UNIVERSITÄT BONN

Secure Software Engineering

Winterterm 2024/25

Architectural Risk Analysis & Distrustful Decomposition

Dr. Christian Tiefenau

Outline

- Vulnerability of the Day

Cross-site scripting

- Risk analysis

Threat Modeling

Determine, Rate, Countermeasures

Data Flow Diagrams and Trust Boundaries

- Distrustful Decomposition

Principle of Least Privilege

Defense in Depth

Permissions

IPC

Behavioural Security Group

UNIVERSITÄT BONN

Vulnerability of the Day

Cross-site Scripting (XSS)

XSS - Setting

- Webpage includes data in dynamic content and is sent to a web user

```
<?php
//Get user input
if($_GET[ 'name' ] != NULL ) {
    //Create Dynamic Content
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}
?>
```

- Goal: https://www.example.com/?name=JohnDoe

```
>Server Output: “Hello JohnDoe”
```

XSS - Example

- Inject code such that this code is executed by the clients browser as valid HTML / JavaScript code

JavaScript

```
https://www.example.com/?name=<script>alert('Pwnd');</script>
>What's your name?: <script>alert('Pwnd');</script>
```

HTML + JavaScript Create a Login form and send data to another server

```
>What's your name?: <form>Login:<input type="text" name="firstname"
value="Mickey">  <br> PW  <input type="text" name="lastname"
value="Mouse"><br> <input type="submit" value="Submit"></form>
```

- Injected Code stored in the parameter ?name=<injectedCode>
  Link with this parameter can be distributed

Different kinds of XSS

- Reflective:
  - Injected input is sent back to the client, e.g., using GET parameters in a URL the victim got from an attacker
- Persistent/Stored:
  - Injected input is sent stored on server side and delivered in future request, e.g. guest books or user profiles
- DOM-based / local:
  - Exploit is interpreted/executed on the client machine
  - No server interaction needed
  - Some other attack vector needed, e.g. malicious plugin with access to browser content

Mitigations

- Escaping “bad” characters via built-in function (no self-made regex)

```php
<?php
//Get user input
if($_GET[ 'name' ] != NULL ) {
    //Escape Special Characters
    $name = htmlspecialchars( $_GET[ 'name' ] );
    //Create Dynamic Content
    echo "<pre>Hello ${name}</pre>";
}
?>
```

Current Example - CVE-2025-55200

BigBlueButton: Update fixes web conferencing system vulnerabilities

Remote troublemakers could sabotage online lessons via three loopholes with a "high" rating in the BBB software. An update locks them out.

(Image: Photon photo/Shutterstock.com)

Oct 13, 2025 at 8:33 pm CEST 2 min. read | Security

By Olivia von Westernhagen

The developers of the open-source web conferencing system BigBlueButton (BBB) for Windows and Linux servers have eliminated several possible attacks with an **update to version 3.0.13**.

Under certain circumstances, authenticated attackers could have remotely abused three vulnerabilities with a high degree of severity ("High") to sabotage the chat functions of all users during video conferences or execute malicious scripts via cross-site scripting (XSS). In addition, it was also possible to crash the current meeting or, in the worst case, all online conferences currently taking place on the server in question (denial of service).

CVE xxxx-yyyyy

- **C**ommon **V**ulnerabilities and **E**xposures (~ 221k)
  - Managed and hosted by MITRE
  - https://cve.mitre.org/
  - We already referred to it every lecture
  - E.g., CVE-2011-1153

CVE-ID	
CVE-2011-1153	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Multiple format string vulnerabilities in phar_object.c in the phar extension in PHP 5.3.5 and earlier allow context-dependent attackers to obtain sensitive information from process memory, cause a denial of service (memory corruption), or possibly execute arbitrary code via format string specifiers in an argument to a class method, leading to an incorrect zend_throw_exception_ex call.	
References	

Listed as CWE-79

Behavioural Security Group

UNIVERSITÄT BONN

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Weakness ID: 79
Abstraction: Base
Structure: Simple

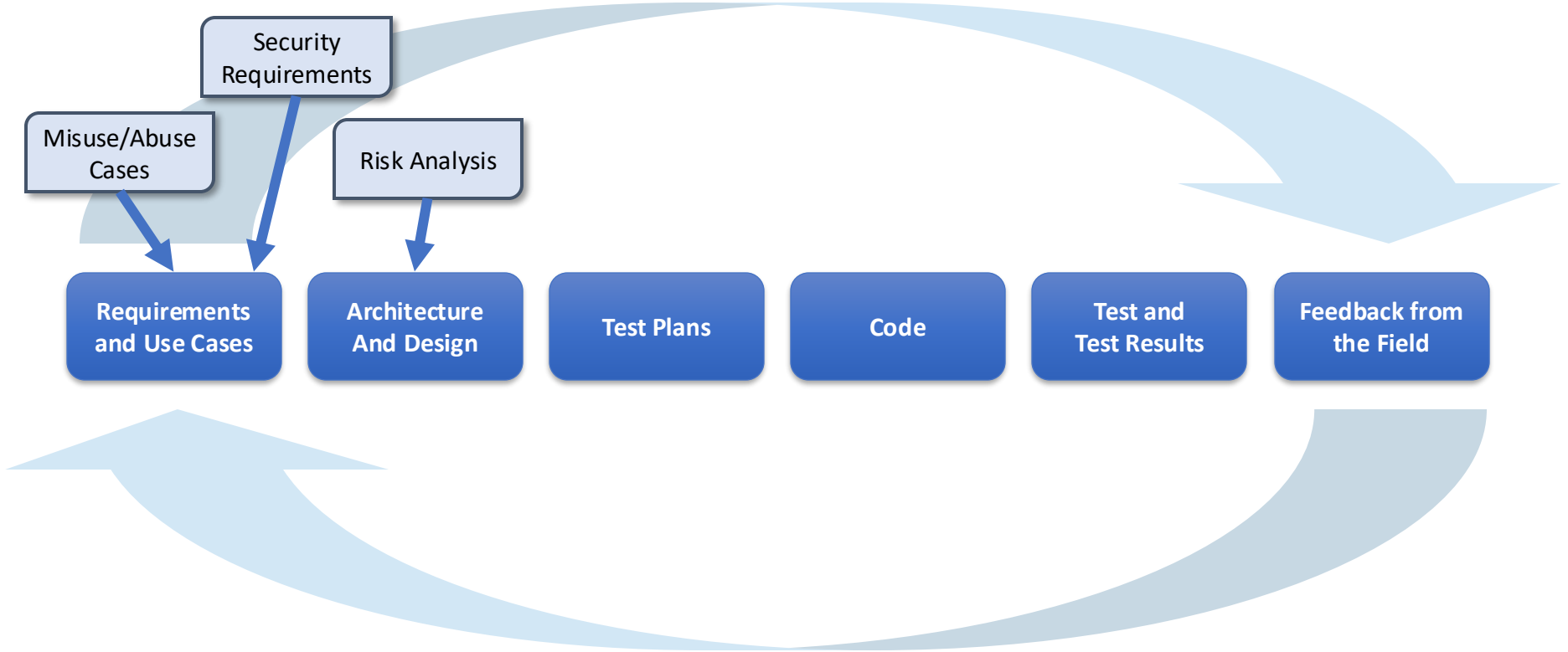
Status: Stable

https://cwe.mitre.org/data/definitions/79.html

Behavioural Security Group

UNIVERSITÄT BONN

Architectural Risk Analysis & Distrustful Decomposition


Security Requirements
Misuse/Abuse Cases
Risk Analysis
Requirements and Use Cases
Architecture And Design
Test Plans
Code
Test and Test Results
Feedback from the Field

