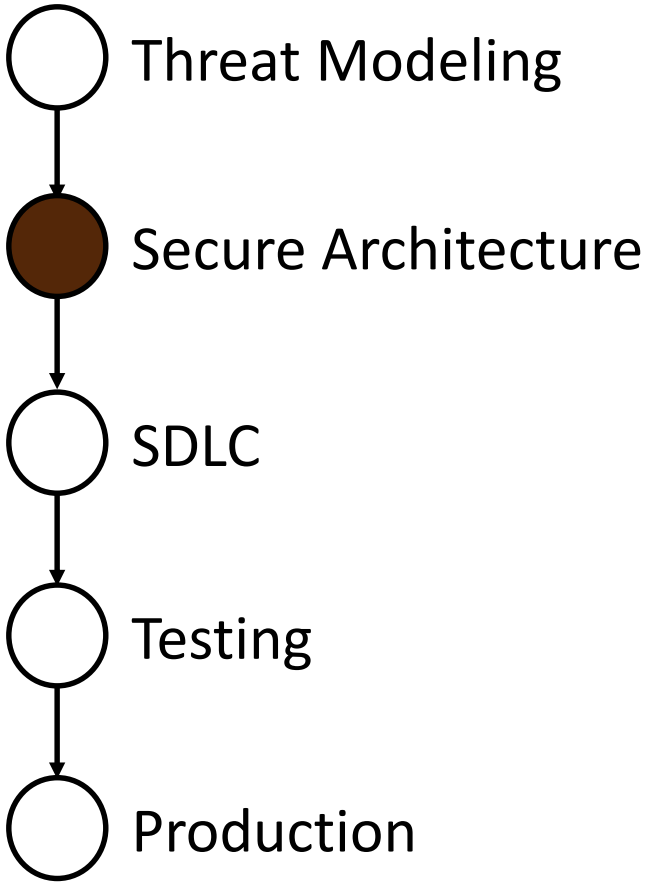
Application & Data Security

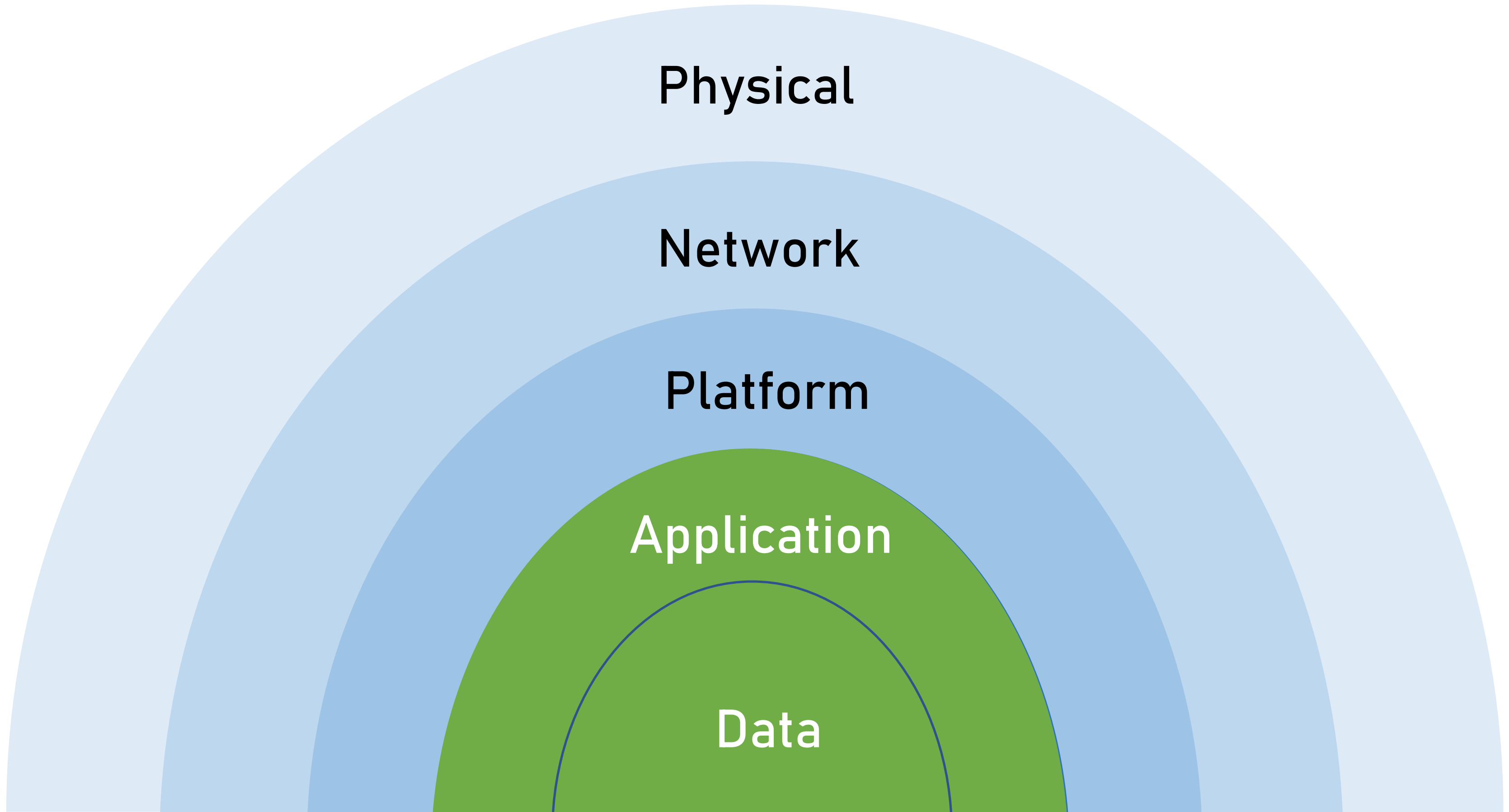
Memi Lavi
www.memilavi.com

Secure Architecture Process



Application & Data Security

Decide & Design



Application & Data Security

- The Architect is responsible for both
- This is where the main work happens
- Need to deal with a lot of threat types
- Touches all layers of the system

How Are We Going to Do That

- Security Topics
- Explain:
  - What it is
  - What it’s for
  - How to design / implement
  - Perhaps some code examples
    - There won’t be actual coding ☺

Authentication

What It Is

- Basically, the answer to the existential question:

Who Are You?

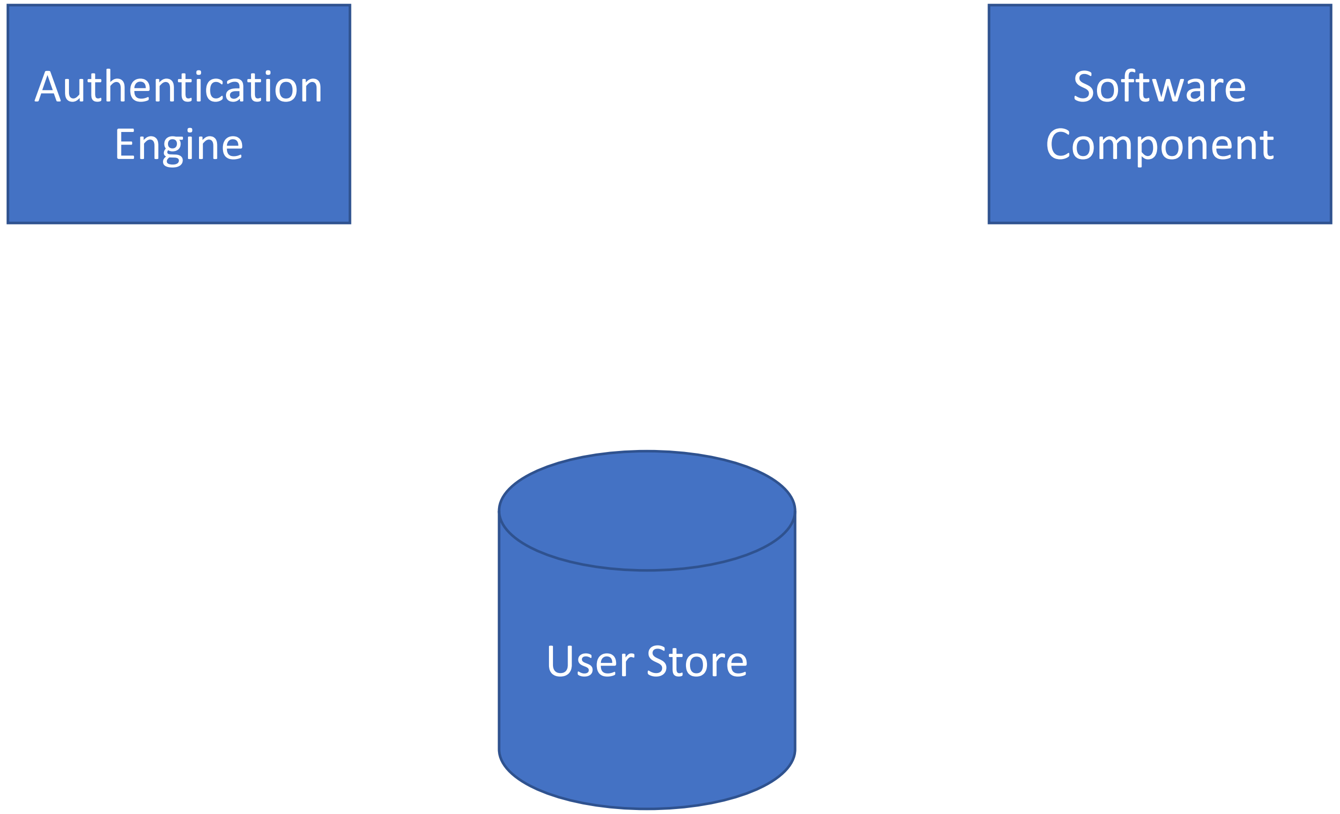
What It Is

- Verifying the identity of a thing
- Where *thing* can be:
  - Human
  - Computer
  - Software
  - And more…

