Web Application Vulnerabilities and Security Flaws Root Causes: The OWASP Top 10

Cincinnati Chapter Meeting
May 26th, 2009

Marco Morana
Cincinnati Chapter Lead

OWASP

Agenda

- The OWASP T10
- Tactical approaches to the OWASP T10
- Mapping OWASP T10 to Web Application Architecture
- Threat Modeling A Sample Web Application
  - Threats, Vulnerabilities and Countermeasures
  - OWASP T10 Security Flaws Root Causes
- Strategic approaches to the OWASP T10
- Security By Design
  - Principles
  - Guidelines
- Appendix
  - Application Threat Modeling Methodology

The OWASP Top 10

- The Ten Most Critical Issues
- Aimed to educate developers, architects and security practitioners about the consequences of the most common web application security vulnerabilities
- Living document: 2007 T10 different from 2004 T10
- Not a silver bullet for software security
- A great start, but not a standard “per se”

Tactical Approach to the OWASP T10

- From the hunting security issue perspective by looking at symptoms, causes and risk factors
  - **The symptoms** are the insecure observed behavior of the application against potential vulnerabilities and exploits
  - **The root causes** are security design flaws, security bugs (coding errors), insecure-configuration
  - **The risk factors** are the quantifiable risks such as how much damage can be done, how easy is to reproduce the exploits, how many users are exposed and how easy is to discover the vulnerabilities