Different View on your System

- **Previously: Abuse & Misuse Cases:**
  - Start with the ***functionality***
    - p(exploit), p(vulnerability)
  - Start with ***what to protect***
    - Assets
    - Domain, domain, domain
    - Goals -> High-level Risks -> Indicators -> Tests
- **Today:** start with ***what to protect against***
  - ***Threats, attacks***

Back in 2002…

- Windows XP was cutting edge technology (and ran on a tablet!)
- But also…

2002 – Bill Gates on Trustworthy Computing

UNIVERSITÄT BONN

- … Bill Gates informed about the importance of Trustworthy Computing

From: Bill Gates
Sent: Tuesday, January 15, 2002 5:22 PM
To: Microsoft and Subsidiaries: All FTE
Subject: **Trustworthy computing**

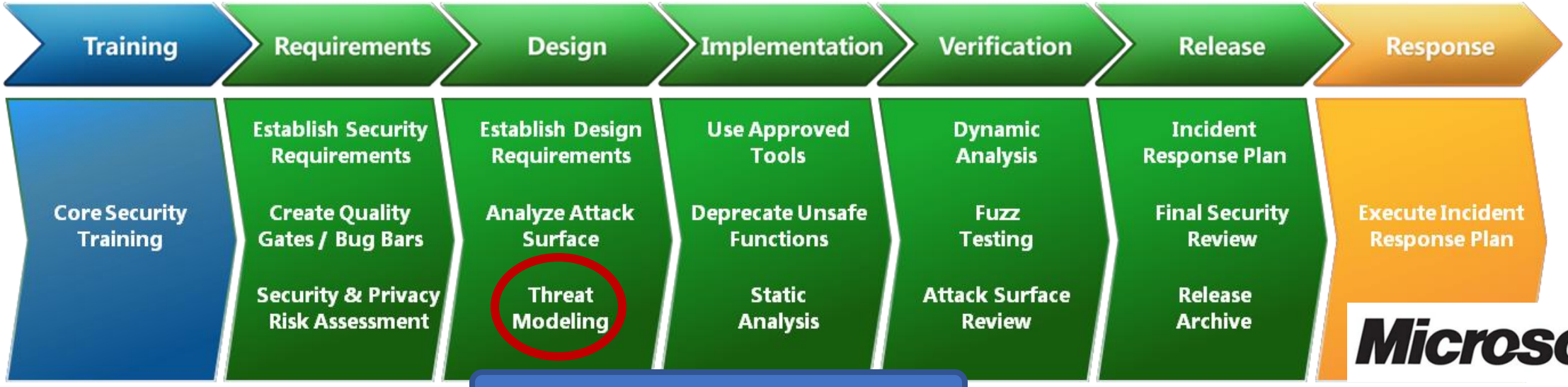
Microsoft

Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that ensuring .NET is a platform for Trustworthy Computing is more important than any other part of our work. If we don't do this, people simply won't be willing — or able — to take advantage of all the other great work we do. ...

https://www.wired.com/2002/01/bill-gates-trustworthy-computing/

Trustworthy Computing Initiative

- The Microsoft initiative really changed the way in which people engineered software
- First big company known to adopt dedicated secure software engineering practices
- Over the years, Microsoft also researched and developed many appropriate methodologies



- We'll see some of them in this lecture

- Create Data Flow Diagram

Threat Modeling Tool

- Identify Threats for each element
- Discuss, Refine & Rate Threats

STRIDE

- Provide Mitigation for Threats
- Create Threat Model Report

Influences on todays software

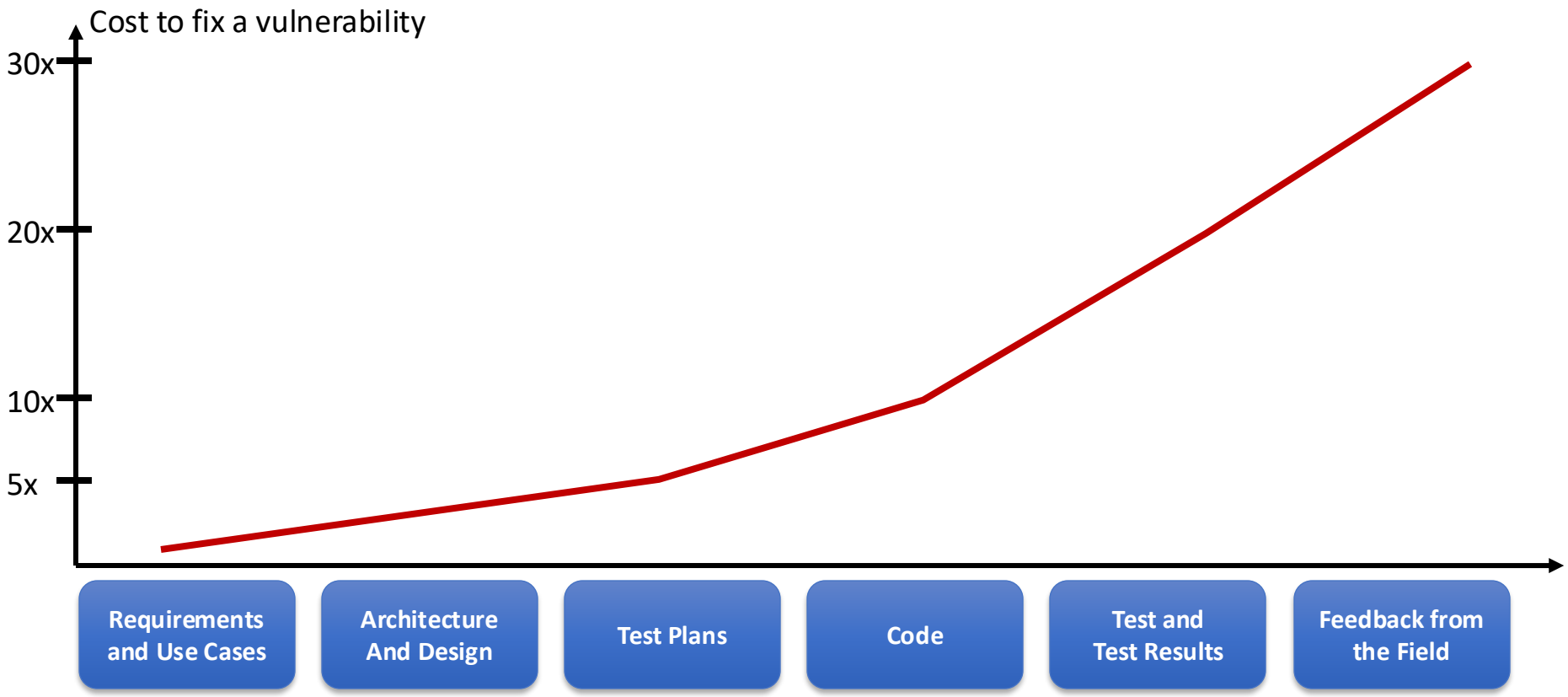
- Also influenced current standards and certifications:

- IT-Sicherheitsgesetz 2.0 (https://www.bsi.bund.de/EN/Das-BSI/Auftrag/Gesetze-und-Verordnungen/IT-SiG/2-0/it_sig-2-0_node.html)
  - "KRITIS" companies have to report cyber attacks
  - Develop and define security processes and standards
    - Will be certified and controlled by BSI
  - Also new EU directive ("NIS-Richtlinie")

- BSI Grundschutz (https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/it-grundschutz_node.html)
  - BSI already provides guidelines for basic security
  - Contains also guidelines for threat modeling and risk analysis

- Certifications, e.g., ISO 27000 family for ISMS (https://www.iso.org/standard/iso-iec-27000-family)
  - Focuses on holistic approach
  - Threat Modeling is part of this process

Secure Development Lifecycle – Phases and Costs

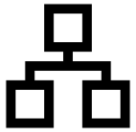


- Detecting vulnerabilities early on helps to reduce costs.