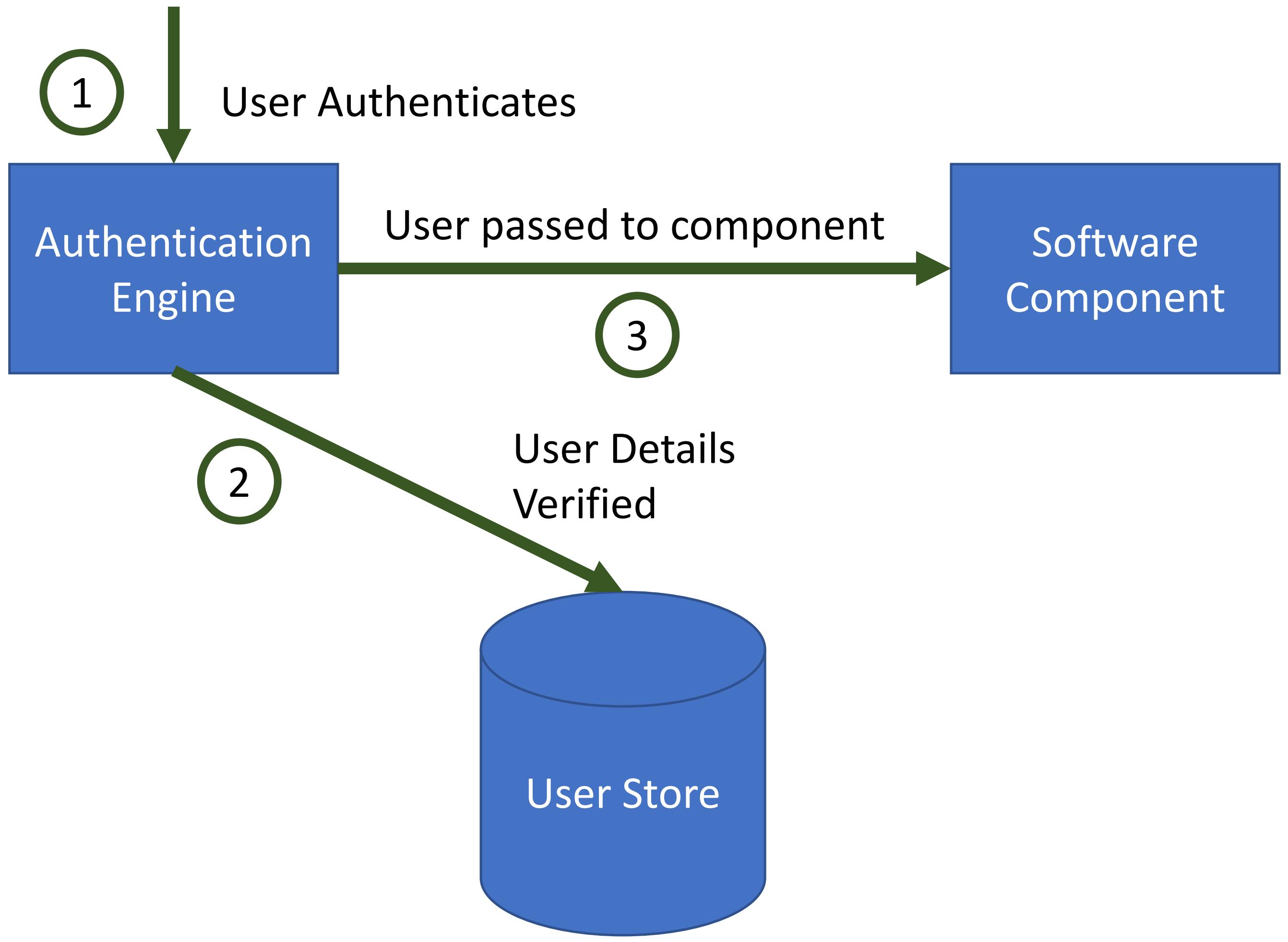
What It’s For

- When knowing who uses the system, we can:
  - Find out what he’s allowed to do
  - Allow / Deny certain actions
  - Log all activities for future research
  - Prevent data leak, data inconsistency

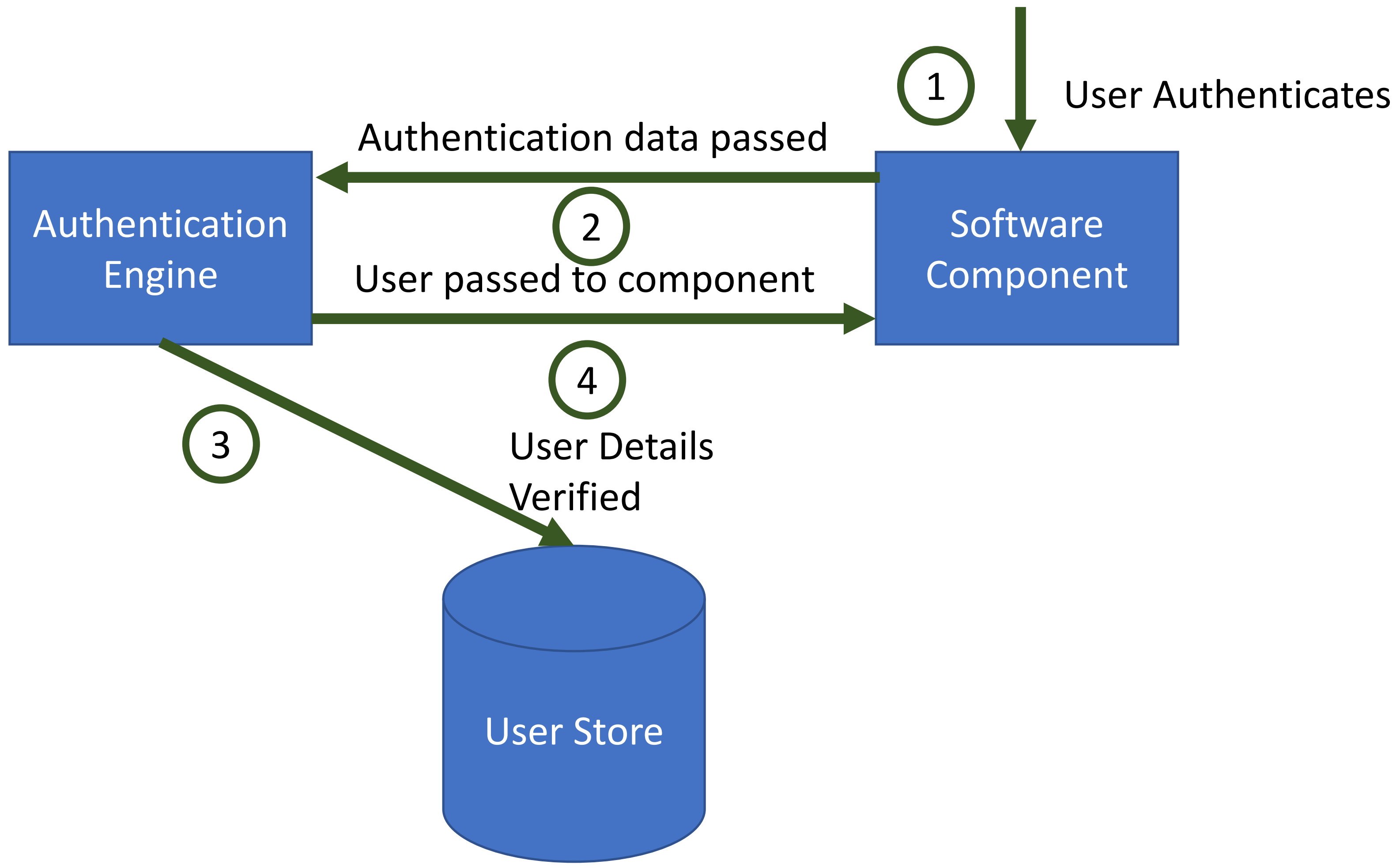
Authentication Design



Authentication Design Flow #1



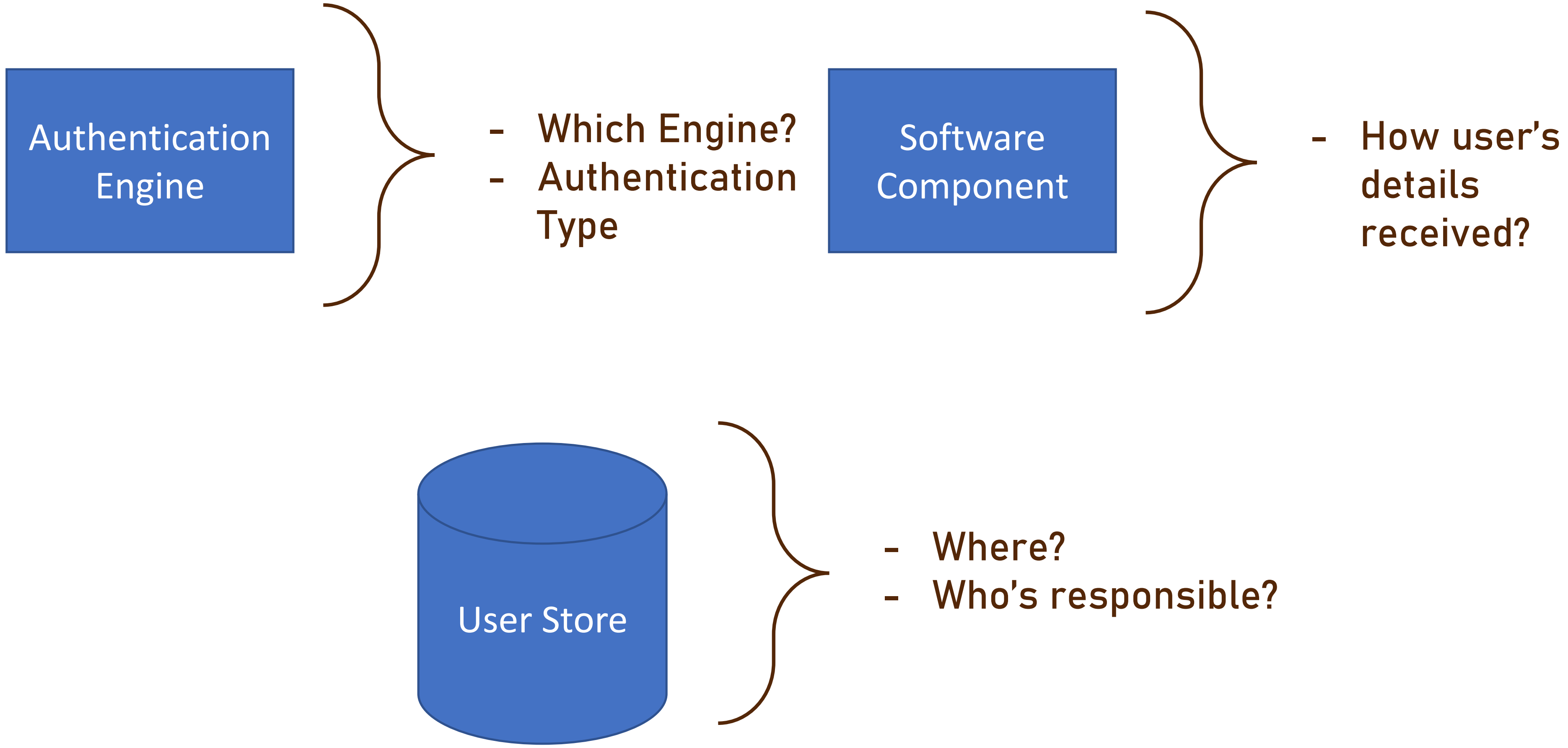
Authentication Design Flow #2



More Flows

- Component self-authenticates
- Authentication Engine uses another engine

Architectural Considerations



- Which Engine?
- Authentication Type

- How user's details received?

