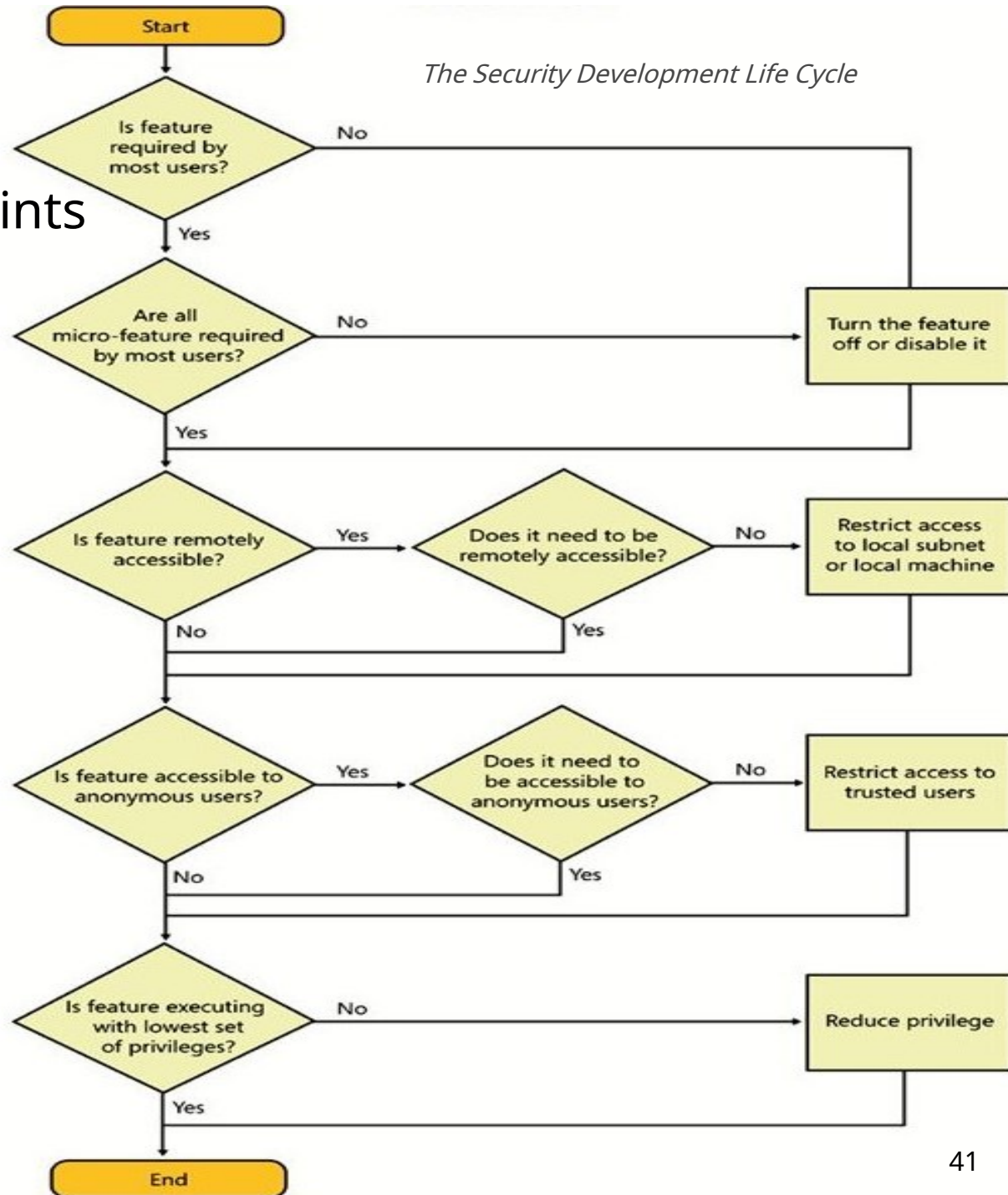
- Establishment of access points
  - network, files, ...

## -Ranking points

- according to the user
  - authenticated – anonymous
  - admin-user*
  - online -*local*

## -Adjustment

*The Security Development Life Cycle*



# It's not *Just* About Turning Stuff Off!

---

## Higher Attack Surface

Executing by default

Open socket

Anonymous access

Constantly on

Admin access

Internet access

SYSTEM

Uniform defaults

Large code

Weak ACLs

## Lower Attack Surface

Off by default

Closed socket

Authenticated access

Intermittently on

User access

Local subnet access

Not SYSTEM!