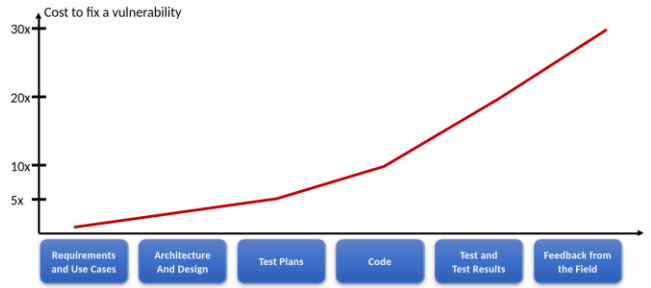
Architectural Risk Analysis

- Structured, repeatable approach to identify potential security flaws in software or systems at design time

- Teamwork (!)
- Modeling aspects of the system for the purpose of finding threats
  - “Threat Modeling”



- Discuss security risk once most of the architecture is settled
- Motivation: a few good early decisions go a long way, e.g.,
  - Incorporating encryption at the right places
  - Authentication & access control concerns
  - Choice of technologies used



Architectural Risk Analysis

- Answer questions like
  - What are potential threats?
  - What are my assets? (also)
  - Who would / can attack me?

- Emphasis of design flaws over code-level vulnerabilities
- Must-haves vs. Nice-to-haves at the design level
- Starting point for documentation and further security actions during the development

Threat Modeling – General Approach

- Determine threats
  - Identify entry/exit points
  - Identify trust boundaries
  - Identify threats
- Rate threats
  - Identify most probable threats
  - Identify most critical threats
- Determine countermeasures
  - Identify solutions to mitigate threats
  - Document non-mitigated threats

Data Flow Diagram Primitives

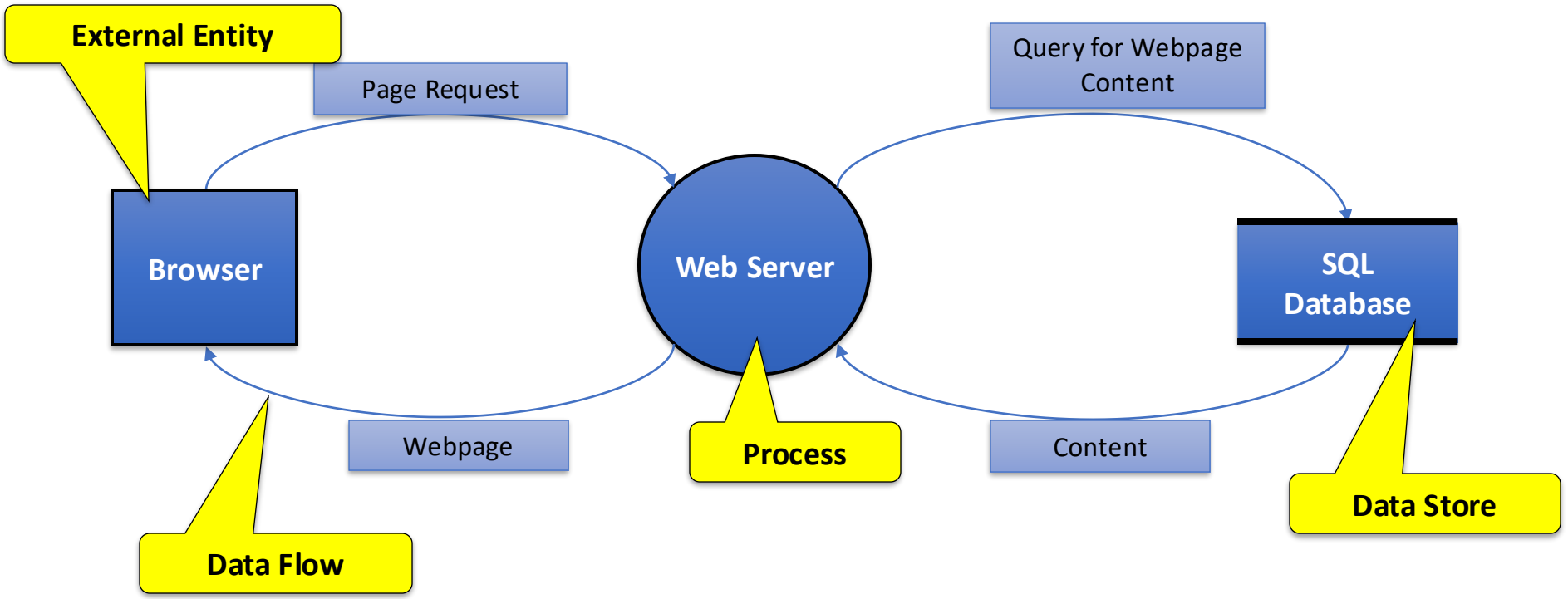
- **External interactors**
  e.g. clients, other systems, dependencies

- **Process**
  Architecture-centered functionality
  e.g. dispatcher, input validator

- **Datastore**
  e.g. database, file system

- **Data flow**
  Domain-specific explanation of data
  e.g. “Login Requests”

Threat Modeling – Data Flow Diagrams

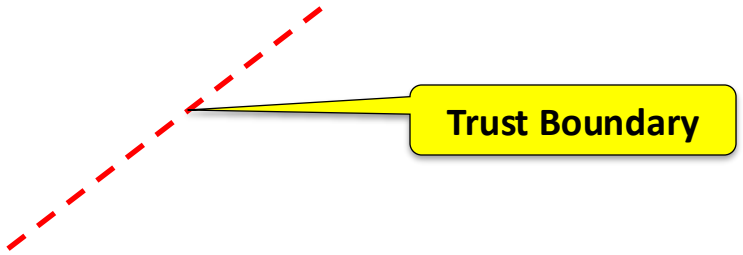


Trust Boundaries

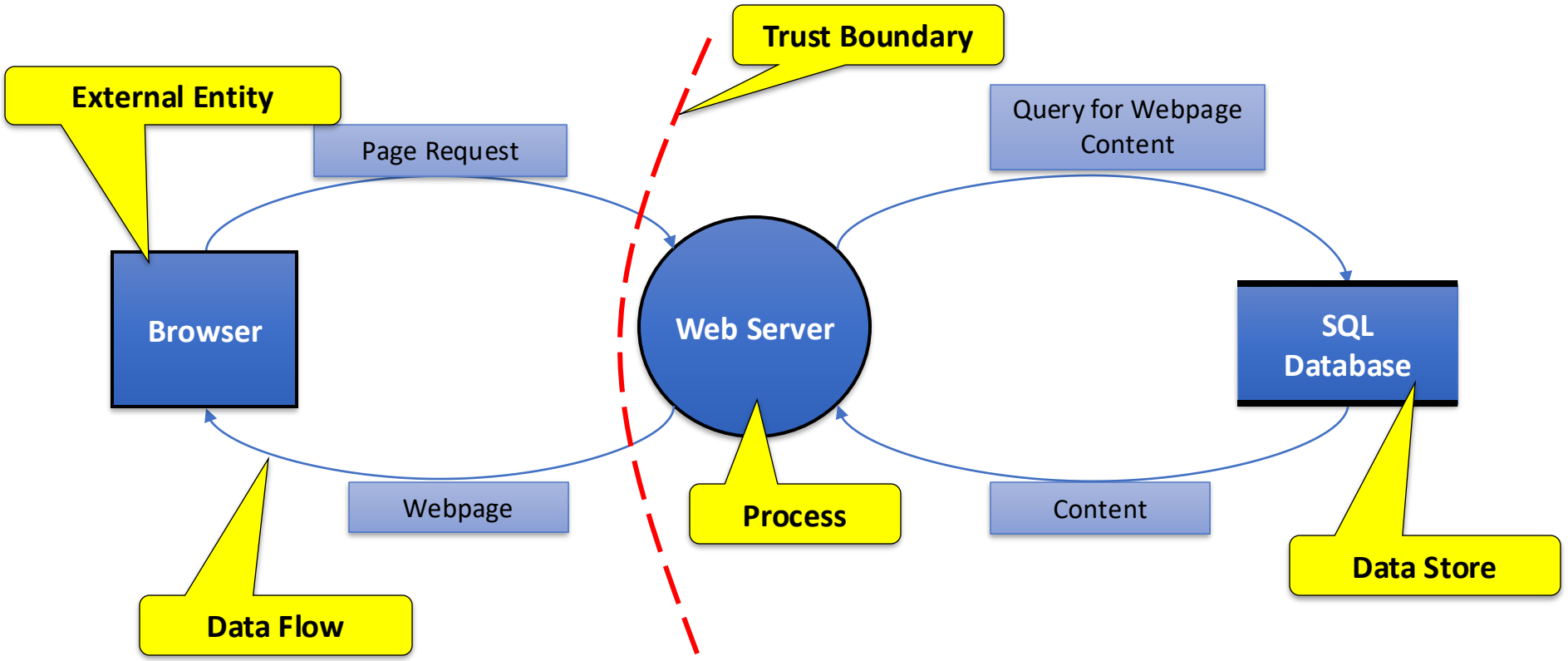
- Draw trust Boundaries: data from one party to another is not trusted