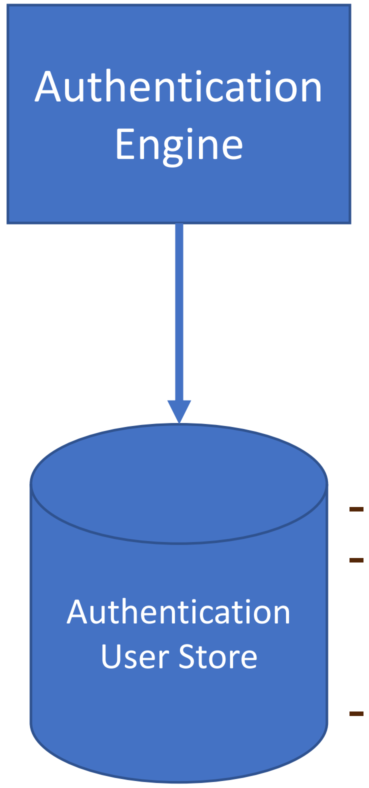
- Where?
- Who's responsible?

User Store

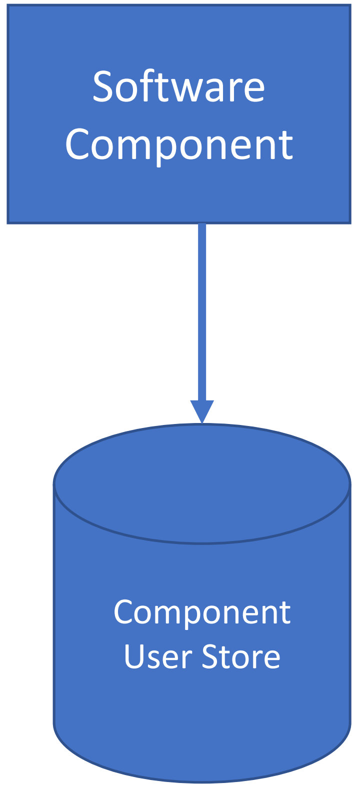
- Part of the Authentication Engine
- Part of the software component
- Hybrid

Usually the preferred way

Hybrid User Store



- Used for authentication only
- Usually contains user ID, password, organizational position, roles, etc.
- Maintained by the IT



- Used for the component's needs
- Usually contains business data, such as financial, personal, etc.
- Maintained by the component's developers

Authentication Engine

- Self developed
- 3rd party engine

Usually the preferred way

3rd Party Authentication Engine

- Robust
- Flexible:
  - Authentication Types
  - MFA
  - Roles

3rd Party Authentication Engine Cont.

- Contains user store (sometimes)
- Monitored
- Managed
- Provides SSO
- Standard based

3rd Party Authentication Engines

Cloud Based

Azure Active Directory

okta

AWS IAM

On-Premise

Active Directory

Centrify®

RSA SecurID®

Authentication Type

- Authentication is divided to three factors:

Something you know	(username / password, security question)
Something you have	(phone, smart card)
Something you are	(fingerprint, iris scan, other biometric data)

Selecting Authentication Type

Feasibility	(Not all computers / phones have fingerprint scanner)
Ease of use	(username / password is much quicker than Text)
Sensitivity	(Biometric data is much more difficult to hack than username / password)

Two Factor Authentication

- For extremely sensitive systems (banks, etc.)

Something you know

(username / password + code in Text)

Something you have

Two Factor Authentication

- For extremely sensitive systems (banks, etc.)

Something you know

(username / password + fingerprint)

Something you are

Two Factor Authentication

- For extremely sensitive systems (banks, etc.)

Something you have

(phone + fingerprint)

Something you are

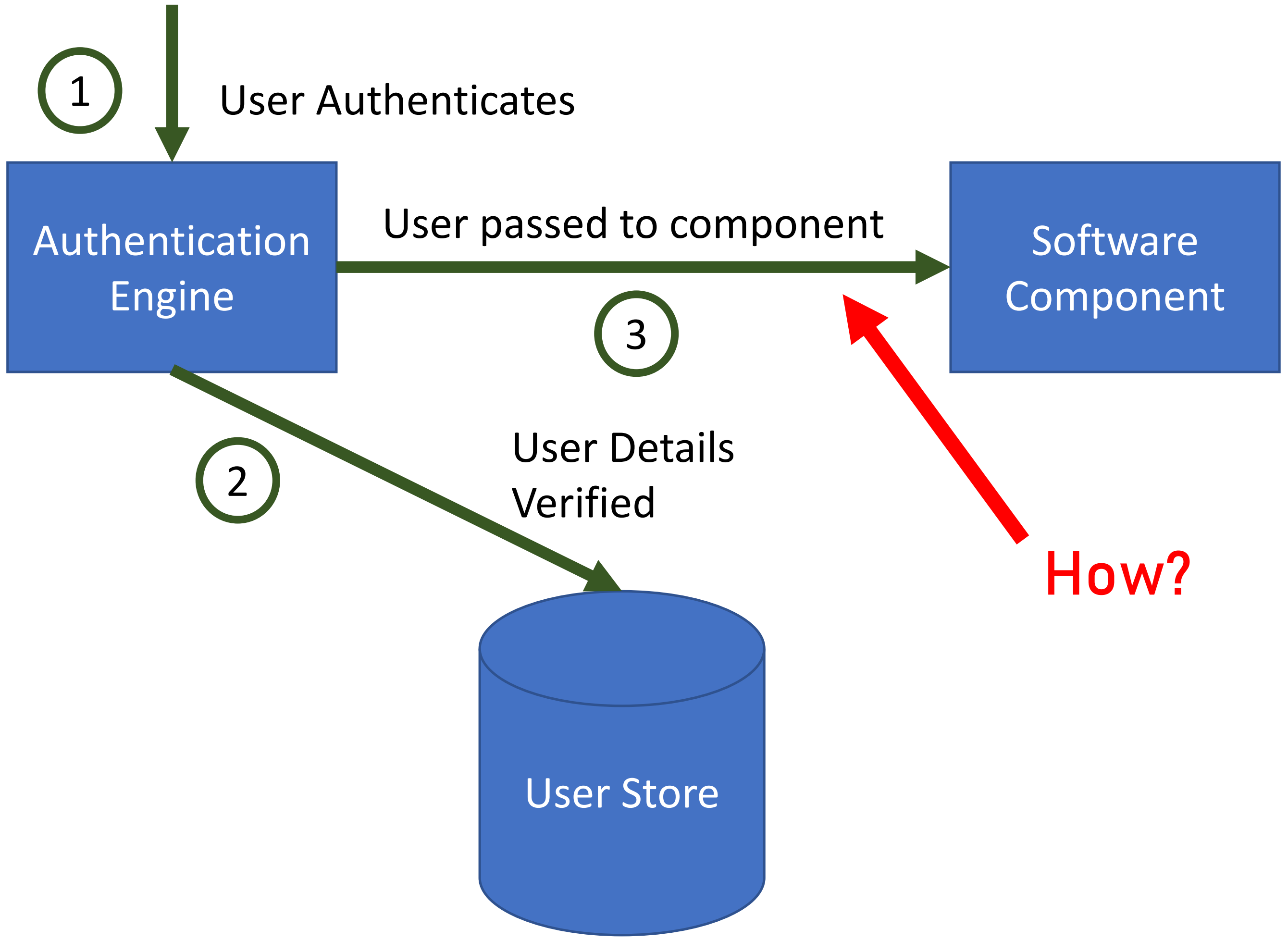
Two Factor Authentication

- Make sure the authentication engine supports two factor authentication
- Should not require development
- This decision directly affects the end users, so system analyst must be in the loop

Software Component

- MUST know who it is working with
    - Human, computer, software, etc.
- NEVER allow anonymous access

Software Component



User Token

- Contains the data generated from the Authentication Engine
- Must be secure
- Must be standard based

Authentication Protocols

- Define 3 things:
    - Participants in the authentication process
    - Flow of the authentication
    - Format of the data passed

Authentication Protocols

- NTLM
- Kerberos
- OAuth2

Authentication Protocols

Merge with REST API course, Section 12,

Lecture 2 and on (until 4)

Authentication And The Architect

- Decide on the Authentication Engine (with the IT)
- Decide on the user store
- Design the user store
- Decide on Authentication Type
  - Usually username / password or MFA

Authentication And The Architect Cont.

- Decide on the Authentication protocol
    - Prefer modern protocols such as OAuth2
- Authentication is usually a cross-organization effort
    - Work closely with the IT and dev team to implement it

Authorization

What It Is

- Basically, another answer to another existential question:

What Can You Do?

What It Is

- Assigning privileges to a thing
- Where *thing* can be:
  - Human
  - Computer
  - Software
  - And more…

What It Is

- Examples:
  - Can read data from the database
  - Cannot delete data from the database
  - Can run the *AddServiceRequest* method
  - Cannot access the *api/v1/orders* URL

What It’s For

- Limits access to users so that they won’t do unintended actions
- Even if authentication was hacked – authorization will limit its impact
- Protects mainly against data leak, data inconsistency

Principle of Least Privilege (PoLP)

- Authorization must be granted only for the intended tasks of the user, no more
- Example:
  - Users that require database read access – should not be granted database write access

Two Types of Authorization

Action Authorization

What actions are allowed or denied

Example:

- Allowed to add service request
- Allowed to update item inventory
- Denied from updating exam score

Data Authorization

Which data is accessible

Example:

- Cannot see data from other teachers
- Cannot see data of other counties
- Can see data of his own patients only