Mapping OWASP T10 to Security Flaws

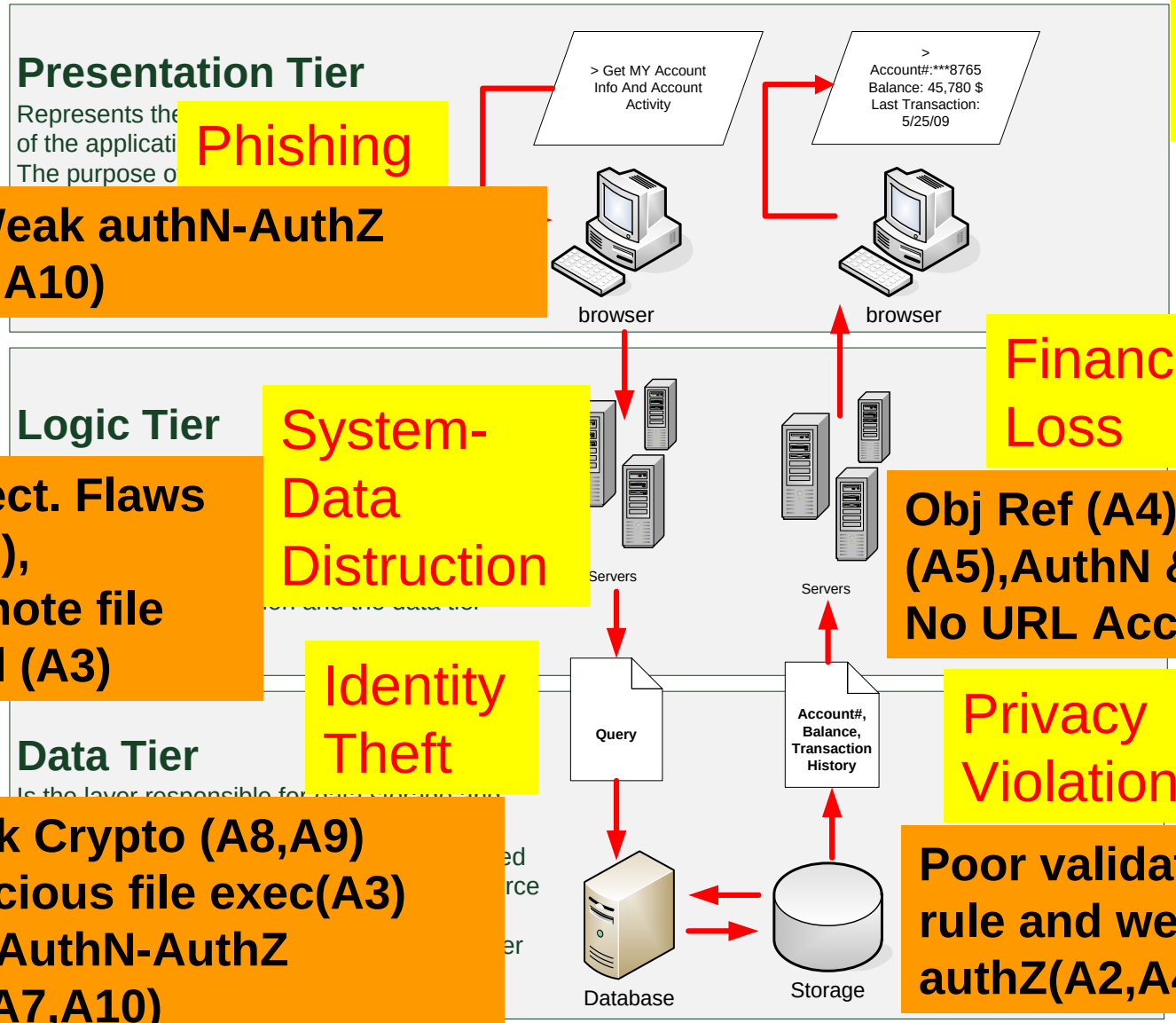
SUMMARY

A1 – Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, etc.
A2 – Injection Flaws	Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
A3 – Insecure Remote File Include	Code vulnerable to remote file inclusion allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise.
A4 – Insecure Direct Object Reference	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.
A5 – Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker.
A6 – Information Leakage and Improper Error Handling	Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to violate privacy, or conduct further attacks.
A7 – Broken Authentication and Session Management	Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.
A8 – Insecure Cryptographic Storage	Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.
A9 – Insecure Communications	Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.
A10 – Failure to Restrict URL Access	Frequently, the only protection for sensitive areas of an application is links or URLs are not presented to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations.

OWASP T10 Mitigated Risks

- **Phishing**