- Untrusted examples
  - Data from a web browser (e.g. external interactor)
  - Data from one machine to another

- Trusted examples
  - Data from another process within the same runtime environment
  - Data from the database



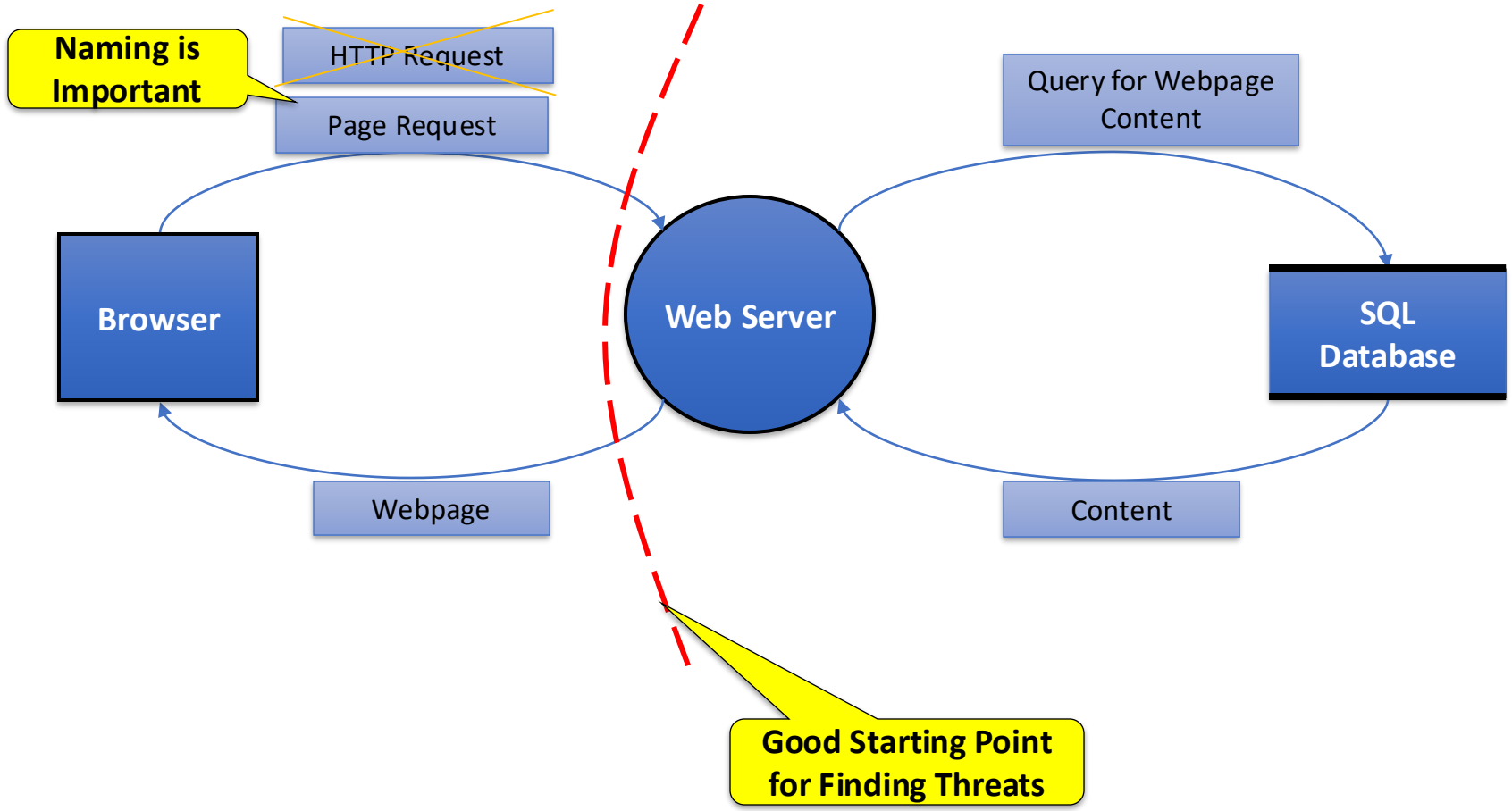
Threat Modeling – Data Flow Diagrams



- Trust boundary here, because we‘re not able to control the things that happen between a browser and our server