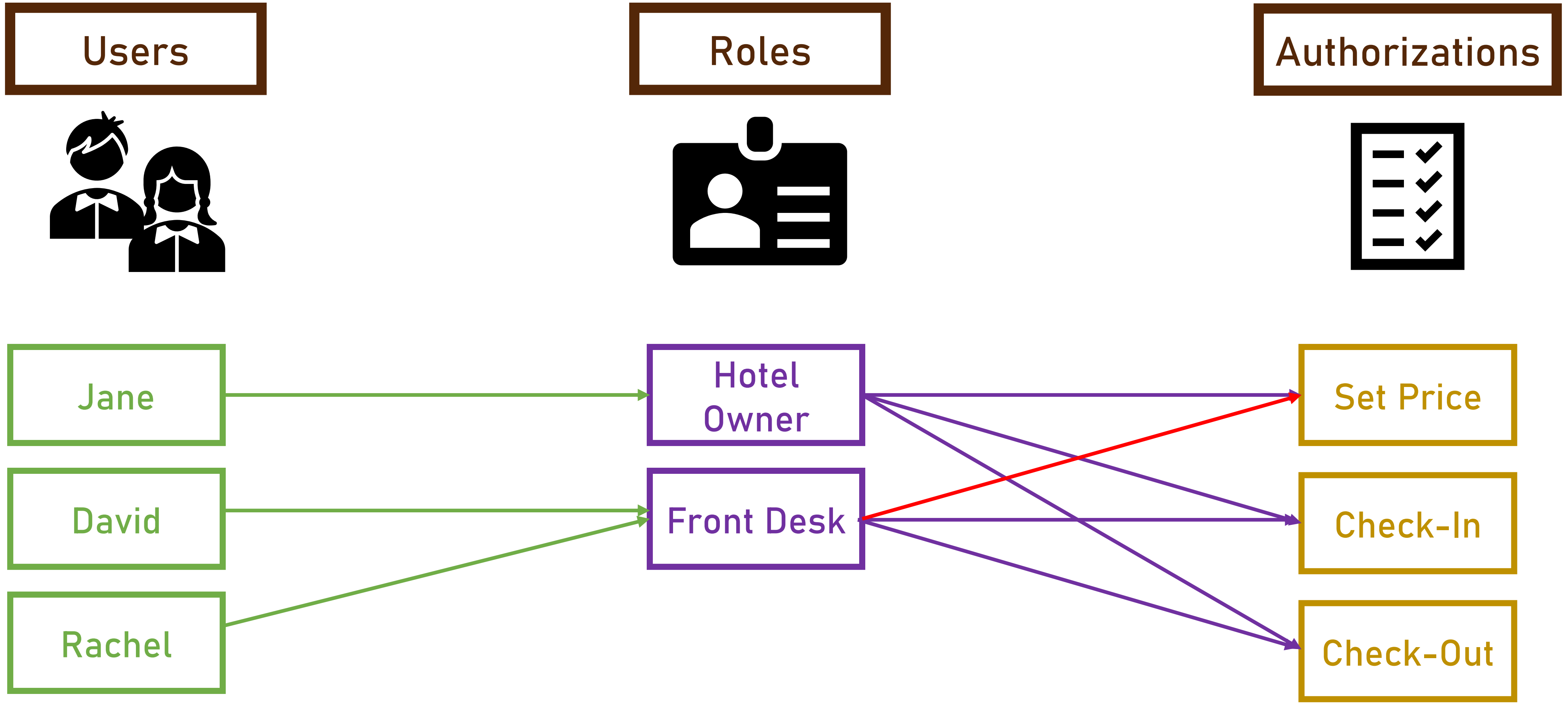
Role Based Access Control (RBAC)

- In the past, authorization was defined per user or user group
- Examples:
  - David is allowed to add items to the inventory
  - John is not allowed to read data of other doctors
- Very granular, extremely hard to maintain

RBAC to The Rescue!

Role Based Access Control (RBAC)

- With RBAC:



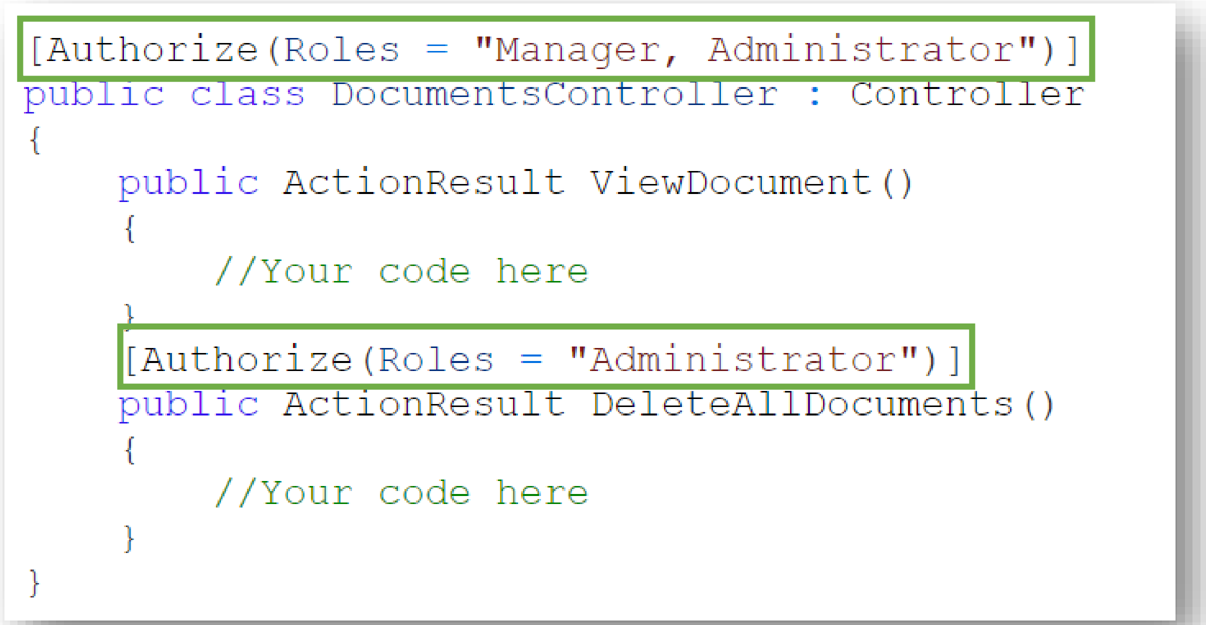
Role Based Access Control (RBAC)

- Where are roles managed?
  - In the Authentication Engine (ie. User Groups in Active Directory)
    - Passed to the component as part of the user’s token
  - In the component’s user store
    - Retrieved after receiving the user’s token from the Authentication Engine

RBAC Implementation

Action Authorization

Usually using built-in support in the development platform

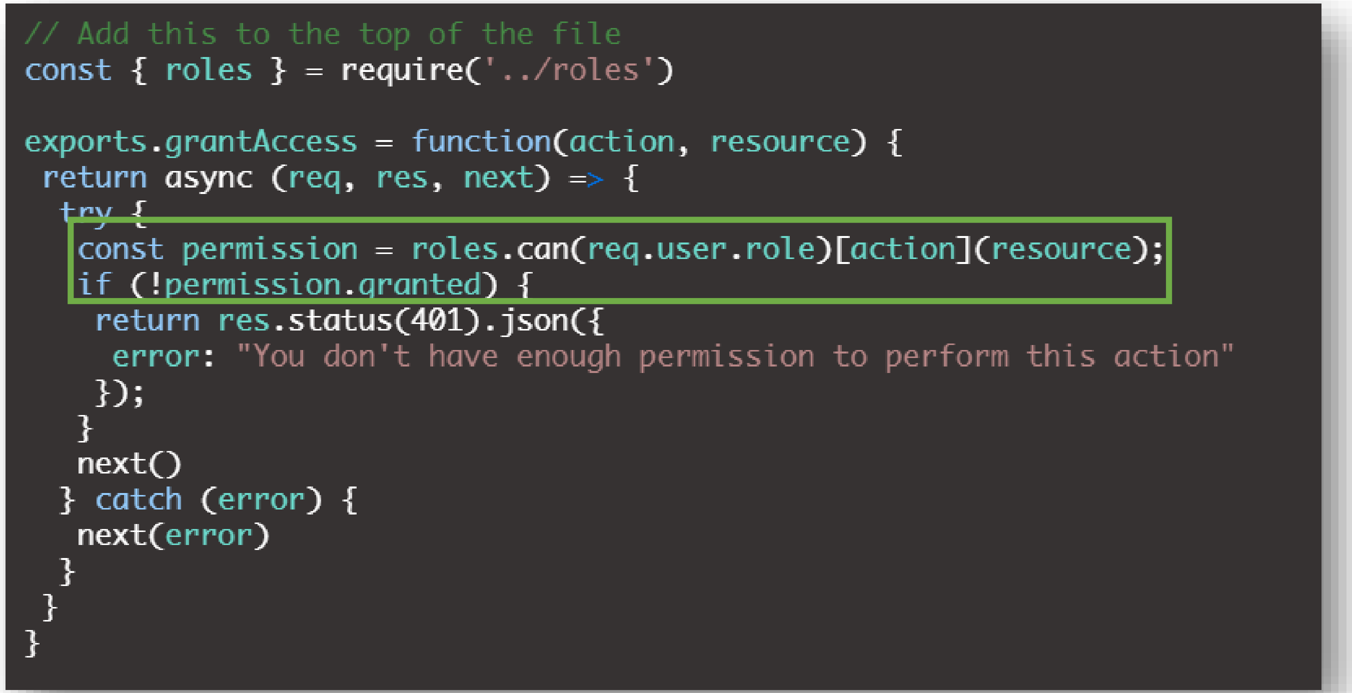


```
[Authorize(Roles = "Manager, Administrator")]
public class DocumentsController : Controller
{
    public ActionResult ViewDocument()
    {
        //Your code here
    }
    [Authorize(Roles = "Administrator")]
    public ActionResult DeleteAllDocuments()
    {
        //Your code here
    }
}
```

RBAC Implementation

Action Authorization

Usually using built-in support in the development platform



```
// Add this to the top of the file
const { roles } = require('../roles')

exports.grantAccess = function(action, resource) {
 return async (req, res, next) => {
  try {
   const permission = roles.can(req.user.role)[action](resource);
   if (!permission.granted) {
    return res.status(401).json({
     error: "You don't have enough permission to perform this action"
    });
   }
   next()
  } catch (error) {
   next(error)
  }
 }
}
```

Source: https://blog.soshace.com/implementing-role-based-access-control-in-a-node-js-application/

RBAC Implementation

Data Authorization

- Using the database's Row Level Security (RLS)
- Self development

```
Public MedData GetMedicalData(doctorId doc)  {

       using (var db = new MedContext())  {

              var medData=db.MedicalData
                  .Where(d=>d.ownerDoctor==doc)
                  .Order(d=>d.creationDate))
                  .ToList();

       }
}
```

Authorization And The Architect

- Decide on the roles management
    - Part of the Authentication Engine / part of the component
- If part of the component – design the roles' table, data access

Authorization And The Architect Cont.