  - Exploit weak authorization/authentication, session management and input validation (XSS, XFS) vulnerabilities
- **Privacy violations**
  - Exploit poor input validation, business rule and weak authorization, injection flaws
- **Identity theft**
  - Exploit poor or non-existent cryptographic controls, malicious file execution, authentication, business rule and auth checks vulnerabilities
- **System compromise, data destruction**
  - Exploit injection flaws, remote file inclusion-upload vulnerabilities
- **Financial loss**
  - Exploit unauthorized transactions and CSRF attacks, broken authentication and session management, insecure object reference, weak authorization-forceful browsing vulnerabilities
- **Reputation loss**
  - Any public evidence of a vulnerability

Mapping OWASP T10 To Web Architecture



Phishing

XSS, Weak authN-AuthZ (A4,A7,A10)

Reputation Loss

All T10

Financial Loss

System-Data Distruction

Inject. Flaws (A2), remote file incl (A3)

Obj Ref (A4), CSRF (A5),AuthN & SessM (A7), No URL Access Rest (A10)

Identity Theft

Privacy Violations

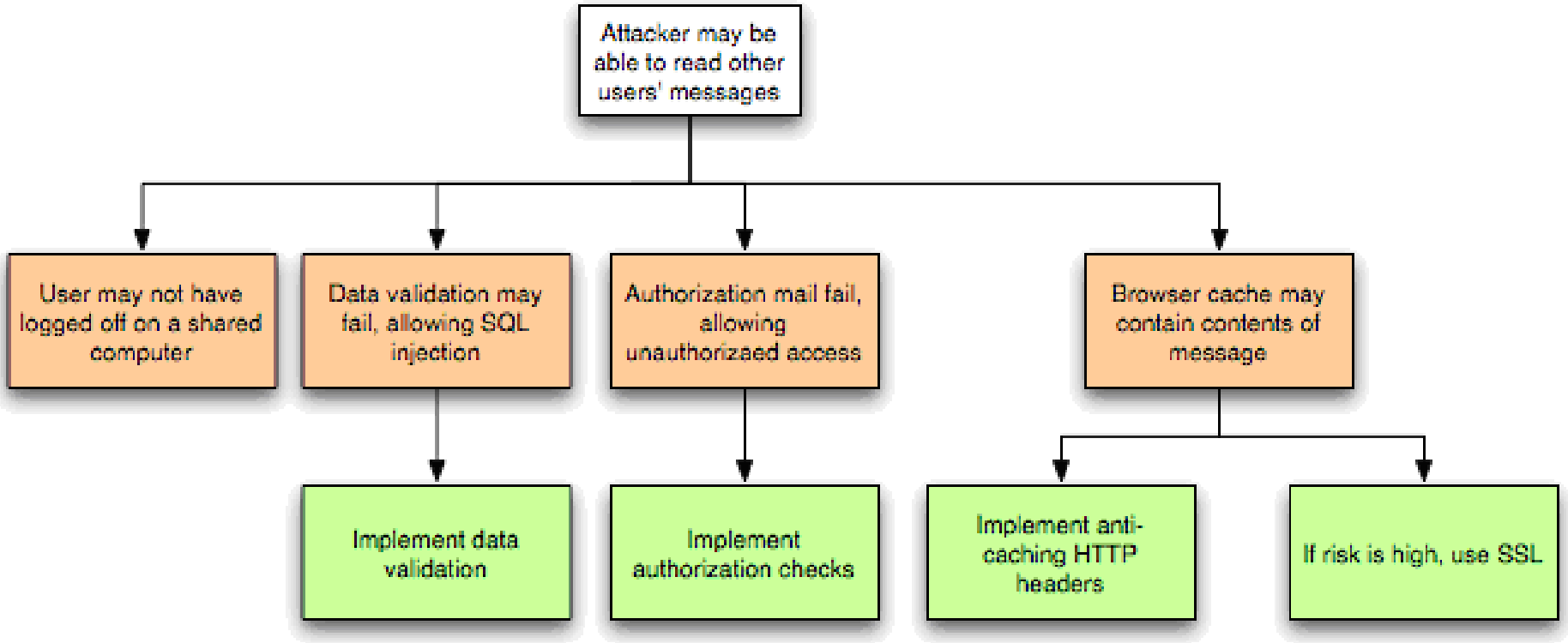
Weak Crypto (A8,A9) malicious file exec(A3) and AuthN-AuthZ (A4,A7,A10)