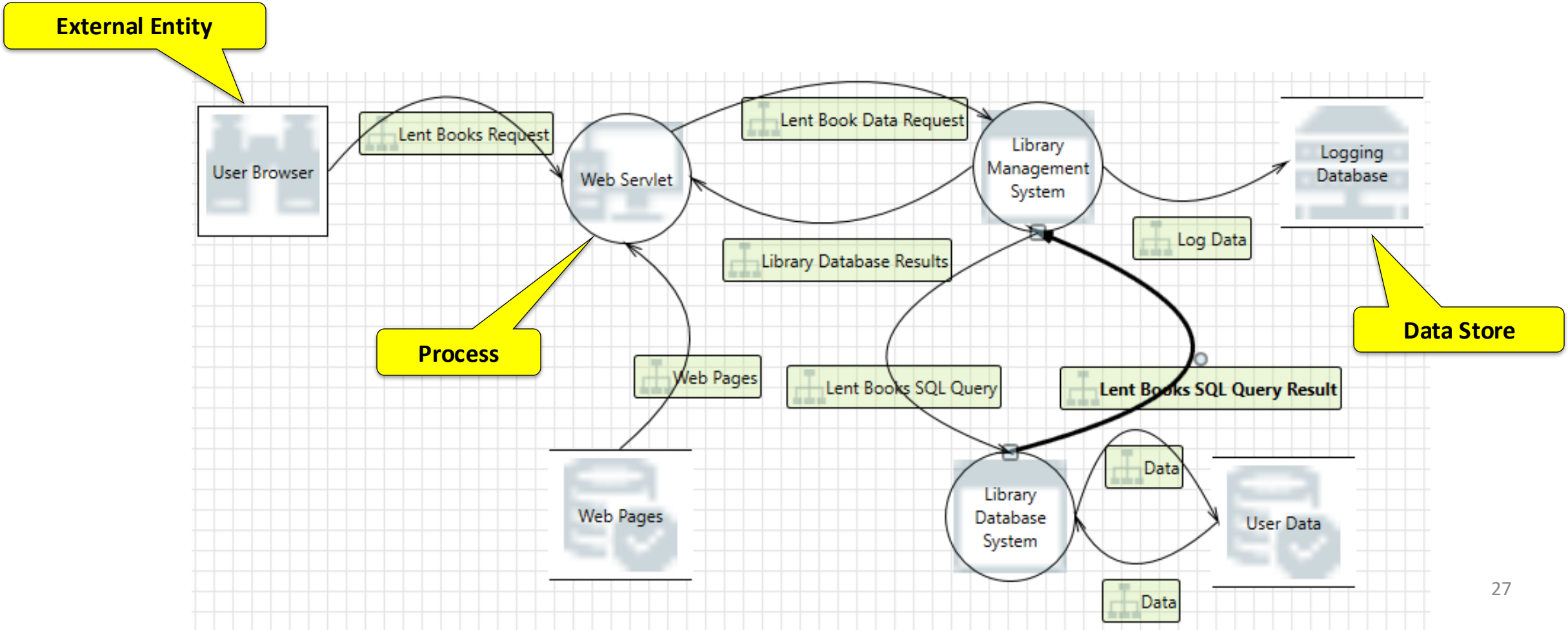
Threat Modeling – Data Flow Diagrams

- Use names that are as specific as possible.
  - HTTP Request could be anything (GET,POST,…)



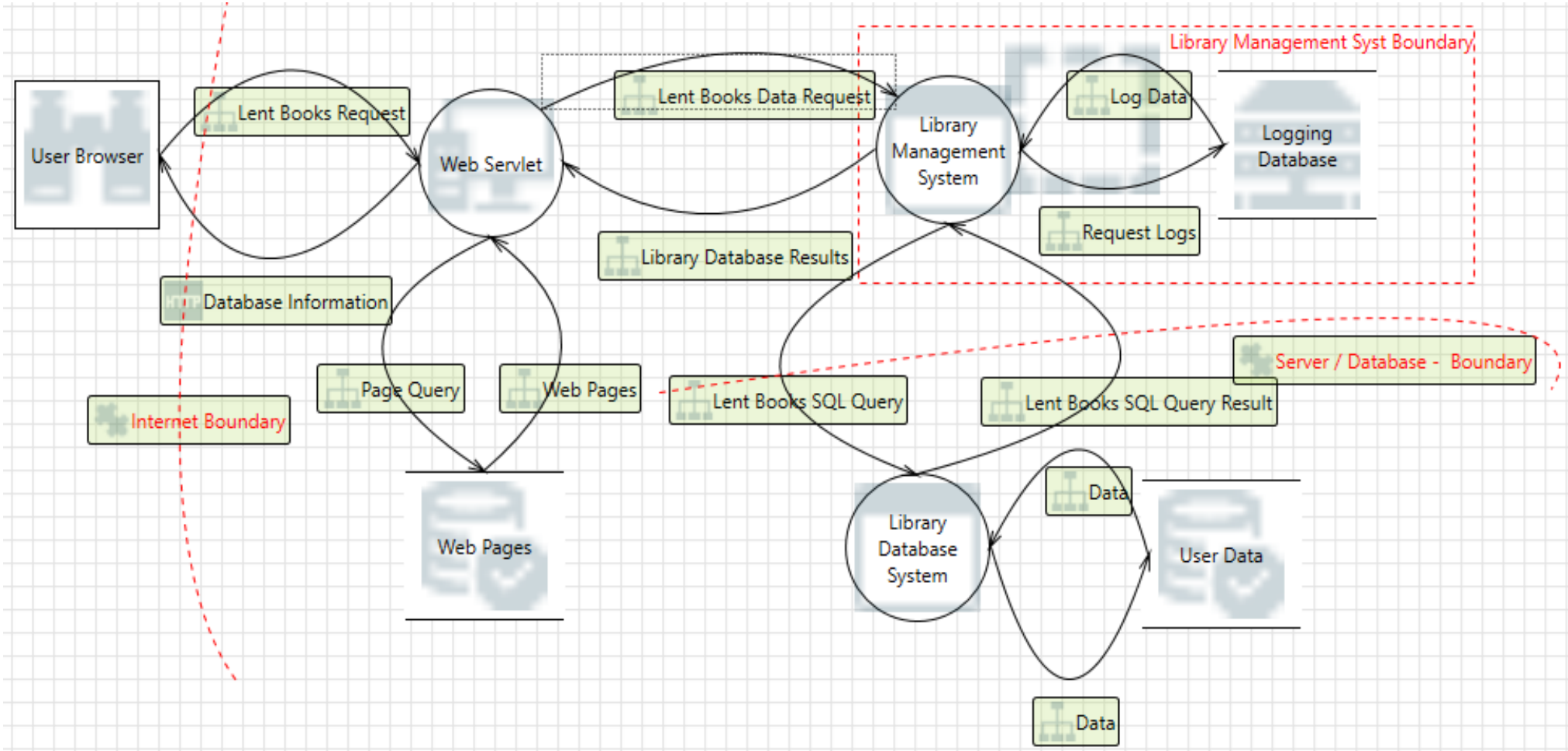
Data Flow Diagrams – Example Scenario

1. What is the scenario?
2. Is something missing?
3. Suggest trust boundaries.



Data Flow Diagrams – Example Scenario

1. What is the scenario?
2. Is something missing?
3. Suggest trust boundaries.



Threat Modeling

STRIDE per Element

Recap: Threat Categories - STRIDE

[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]

STRIDE per Element

- Performed against each and every component
- Pro‘s:
  - Systematic coverage
  - Independent analysis
  - Relating threats to components
- Cons:
  - Time consuming
  - Misses interaction-level threats
  - Duplicate threats if they are the same for different components