- Design authorization implementation:
  - Recommend libraries for action authorization
  - Decide and design data authorization
- Work closely with the dev team and system analyst to define the roles

Secure Communication

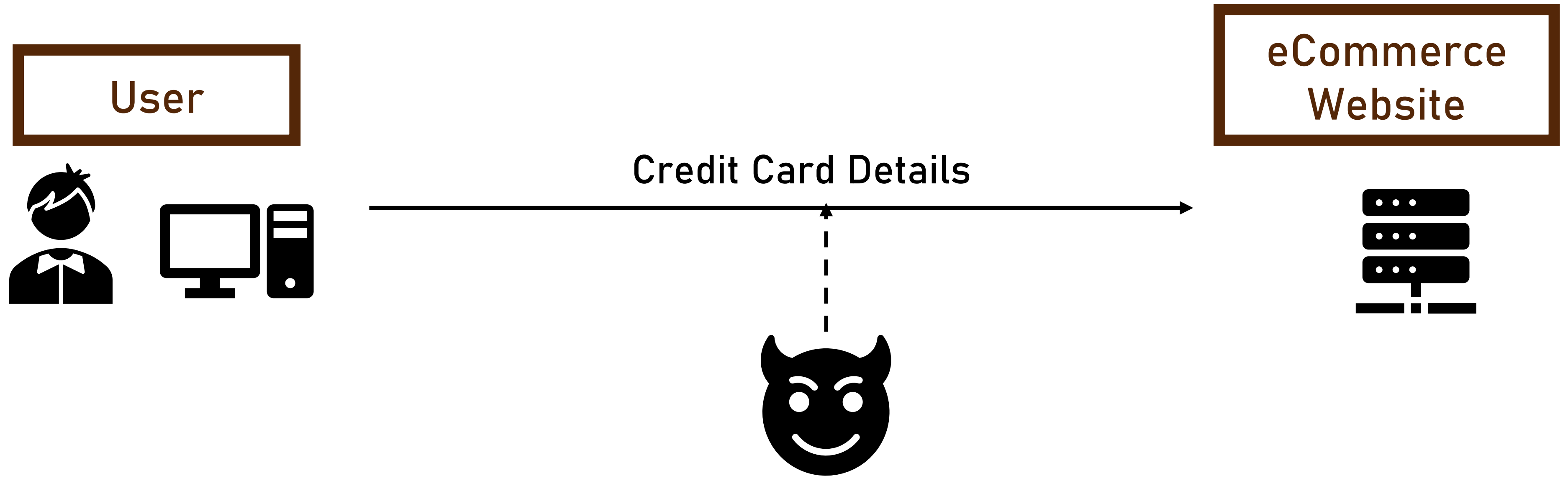
What It Is

- Making sure data in transit is secure
- Ensures:
    - Privacy
    - Data Consistency

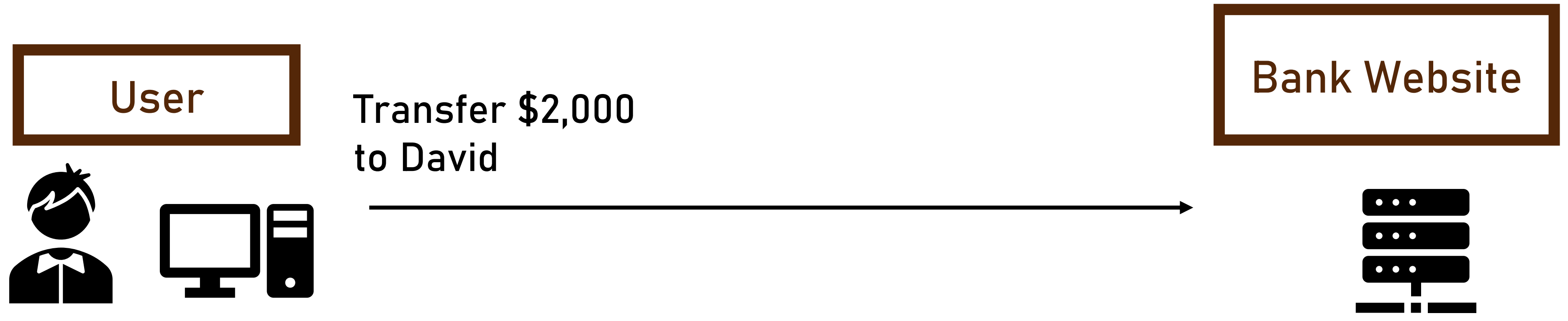
What It’s For

- Ensures sensitive data is not leaked to unauthorized recipients
- Prevents Man-In-The-Middle attacks
- Protects mainly against data leak, data inconsistency

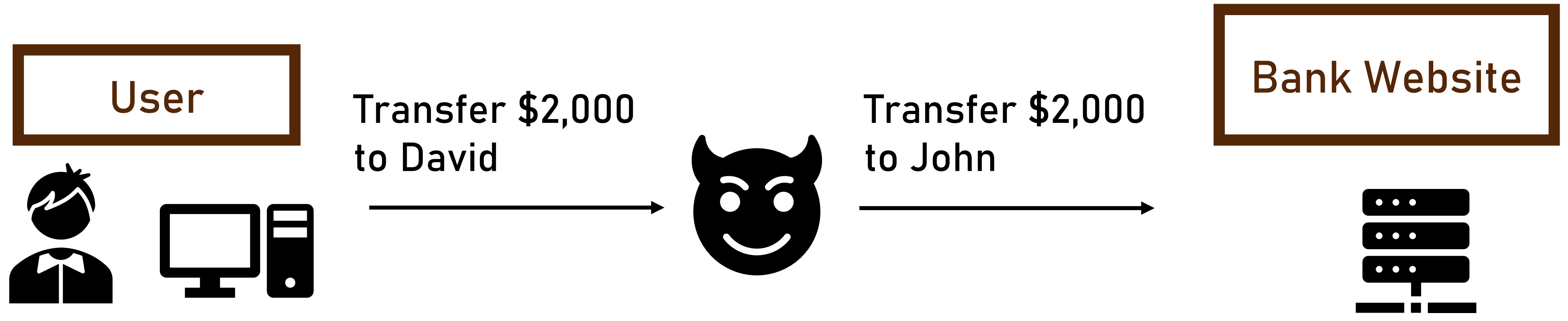
Privacy



Man-In-The-Middle Attack (MITM)



Man-In-The-Middle Attack (MITM)



Secure Communication

- Implemented using Secure Protocol
- Such as:
  - SSL
  - TLS

SSL

- Secure Socket Layer
- Developed by Netscape in 1995
- Has 3 versions
- All deprecated due to security flaws
- DO NOT USE!

TLS

- Transport Level Security
- First version released in 1999
- Latest version (1.3) released in 2018
- Versions 1.0 & 1.1 are deprecated
- DO NOT USE THEM!

TLS

- Uses symmetric cryptography to encrypt the data
- Unique keys are generated for each connection
- Uses shared secret between both the parties
- Parties are authenticated using public key cryptography
- Messages transmitted includes message integrity check

Implementing Secure Protocol

- Should be the IT responsibility
- Requires Certificate Authority & Management and more
- Work closely with the IT to ensure it’s supported

Secure Communication & The Architect

- Secure Communication does not affect the architecture
- Might have negligible effect on performance
- Insist on having secure protocol in place
- Work closely with the IT on that

Secure Code

What It Is

- Making sure the application’s code is secure
- Educating developers on the importance of secure coding
- Adopt secure coding practices

What It’s For

- Unsecure code can cause:
    - Disruption of service
    - Data leak
    - Data inconsistency
    - Data loss

Code Vulnerabilities

- Often a result of mistake or lack of awareness
- There are identified, wide-spread code vulnerabilities that must be handled in all systems
- The basic premise: Code is the last line of defense before the data, and must be well protected

SQL Injection

- One of the most common attacks
- Exploits SQL databases by inserting malicious code into SQL statements

SQL Injection Example

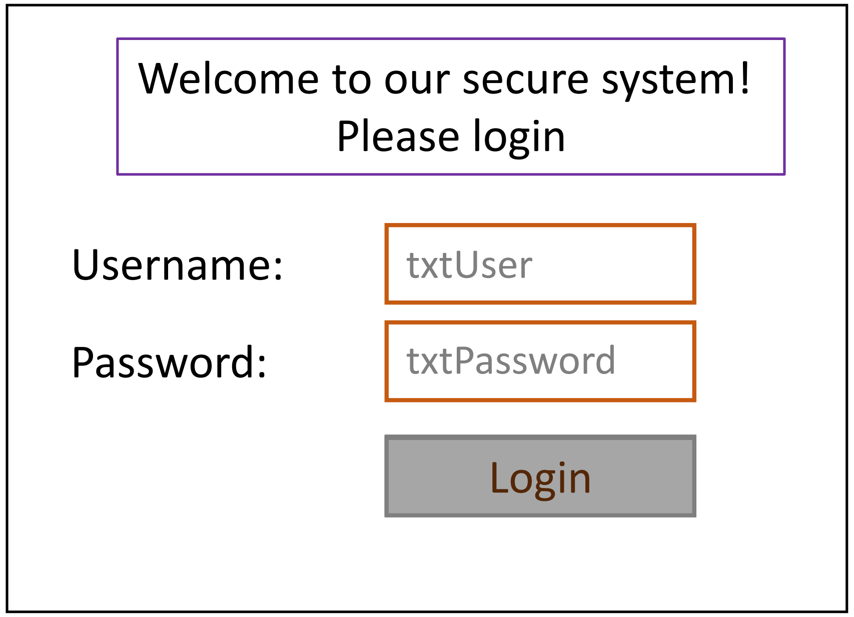
users Table

user_id	user_name	password
1	davidg	66peos
2	joank	q48v5#
3	johnb	9qxnf!x5

SQL Statement

```
“Select user_name, password from users
where user_name=‘“ + txtUser + “’ and
password=‘“ + txtPassword + “’”
```

Login Page



Welcome to our secure system!
Please login

Username: txtUser

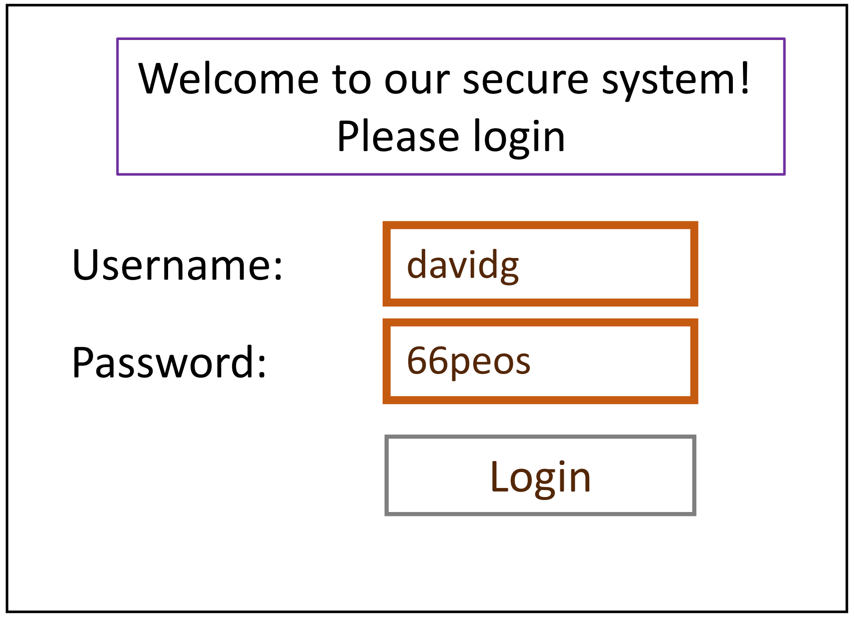
Password: txtPassword

SQL Injection Example

users Table

user_id	user_name	password
1	davidg	66peos
2	Joank	q48v5#
3	Johnb	9qxnf!x5

Login Page



Welcome to our secure system!
Please login

Username: davidg

Password: 66peos

SQL Statement

```
“Select user_name, password from
users where user_name=‘davidg’ and
password=‘66peos’”
```