Poor validation, business rule and weak authZ(A2,A4,A6,A7,A10)

What is more actionable than a checklist?

- ***Threat Modeling***
  - ***A systematic & strategic approach for enumerating threats to an application environment, with the objective of minimizing risk and associated impact levels to the business***
- Different models to categorize threats and vulnerabilities and identify countermeasures
- Different artifacts can be used for the threat analysis:
  - Threat Trees
  - Use-Misuse Cases
  - Data-Flow Diagrams

Mapping Threats, Vulnerability Conditions and Countermeasures Using Threat Trees

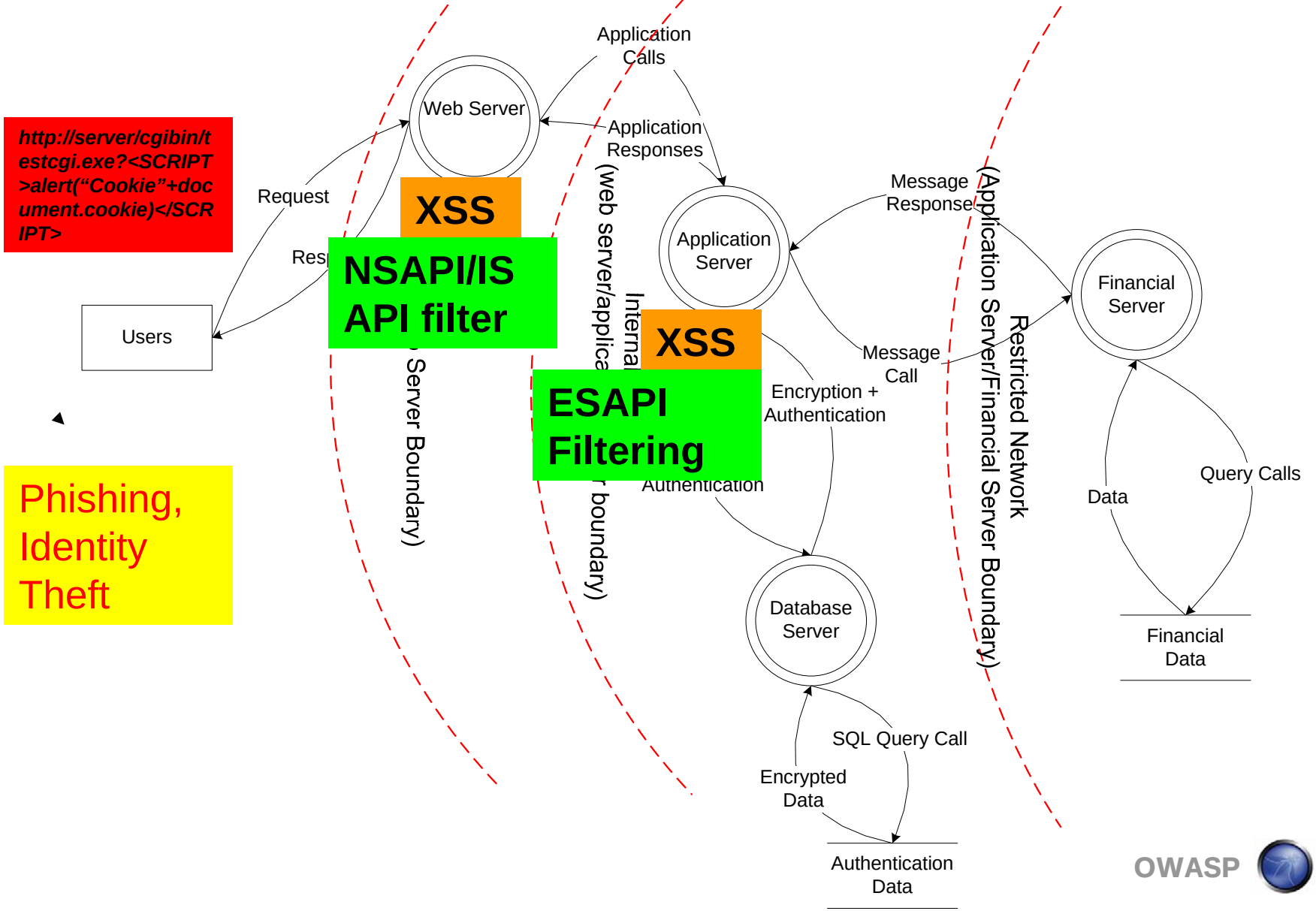


Source: OWASP Application Threat Modeling, https://www.owasp.org/index.php/Application_Threat_Modeling

A1: Cross Site Scripting

- **Threats**
  - Attacker crafts a URL by attaching to it a malicious script that is sent via phishing or posted as a link on a malicious site. The malicious script executes on the user victim browser
- **Vulnerabilities**
  - Lack of filtering and encoding of malicious script carried by XSS attack vectors
- **Countermeasures**
  - Filtering should be in place at the different layers of the architecture client, web server and web application

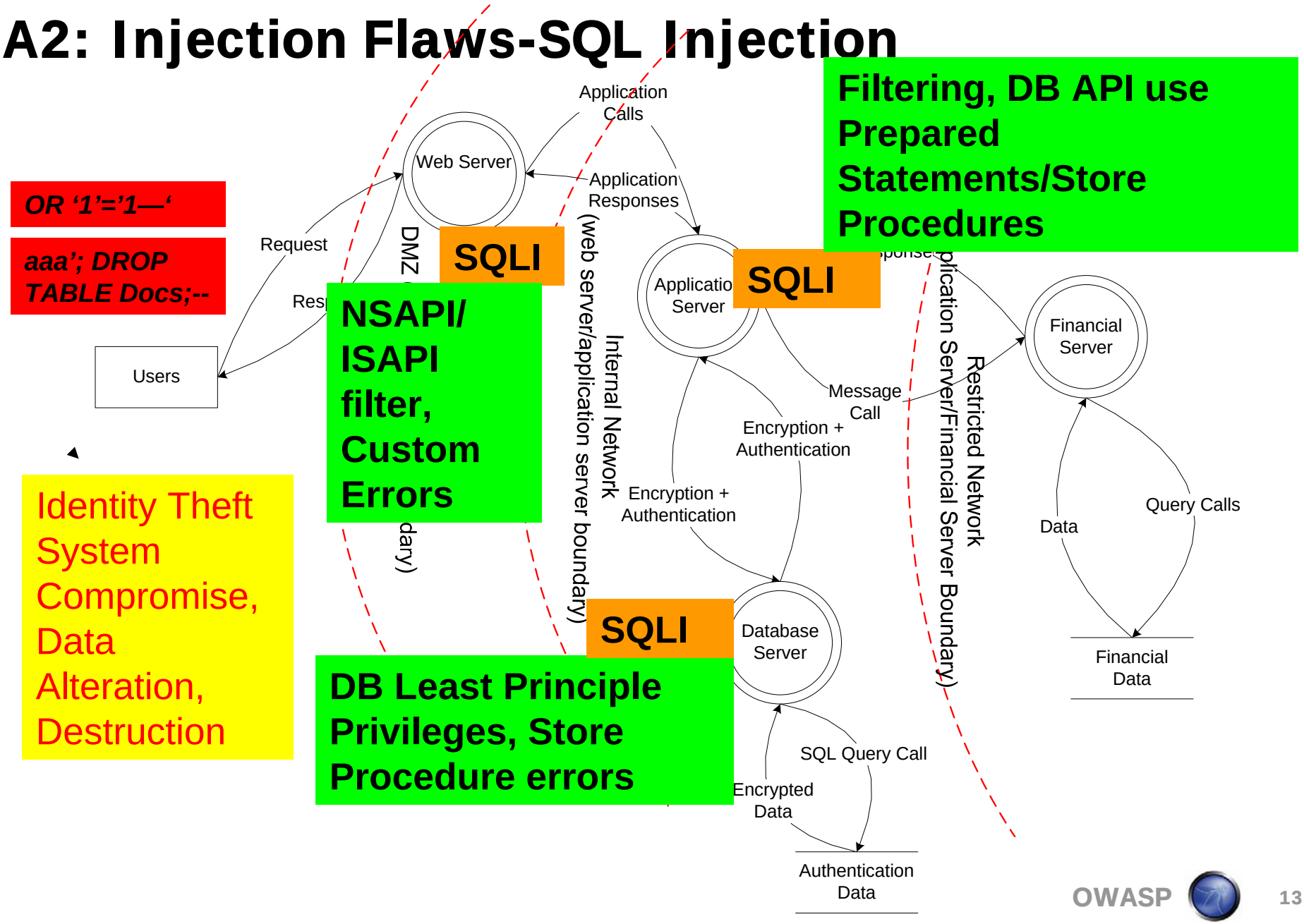
A1: Cross Site Scripting – Security Flaws