User-chosen settings

Small code

Strong ACLs

# Best practices

---

- Reduction of the code being executed *by default*
  - Turn off an option that is not used by at least 80% of users
  - A stopped service cannot be attacked
    - dynamic web content should be optional - it affects only those who initiate it
  - The solution is not just to switch off
    - restriction of access to running code
- Reducing access by untrusted users
  - Restricting access to a local network or IP address range
  - Authentication

## Best practices (continued)

---

### -Privilege reduction to limit potential damage

- Revoking privileges that are not absolutely necessary
- Running code in the security box (sandbox running code)
  - by limiting the permissions to which the code is entitled,
  - Java.Class.SecurityManager*, or *.NET System.Security.SecurityManager*
- If necessary, pick up permits - temporarily, as short as possible
- Paying attention to confidence limits (threat modeling procedure)
  - particularly *road*sof anonymous threats (anonymous threat paths) → authorization where necessary
- Do not run services as SYSTEM (daemons as *root*) or with administrator rights until other options are exhausted!

### -Example:

- Backup Operator account*-reads all files regardless of their ACL
- SYSTEM works the same but also *restore, debug, "act as part of OS"* that's it *admin*

# Best practices (rest)

---

- Defining the attack surface during design/engineering
  - sketch the attack surface and establish
    - protocols
    - endpoints that need authentication and authorization
    - off-by-default options – autostart (eg Windows \ Services, starter.exe)
    - reusable components (ActiveX, COM, .NET assemblies, etc.)
    - process identities
    - installed user accounts
- Other procedures
  - Threat modeling
  - Attack surface overview – analyzer: base + differences
  - Design review – looking for threats and mitigation opportunities
  - Code Review - Defensive Programming and Secure Coding

# Example: Attack Surface Analyzer

## Attack Surface Report: Table Of Contents

**Welcome to Attack Surface Analyzer**

Attack Surface Analyzer scans the system to identify potential security issues. To isolate the results to those specific to your product, you should scan the system at least twice:

- The first scan, called the *baseline*, should be run on a *clean* system without your product installed, but with external dependencies such as SQL Server already installed.
- The following scan, called the *product* scan, should be run after installing your product to the system.

Each scan will generate a .CAB file that can be analyzed to generate a report identifying potential issues. Pairs of scans, made up of a .CAB file generated before a product installation and a .CAB file generated after, can be analyzed to determine issues present on the system and changes to the system's attack surface resulting from the installation. Generating new .CAB pairs while enabling and disabling different product features may allow you to better isolate the source of identified issues.

**Please select an action:**

☐ Run new scan

☒ Generate standard attack surface report

**Select options:**

Baseline Cab: C:\Users\kreso\Attack Surface Analyzer\GOLIJAT\_1.0.0\_2014-11-27\_15-

Product Cab: C:\Users\kreso\Attack Surface Analyzer\GOLIJAT\_1.0.0\_2015-12-01\_22-

Report Filename: C:\Users\kreso\Attack Surface Analyzer\GOLIJAT\_1.0.0\_2015-12-01\_22-31-

**Microsoft**

- [System Information](#)
  - [Running Processes](#)
  - [Executable Memory Pages](#)
  - [Windows](#)
  - [Impersonation Tokens](#)
  - [Kernel Objects](#)
  - [Window Stations](#)
  - [Desktops](#)
  - [Modules](#)
- [Service Information](#)
  - [Services](#)
  - [Drivers](#)
- [ActiveX, DCOM, COM, File Extensions](#)
  - [COM Controls](#)
  - [ActiveX Controls](#)
  - [DCOM Controls](#)
  - [File Registrations](#)
- [Internet Explorer](#)
  - [Pluggable Protocol Handlers](#)
  - [IE Silent Elevations](#)
  - [IE Preapproved Controls](#)
  - [Browser Helper Objects](#)
- [Network Information](#)
  - [Network Ports](#)
  - [Named Pipes](#)
  - [RPC Endpoints](#)
  - [Network Shares](#)
- [Firewall](#)
  - [Firewall Rules](#)
- [System Environment, Users, Groups](#)
  - [%PATH% Entries](#)
  - [Groups](#)

# References

---

## -Procedures

- [A systematic review of security requirements engineering, Mellado et.a., 2010](#)
- [Threat Modeling with STRIDE](#)

## -Tools

- [Attack Surface Analyzer](#)
- [Microsoft Threat Modeling Tool](#)
- [Top 10 Threat Modeling Tools in 2021](#)