SQL Injection Example

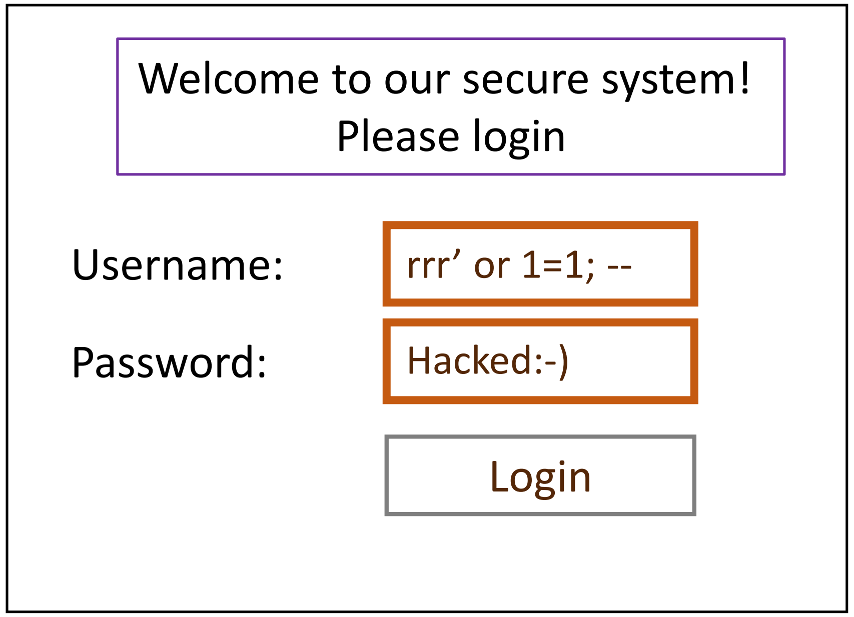
users Table

user_id	user_name	password
1	davidg	66peos
2	Joank	q48v5#
3	Johnb	9qxnf!x5

SQL Statement

```
“Select user_name, password from
users where user_name=‘rrr’ or 1=1;
-- and password=‘66peos’”
```

Login Page



Welcome to our secure system!
Please login

Username:	rrr’ or 1=1; --
Password:	Hacked:-)

SQL Injection Example

users Table

user_id	user_name	password
1	davidg	66peos
2	Joank	q48v5#
3	Johnb	9qxnf!x5

Login Page

Welcome to our secure system!
Please login

Username:	rrr’ or 1=1; DROP TABLE users: --
Password:	Hacked:-)

Login

SQL Statement

```
“Select user_name, password from
users where user_name=‘rrr’ or 1=1;
DROP TABLE users; -- and
password=‘66peos’
```

SQL Injection Mitigation

- ALWAYS validate input

- Use parameterized query

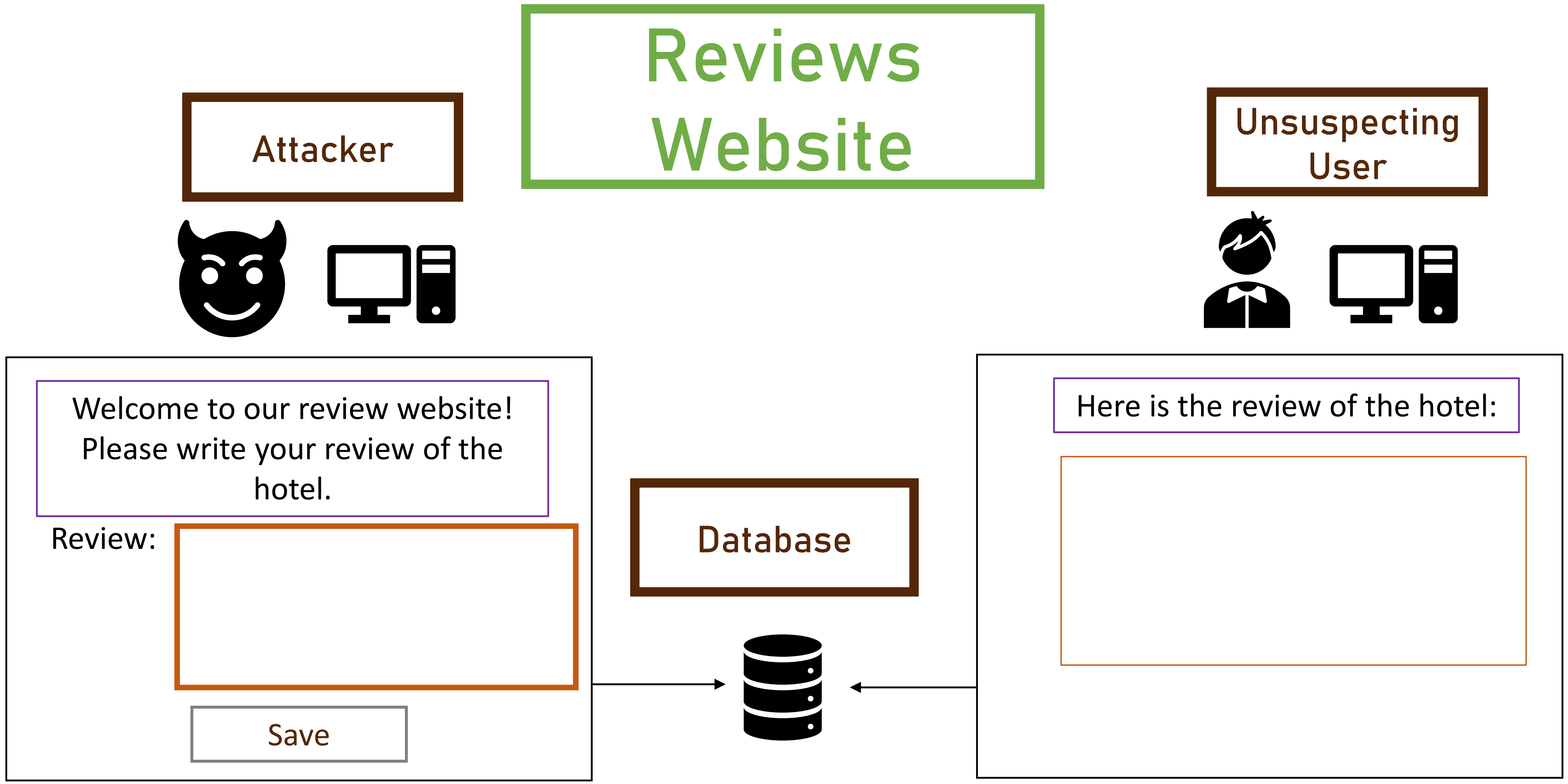
```
String sql="Select user_name, password from users where user_name=@userID
and password=@pwd"

SqlCommand cmd=new SqlCommand(sql, con);
cmd.Parameters.AddWithValue("@userId", txtUser);
cmd.Parameters.AddWithValue("@pwd",txtPassword);
```

Cross-Site Scripting (XSS)

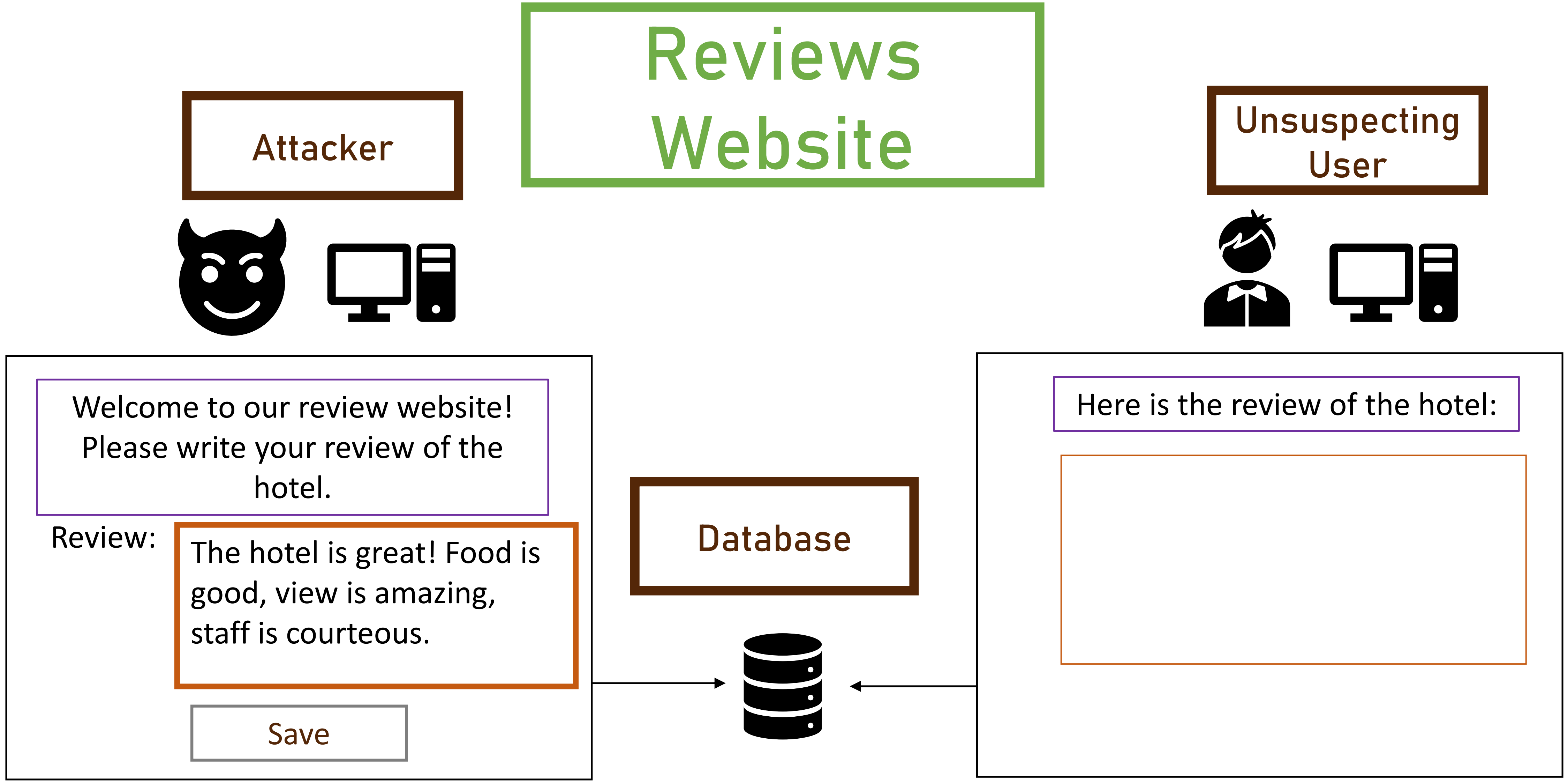
- One of the most common attacks
- Injects malicious scripts to an unsuspecting user’s browser