A2: Injection Flaws –SQL Injection

- **Threats**
  - Malicious user constructs an input containing malicious SQL query, supplies it to the application and the application executes it. The query can be used for information disclosure, elevation of privileges, breaking of authentication, disruption of data and denial of service
- **Vulnerabilities**
  - Unfiltered input and dynamic SQL query construction, non enforcement of minimum privileges
- **Countermeasures**
  - Filtering input, use of parameterized queries-store procedures/prepared statements, limits of DB privileges, custom errors

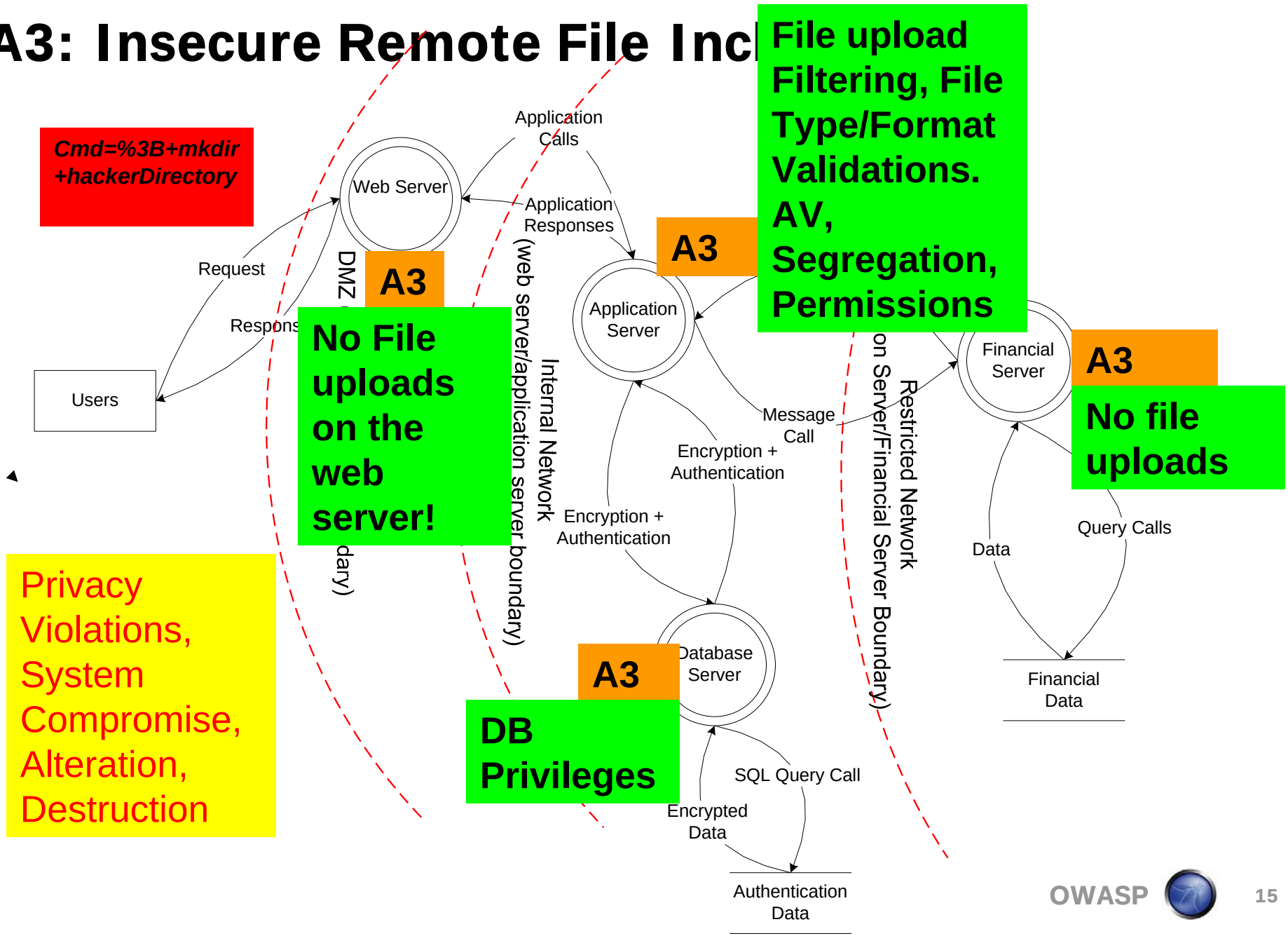
A2: Injection Flaws-SQL Injection



A3: Insecure Remote File Include

- **Threats**
  - Malicious user manipulate parameters to run commands in the application context by the operating system or to upload malicious files (e.g. script) that can be executed on the application server