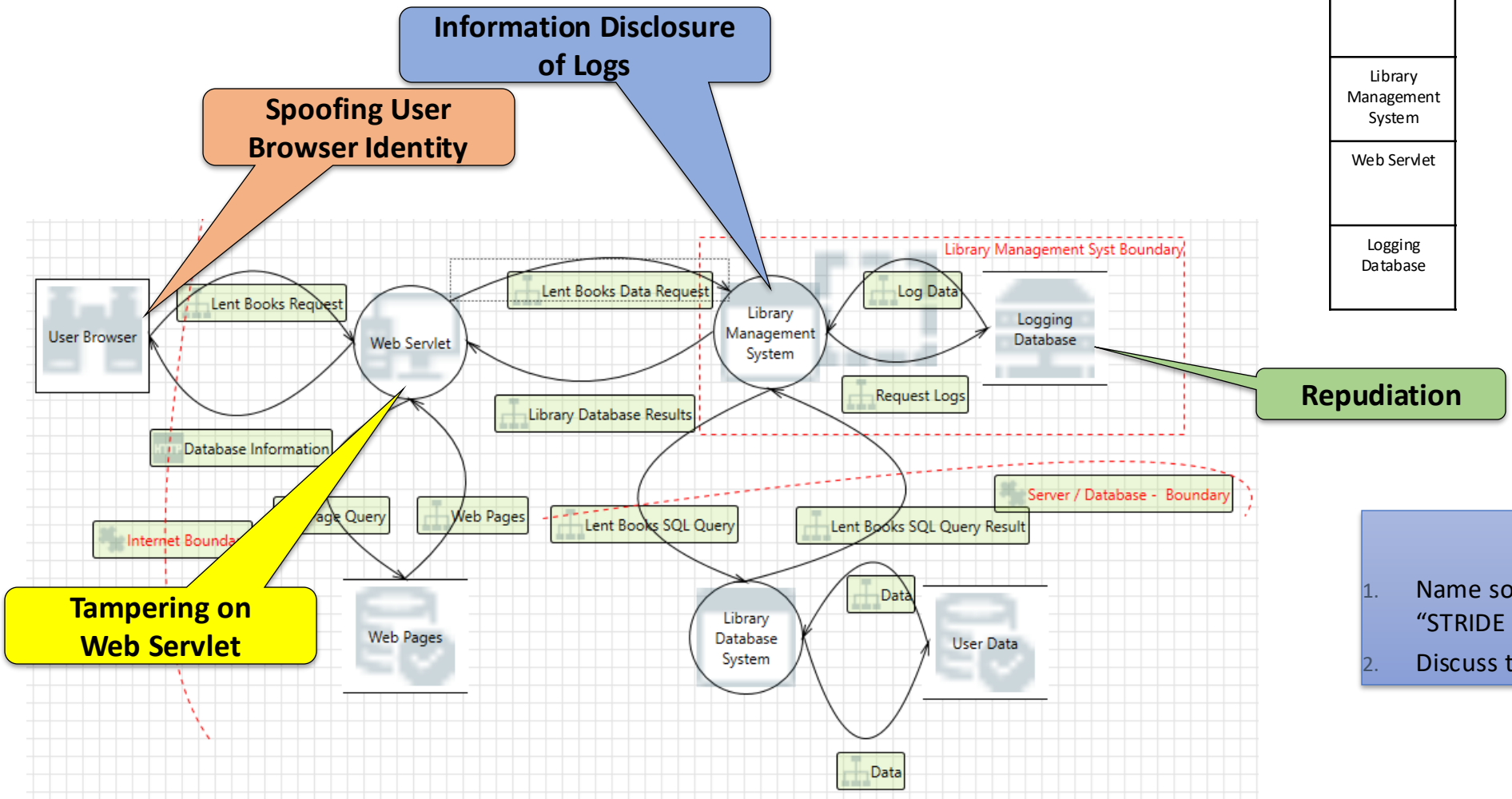
STRIDE per Element

- Analysis example for our scenario. Result can look like this:

Component	S	T	R	I	D	E
External Entity 	✓		✓	✓	✓	
Process 	✓	✓	✓	✓	✓	✓
Data Store 		✓	✓	✓	✓	
Data Flow 		✓		✓	✓	

Data Flow Diagrams – Mapping Threats

	S	T	R	I	D	E
User Browser						
Library Management System						
Web Servlet						
Logging Database						



Task:

1. Name some threats based on “STRIDE per element”.
2. Discuss the threats. Are they applicable?

STRIDE per interaction

UNIVERSITÄT BONN

- Performed against each interaction
- Pro‘s:
  - Focuses on real attack paths (across interactions instead of isolated components)
  - Less duplication
  - Less time consuming
- Cons:
  - Misses threats on component level (e.g., vulnerabilities)
  - Achieving full coverage is hard

[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]

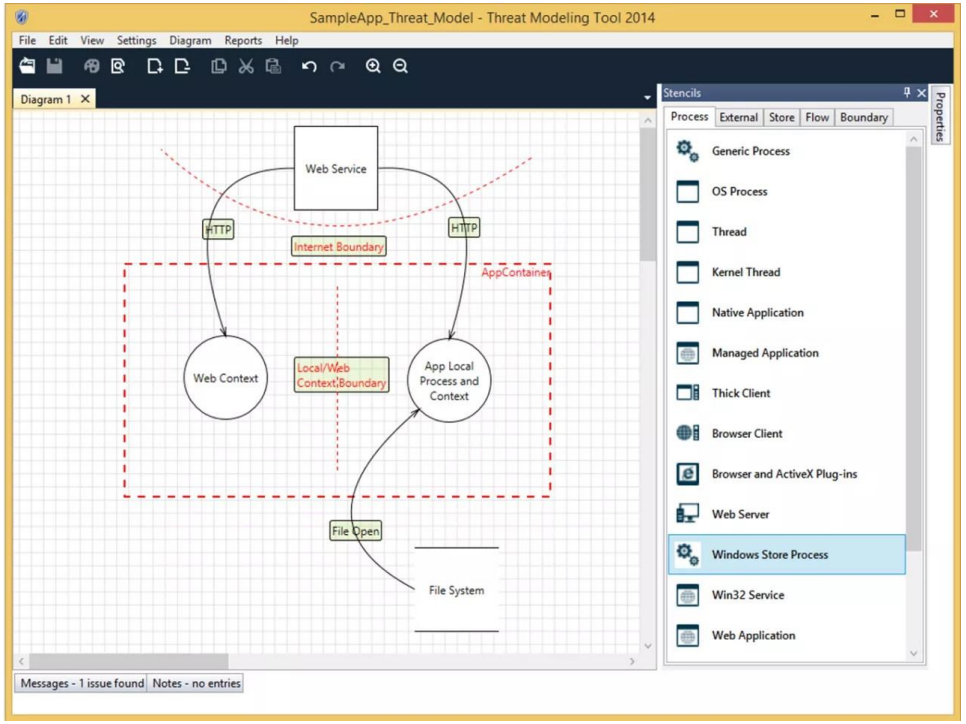
#	ELEMENT	INTERACTION	S	T	R	I	D	E
1	Process (Contoso)	Process has outbound data flow to data store.	x			x		
2		Process sends output to another process.	x		x	x	x	x
3		Process sends output to external interactor (code).	x		x	x	x	
4		Process sends output to external interactor (human).			x			
5		Process has inbound data flow from data store.	x	x			x	x
6		Process has inbound data flow from a process.	x		x		x	x
7		Process has inbound data flow from external interactor.	x				x	x
8	Data Flow (com-mands/ responses)	Crosses machine boundary		x		x	x	
9	Data Store (database)	Process has outbound data flow to data store.		x	x	x	x	
10		Process has inbound data flow from data store.			x	x	x	
11	External Interactor (browser)	External interactor passes input to process.	x		x	x		
12		External interactor gets input from process.	x					

MS Threat Modeling Tool | Tool of your choice

Behavioural Security Group

UNIVERSITÄT BONN

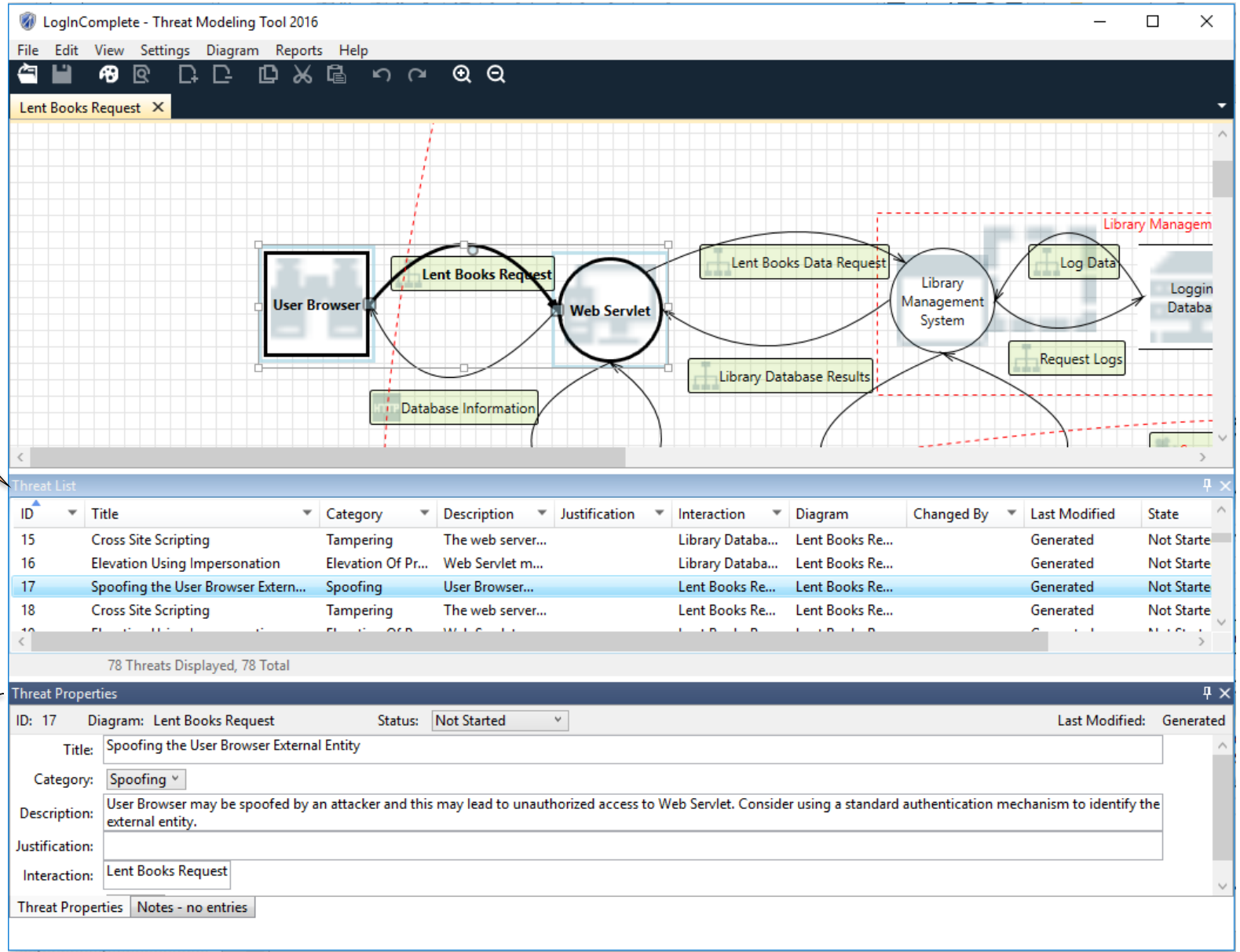
- **Architectural risk analysis tool**
  - Originally built on top of Visio
  - 2014 and 2016 versions largely updated
  - Follows STRIDE concept