Cross-Site Scripting Example

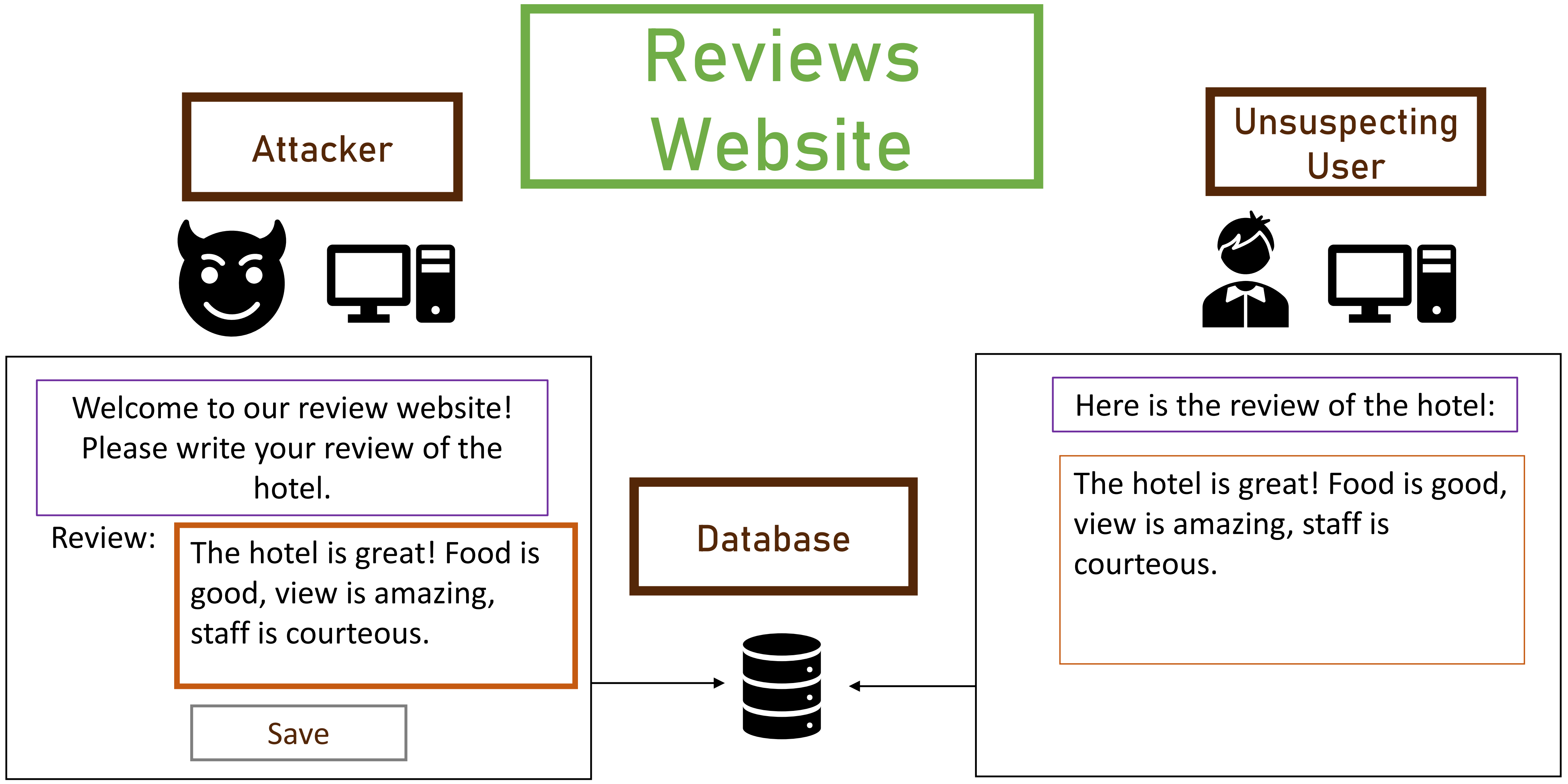


Reviews Website
Attacker
Unsuspecting User
Welcome to our review website! Please write your review of the hotel.
Review:
Save
Database
Here is the review of the hotel:

Cross-Site Scripting Example


Reviews Website
Attacker
Unsuspecting User
Welcome to our review website! Please write your review of the hotel.
Review:
The hotel is great! Food is good, view is amazing, staff is courteous.
Save
Database
Here is the review of the hotel:

Cross-Site Scripting Example



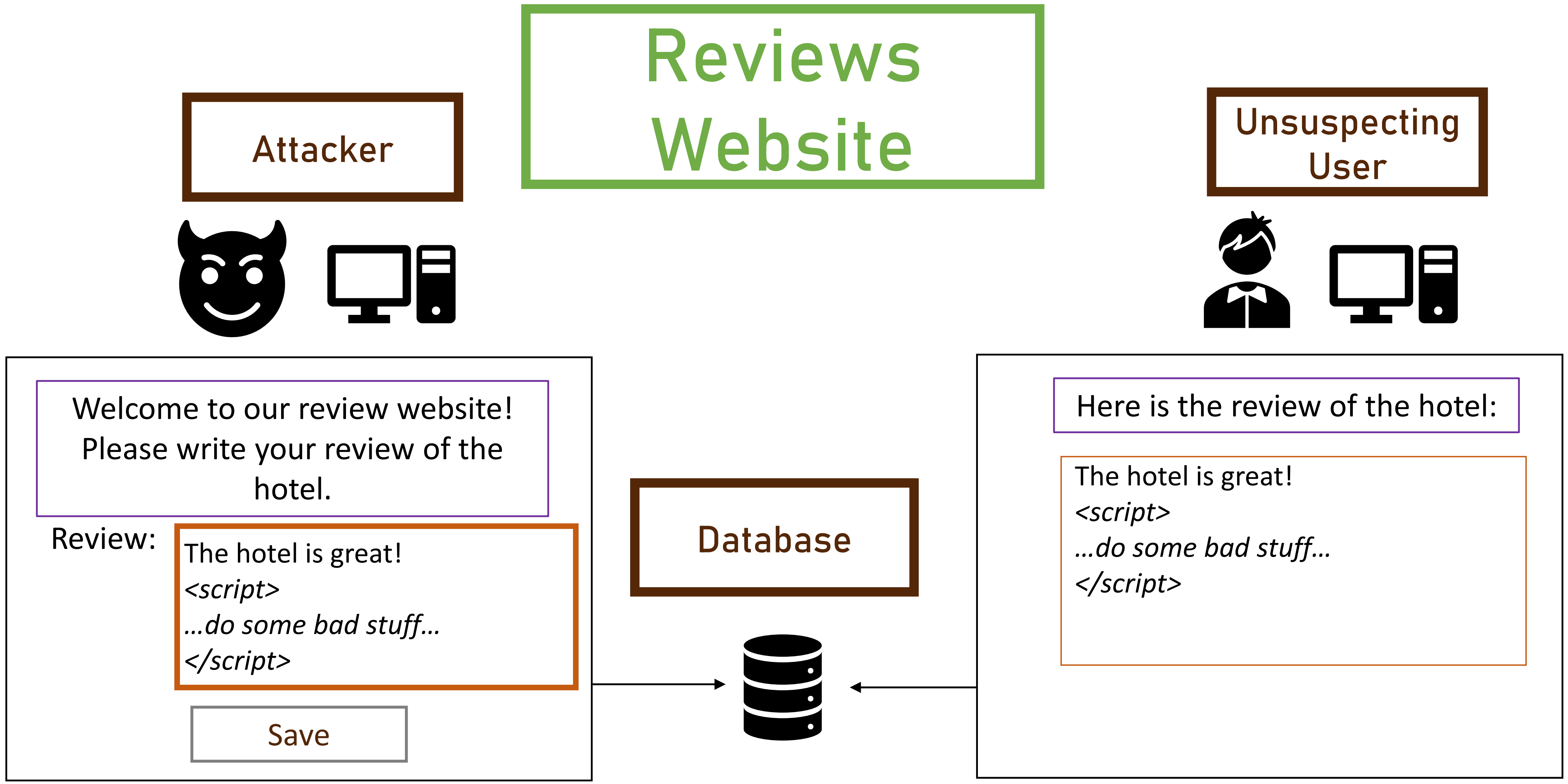
Welcome to our review website! Please write your review of the hotel.

Review: The hotel is great! Food is good, view is amazing, staff is courteous.

Here is the review of the hotel:

The hotel is great! Food is good, view is amazing, staff is courteous.

Cross-Site Scripting Example



Welcome to our review website! Please write your review of the hotel.

Review:

```
The hotel is great!
<script>
...do some bad stuff...
</script>
```

Here is the review of the hotel:

```
The hotel is great!
<script>
...do some bad stuff...
</script>
```

Cross-Site Scripting Mitigation

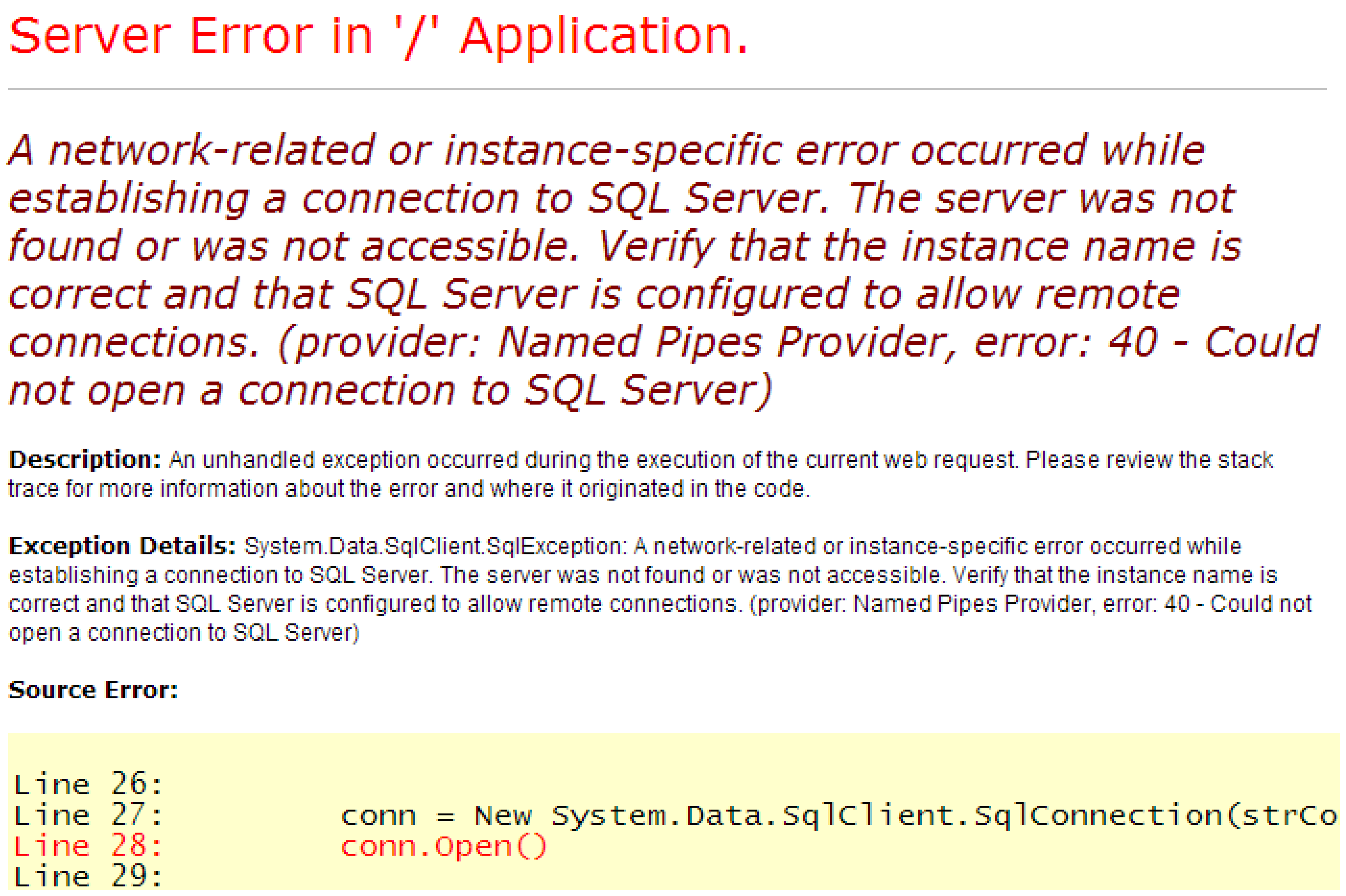
- ALWAYS validate input
- HTML-Encode the input

$$\texttt{<script>} \longrightarrow \texttt{&lt;script&gt;}$$

Data Exposure

- Exposing data about the inner-workings of the system
- Exposes the system to attacks focused on its actual structure
- Usually a result of a too-revealing error messages

Data Exposure Example



Server Error in '/' Application.