- **Vulnerabilities**
  - Lack of parameters validations, lack of authentication and enforcement of minimum privileges and lack
- **Countermeasures**
  - Do not rely on user inputs/ use hash-tables, white-list filter/escape commands, validate file type-format, run AV on uploaded files, segregate uploads

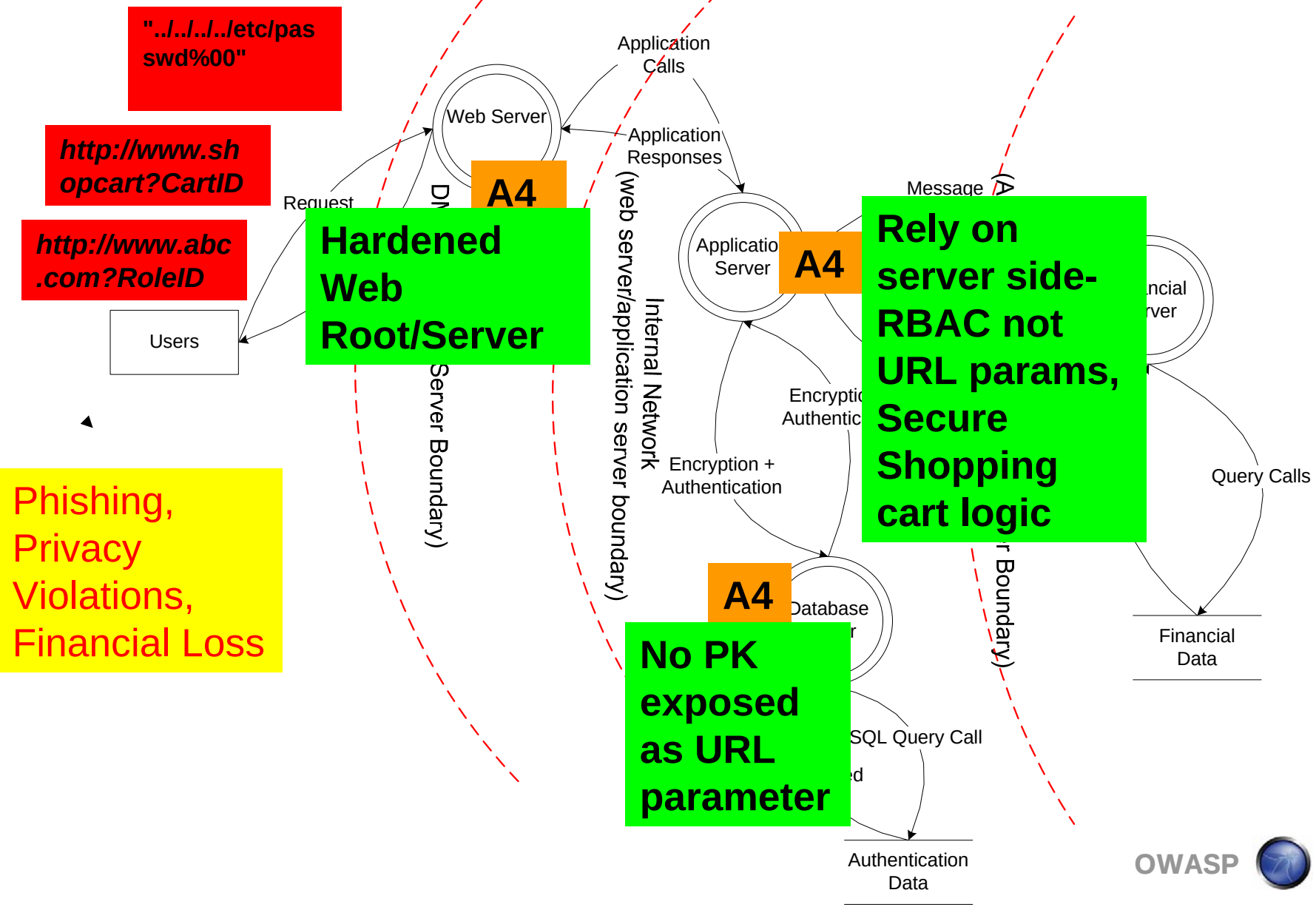
A3: Insecure Remote File Incl



A4: Insecure Direct Object Reference

- **Threats**
  - An attacker can manipulate direct object references to access other objects without authorization, unless an access control check is in place.
- **Vulnerabilities**
  - Invalidated reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter
- **Countermeasures**
  - Enforce server side authorization controls, only expose direct object references to authorized users, do not expose references to primary keys and filenames, enforce strict business logic/rules depending on server side parameters