- Provides Modeling & Analysis View
- You can use that in the exercises
  - But www.draw.io (or similar tools) work too

https://blogs.microsoft.com/microsoftsecure/2015/10/07/whats-new-with-microsoft-threat-modeling-tool-2016/
or
https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

Analysis View



List of threats

Detailed description

Analysis View

- Generate potential threats that must be mitigated
  - Based on components, their types and their connections
  - Forces you to describe mitigations
  - Helps you to record assumptions
  - Go directly to file a bug

- Threats are particularly bad when…
  - Flows cross boundaries

Threat Models != Architecture

UNIVERSITÄT BONN

- Threat models are specifically about modeling security
  - We do not create our architecture with them
- Architecture diagrams depict *subsystems responsibility*
  - What component exists and what is each one responsible for?
  - How does the system work?
- Threat Model is security focused
  - Based on *data flows*
  - Naming and characterizing the data that will be transported from one part of the system to another
  - Iteratively mitigating risks the tool tells us about
  - How could the system be attacked
- Big architectures may end up being one big bubble
  - It‘s not sufficient to look at modules on their own
- Small features in the architecture might constitute an entire process in a threat model

Tip: Naming Is Huge

- The flow names should be meaningful
  - Bad: “HTTP”, “Request”
  - Good: “Blog Content”, “Scrape Requests”
- Name your trust boundaries
  - Bad: “Machine boundary”, “Database boundary”
  - Good: “External Partner API Boundary“, “Identity Provider Boundary“
- Processes & external interactors
  - Bad: “server”
  - Good: “feedly.com”
- Ultimate test: outsiders should understand your system without much other info than the data flow diagram

More Tips for Threat Modeling

UNIVERSITÄT BONN

- Be honest with the process
  - Make sure the model represents reality
  - Consider *all* types of threats –
    code-level vulnerabilities are just a “for example”

- As with all modeling, use appropriate complexity
  - Overly-simplified?
    - Departs from reality
    - You get exactly what you put into it – no new knowledge
  - Overly-complicated?
    - Too much to analyze
    - “Check it off the list” syndrome

- Test your model

  Think of a specific security concern, then try to see where it fits in your threat model

Architectural Risk Analysis & Distrustful Decomposition

Key Principles

- Principle of Least privilege
- Defense in Depth
- Obviously No Vulnerabilities
- Assuming the attacker gets past that other defense, how do we:
  - Protect the inner parts of the system?
  - Limit the capabilities of the attacker?
  - Believe our critical subsystems are secure?

Permissions Challenges

UNIVERSITÄT BONN

- When programs are executed, the running code has permissions
  - via executing user, setuid, setgid, Java security manager, etc.

- Many programs need elevated (root) permissions to function
  - Web servers: listen & respond on HTTP, e.g. port 80
  - Email servers: listen & respond on SMTP, e.g. port 25
  - Browser plugins: website access, history access, …
  - Mobile apps: listening for text messages, accessing geo location
  - What permissions are *truly* needed, and when?

- Solution: design your architecture around your permissions needs

Distrustful Decomposition

UNIVERSITÄT BONN

- Decompose the system into separate processes with separate permissions
  - i.e. fork(), NOT threads. Why?
  - Threads still have shared memory!
- Communicate via inter-process communication (see next slides)
- Each operating-system process distrusts the other
  - i.e. validate the input from the other process
  - i.e. re-check credentials and integrity mechanisms
- Simplify the root-level subsystems to an extreme extent
- Outcomes
  - Complex code gets sandboxed -> reduce impact of an exploit
  - More security checks get incorporated throughout the code
  - Incorporate distrust at the architecture level

IPC: Simple Techniques

UNIVERSITÄT BONN

- Files
  - Simplest for most developers, but…
  - Files have no structure to them -> complexity in inputs!
    - File parsers are themselves among the most vulnerable components!
  - Locking gets tough. It must be respected (concurrency challenges -> complexity!!)

- Signals
  - Simple messaging, sometimes allow data to be transferred (e.g. Android)
  - Pre-defined by OS (Unix has ~30 of these)
  - e.g. SIGINT, SIGTERM, SIGKILL, SIGSEGV, SIGPOLL

IPC: More Techniques

Behavioural Security Group

UNIVERSITÄT BONN

- Clipboard
  - Can set and get clipboard programmatically
  - Users don’t like you messing with this
  - Also can be a security risk in itself!