A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)

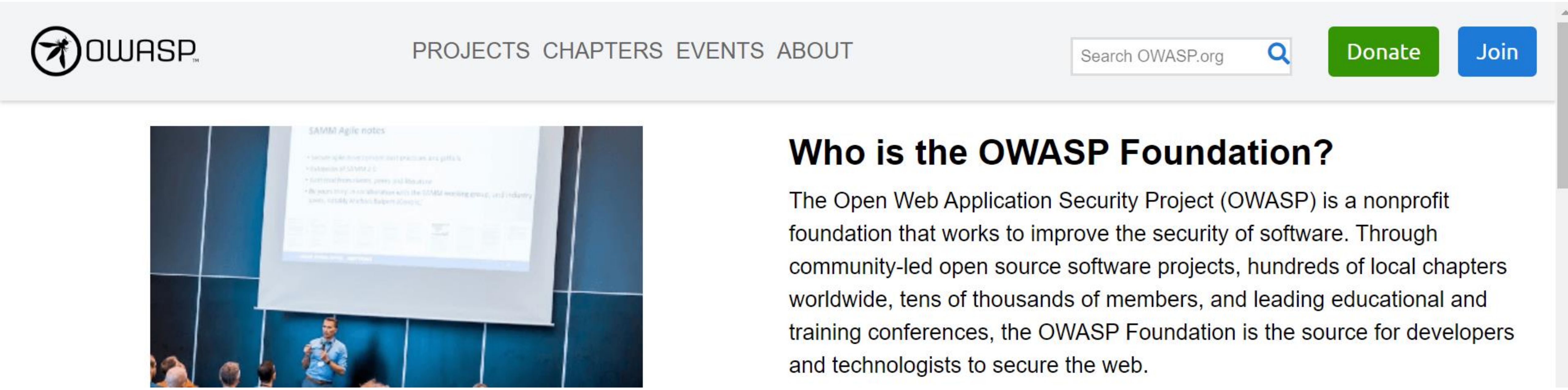
Source Error:

```
Line 26:
Line 27:        conn = New System.Data.SqlClient.SqlConnection(strCo
Line 28:        conn.Open()
Line 29:
```

Data Exposure Mitigation

- Make sure to turn on the Custom Error Pages option
- Always handle exception before returning to the front end

Meet OWASP

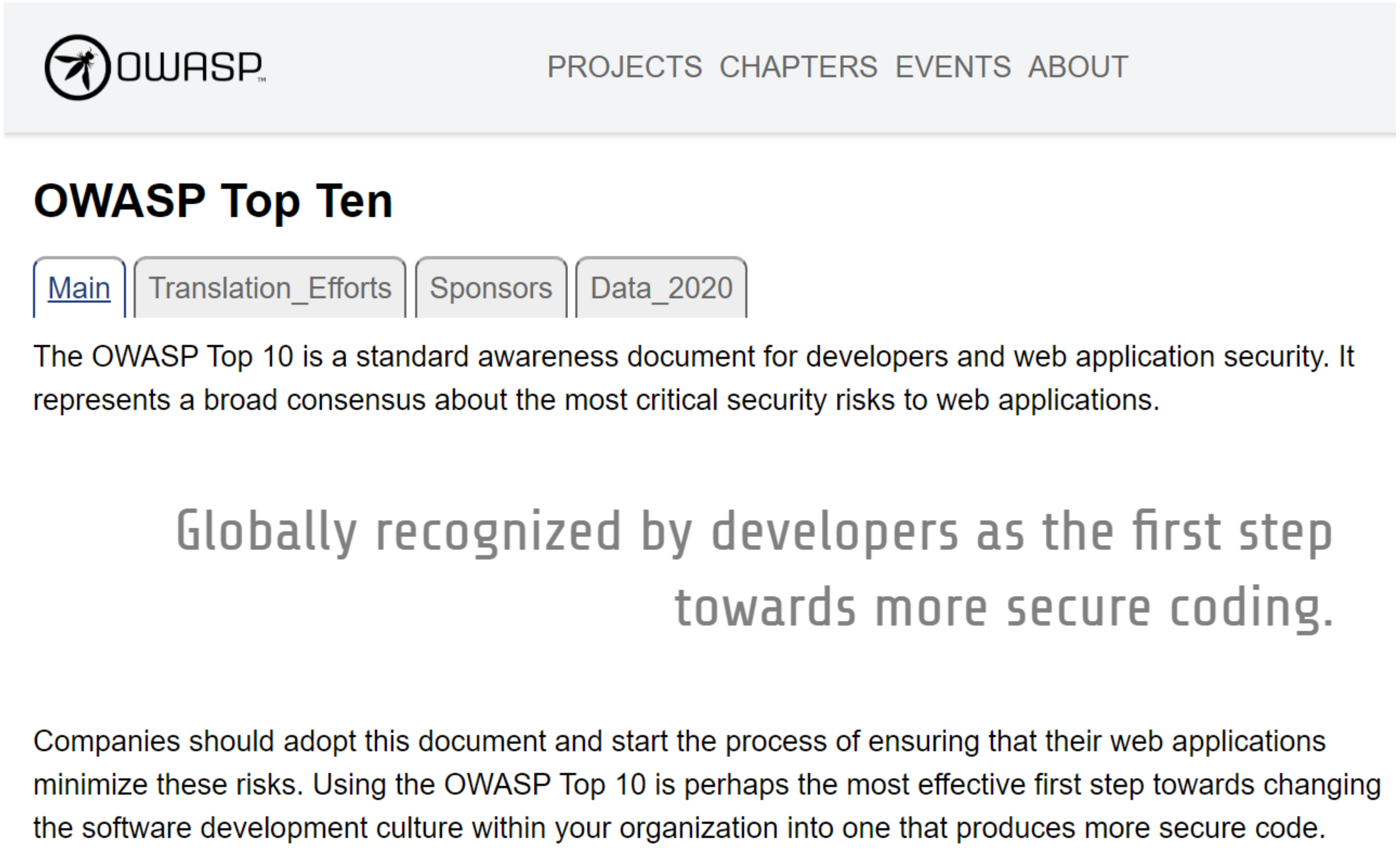


Who is the OWASP Foundation?

The Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

https://owasp.org/

OWASP Top Ten



OWASP Top Ten

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

Globally recognized by developers as the first step towards more secure coding.

Companies should adopt this document and start the process of ensuring that their web applications minimize these risks. Using the OWASP Top 10 is perhaps the most effective first step towards changing the software development culture within your organization into one that produces more secure code.

https://owasp.org/www-project-top-ten/

Secure Code & The Architect

- Architecture should mention the Secure Code techniques
- Make sure dev team is aware of them
- Check if training is necessary, if so – make it happen

Secure Data

What It Is

- Making sure the data is protected as much as possible
- Usually geared towards data in databases
  - Which is what we’ll discuss

What It’s For

- Data is the heart of the system
- This is what hackers are after
- Data Security protects against:
    - Data leak
    - Data loss
    - Data inconsistency

Secure Data

- Two main methods:
  - Make accessing the data as hard as possible
  - If data is stolen – make it as hard as possible to read it

Secure Data

Hard to access

- Have full-blown authorization in place
- Employ the least-privilege principle

Hard to read

Encrypt sensitive data

Encryption

Using built-in DB capabilities

Self developed

Built-In Database Encryption

- Using the database's encryption capabilities

mongoDB® Microsoft® SQL Server® PostgreSQL ORACLE®

- The preferred way to go
- Robust
- Secure
- (Almost) no code changes

Self Developed Encryption

- Using the platform's encryption libraries

Crypto

System.Security.
Cryptography

Javax.crypto.
Cipher

- Less preferred
- Tedious coding
- Not flexible

Key Management

- Encryption uses keys
- Usually (with DB encryption) there are column keys and master key
- The master key MUST be kept in a secure key store

Key Stores

- Securely store keys, certificates and more
- Examples:
  - Windows Certificate Store
  - Azure KeyVault
  - Java KeyStore

Key Stores

- In most cases, the key store is dictated by the database
- When developing your own encryption make sure you use a key store
- DO NOT store the key in your code or config file
- DO NOT develop your own key store

Secure Data & The Architect

- Map the data that should be encrypted
- Decide on the encryption strategy
  - Always prefer database built-in encryption
  - Consult with the DBA

Secure Data & The Architect

- Make sure there is a secure key store in the organization
- If not – consult with the IT team about creating one
- If no store can be created in the organization consider using a cloud-based one (ie. Azure KeyVault)

Logging & Monitoring

What It Is

- Making sure we know what’s going on with our system
- Get notifications when something suspicious happens
- Collect data for future analysis of the system’s behavior

What It's For

- Great way for detecting attacks of all kinds
- Provides wholistic view on the system
- Warns against:
    - Data leak
    - Data loss
    - Data inconsistency
    - Disruption of service

Logging

- Should be part of the existing logging mechanism
  - Do not implement separate security logging
- Log everything that might be security related
  - Last log-in time of users
  - Users' activity
  - Validation problems

Logging Cont.

- Log metrics to provide aggregated data
  - No. of requests
  - No. of logins
  - No. of errors

Monitoring

- Use existing monitoring solution
- Define alerts for suspicious security-related activities
  - More than X validation problems in minute
  - More than 1,000 requests in 10 seconds
  - Log-in from an unknown location

Logging & Monitoring and The Architect

- Make sure there are logging and monitoring mechanisms in place
- Define the security events that should be logged
- Define the alerts that should be raised