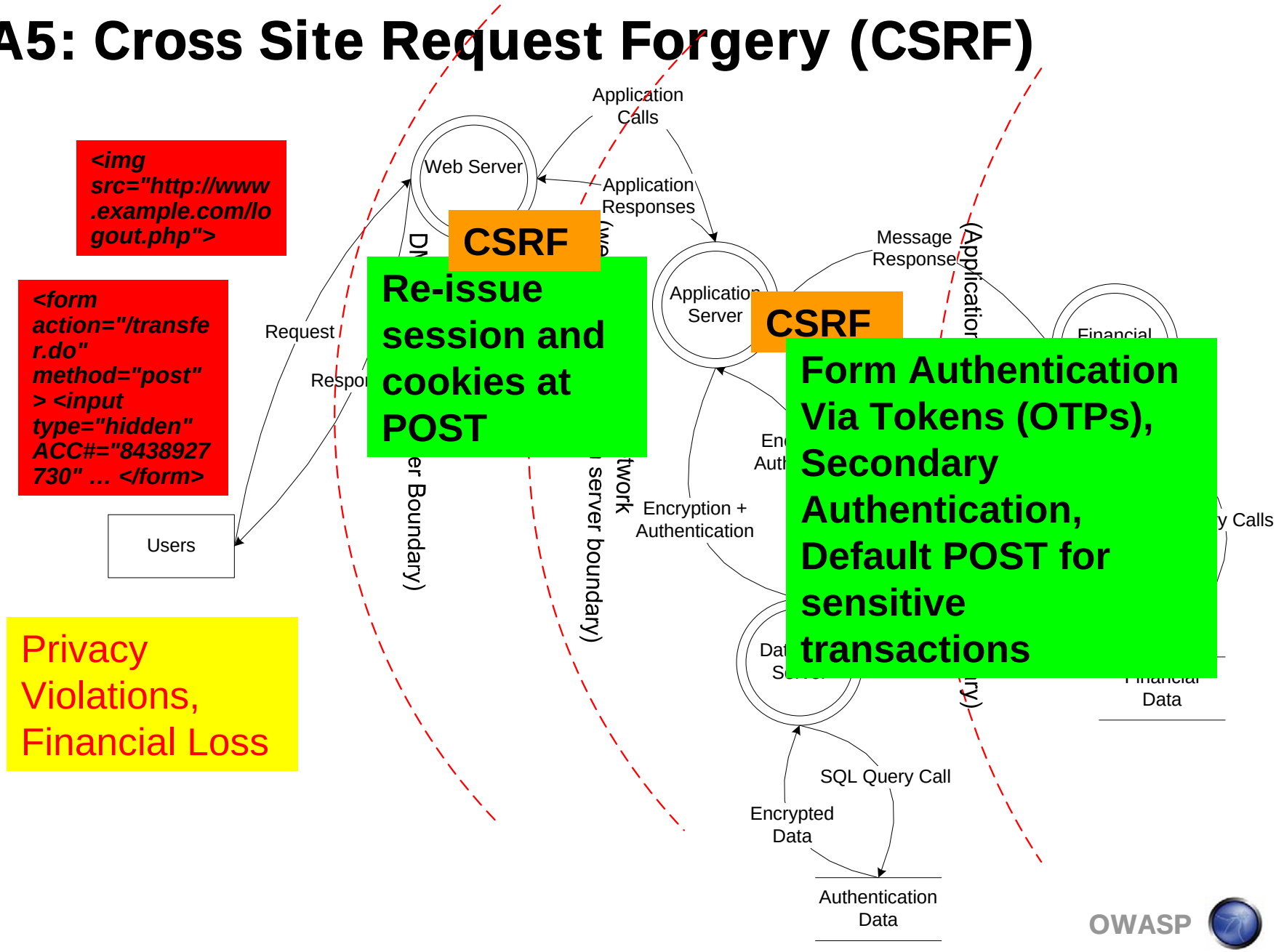
A4: Insecure Direct Object Reference



A5: Cross Site Request Forgery

- **Threats**
  - May direct the user to invoke logouts and steal user credentials. In a bank application might invoke processing requests such as transfer of funds.
- **Vulnerabilities**
  - A CSRF attack forces a logged-on victim's browser to send a request to a vulnerable web application, which then performs the chosen action on behalf of the victim. Any web application without a build in CSRF control is vulnerable
- **Countermeasures**
  - Validate each HTTP request with one time use token, leverage struts-tokens, ESAPI, ViewStateUserKey (.NET), re-authenticate when performing high risk transactions, enforce POST only for forms with sensitive data