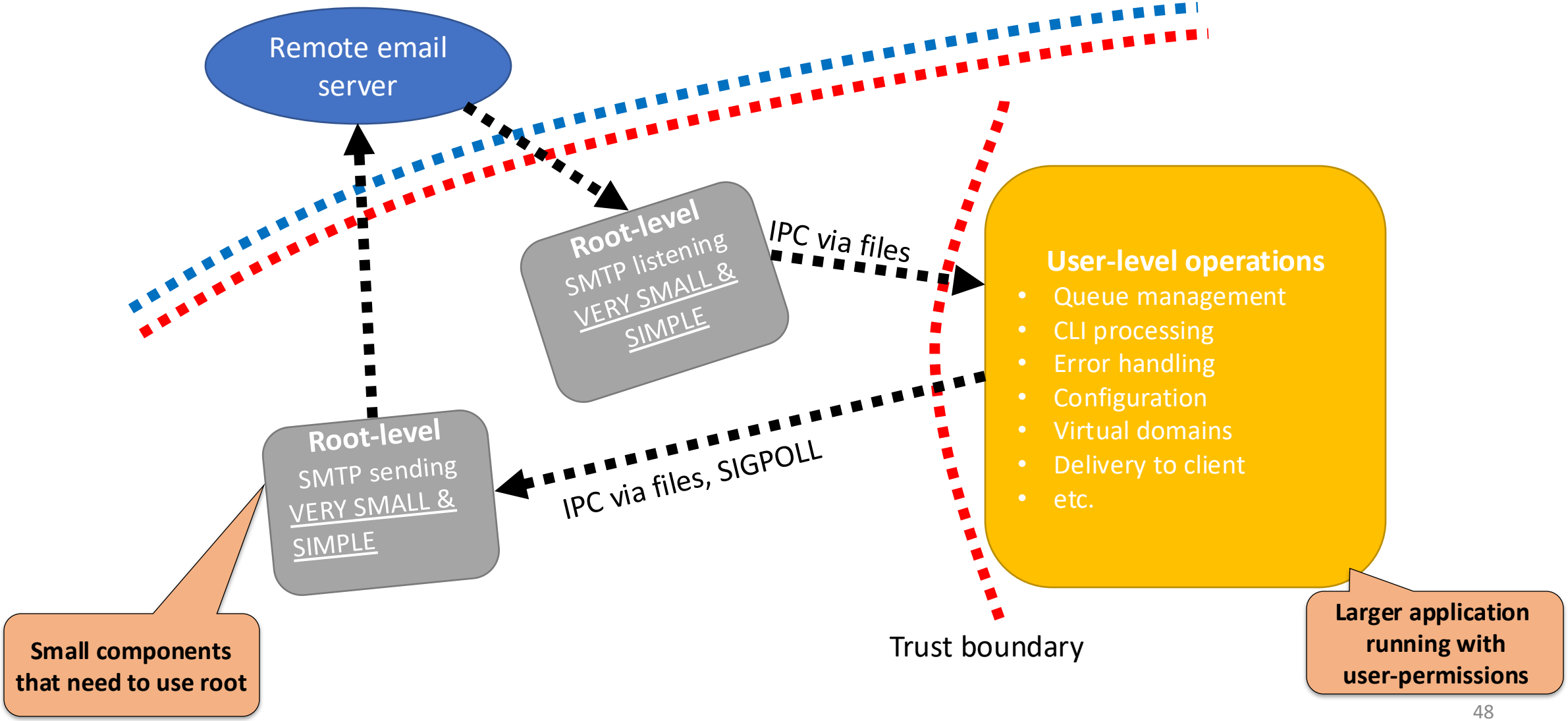
- Named pipes a.k.a. a FIFO buffer
  - Define a “virtual file” that one process can write to and another process reads from
  - Supported at the OS (file system) level
  - BTW: stdin and stdout are part of using *unnamed pipes*!
  - Pro: fast!! Cons: on the same system, still may need parser

IPC: Even more Techniques

- Sockets
  - Write to a traditional networked socket via the OS network interface
  - Sockets can be set to “loop back” to the original machine if needed
  - Pro: scalable to multiple systems! Con: slower

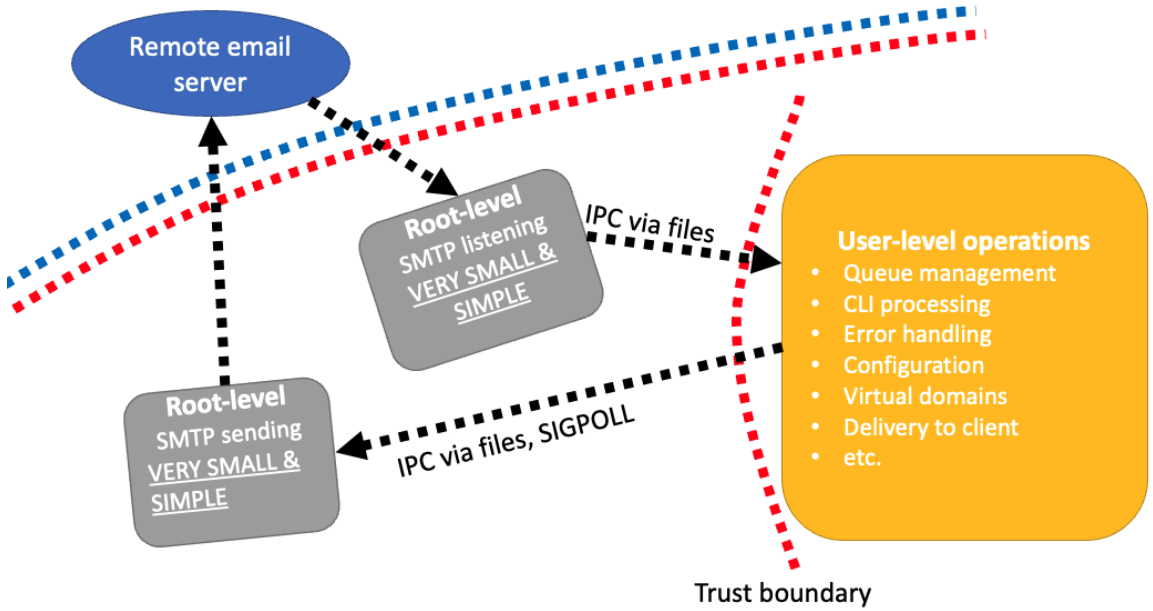
- MANY more IPC strategies: message passing, message queues, technology-specific solutions (e.g. Java RMI), memory-mapped files
  - Constantly evolving ecosystem due to security, reliability, simplicity, etc.

e.g. Qmail (designed as secure replacement for sendmail)



e.g. Qmail (designed as secure replacement for sendmail)

- Small components that really need root rights
  - For assigning ports
- A larger component that runs with user rights
  - Queue management…
- If there is a vulnerability there, the attacker does not have root rights



“Sort of” Distrustful Decomposition: Google Chrome

- Each browser tab is a separate process
  - IPC is accomplished primarily through files via their own IPC subsystem
  - Some IPC OS-specific mechanisms for synchronous message passing
  - Restrict rendering processes to their individual tabs

- BUT! Not *exactly* “Distrustful”
  - OS-level file permissions are not employed
  - Still a lot of threads and shared memory for performance
  - Vulnerabilities can occur where direct file access results in effective IPC

e.g. Gone Wrong: Stagefright

- Stagefright Android Vulnerability of 2015
  - Buffer overflow in video transcoding function that produces a preview thumbnail on Android text messages
  - Send a crafted video to a phone -> arbitrary code execution in messages
  - No user intervention required (i.e. previews are automatically generated)

- BUT!
  - Process listening to text messages requires high privileges
  - Process for producing thumbnails *also ran with high privileges*
  - Arbitrary code execution with these privileges -> cover your tracks

e.g. Gone Wrong: Stagefright

Telekom shuts down MMS due to Android vulnerabilities

Telekom has resorted to drastic measures to protect its customers from attacks exploiting the Stagefright vulnerabilities: effective immediately, MMS messages will no longer be delivered.

05.08.2015, 14:13 Uhr Lesezeit: 1 Min. | Security

e.g. Gone Wrong: Stagefright

Telekom shuts down MMS due to Android vulnerabilities

Telekom has resorted to drastic measures to protect its customers from attacks exploiting the Stagefright vulnerabilities: effective immediately, MMS messages will no longer be delivered.

05.08.2015, 14:13 Uhr Lesezeit: 1 Min. | Security

- Lessons learned:
  - Separate complex, non-root functionality from root ones
  - Avoid buffer overflows, of course.

e.g. Gone Wrong: Stagefright

Patch day: Critical vulnerabilities in Android's media framework – again

Google has released several security updates for Android devices. The company's own Pixel series will benefit most from these updates.

02.04.2019, 10:50 Uhr Lesezeit: 2 Min. | Security

Design & Project Implications

UNIVERSITÄT BONN

- Distrustful Decomposition is not something you want to procrastinate
  - Easier to build upon than bolt on
  - Introduces distrust at key points in the system
  - Introduces necessary complexity into your system

- IPC abstracted into a subsystem is often the next step after DD
  - Requires input validation and output normalization/sanitization
  - Requires defining the communication protocols