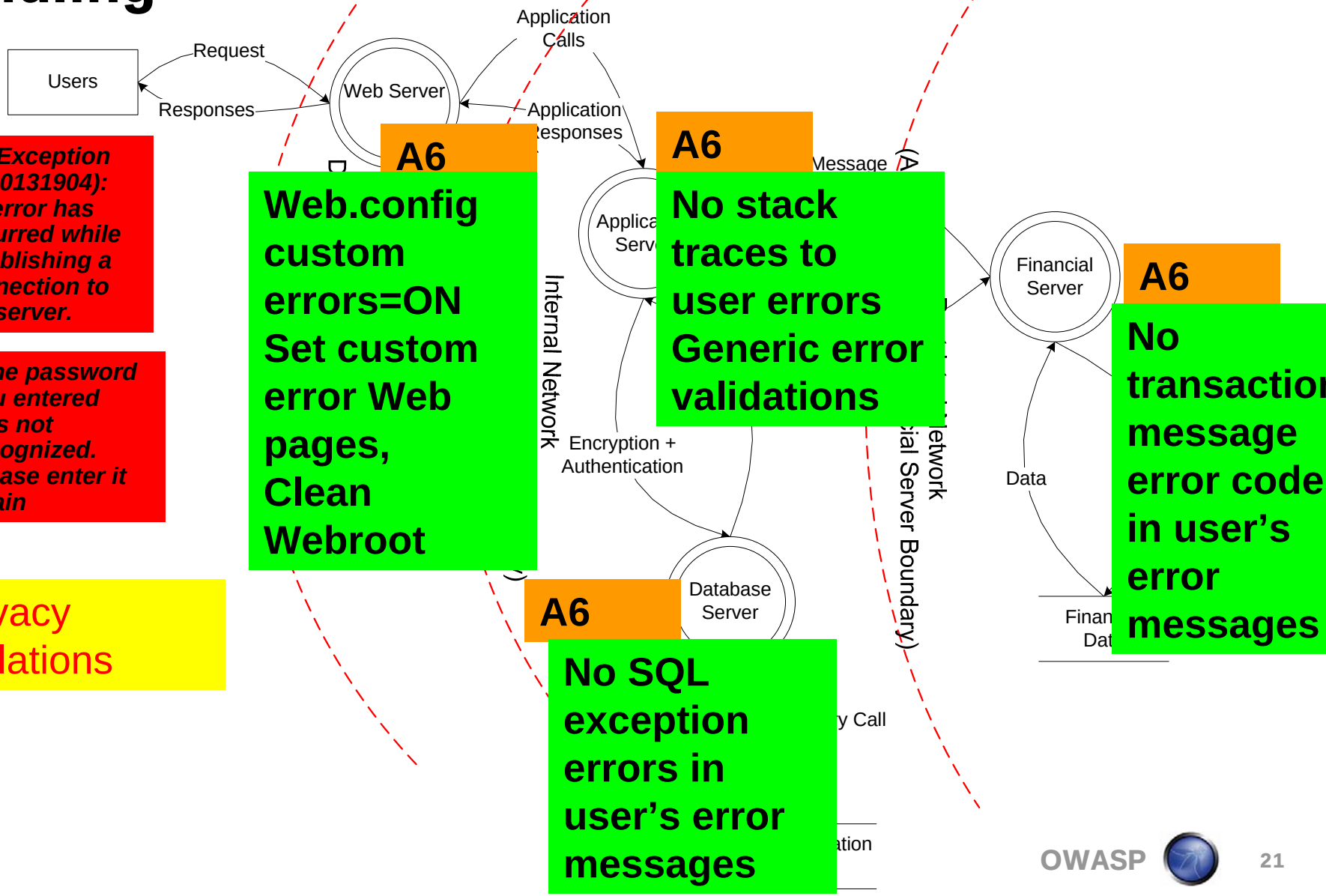
A5: Cross Site Request Forgery (CSRF)



A6: Information Leakage and Improper Error Handling

- **Threats**
  - A malicious user can gather information to attack the application from too informative error message (e.g. stack traces), enumerate user accounts and harvest credentials
- **Vulnerabilities**
  - Leaking of information in during exception handling and error reporting. Lack of generic error messages upon validation of credentials
- **Countermeasures**
  - Provide generic custom error messages during error validations, do not display stack traces with exception messages

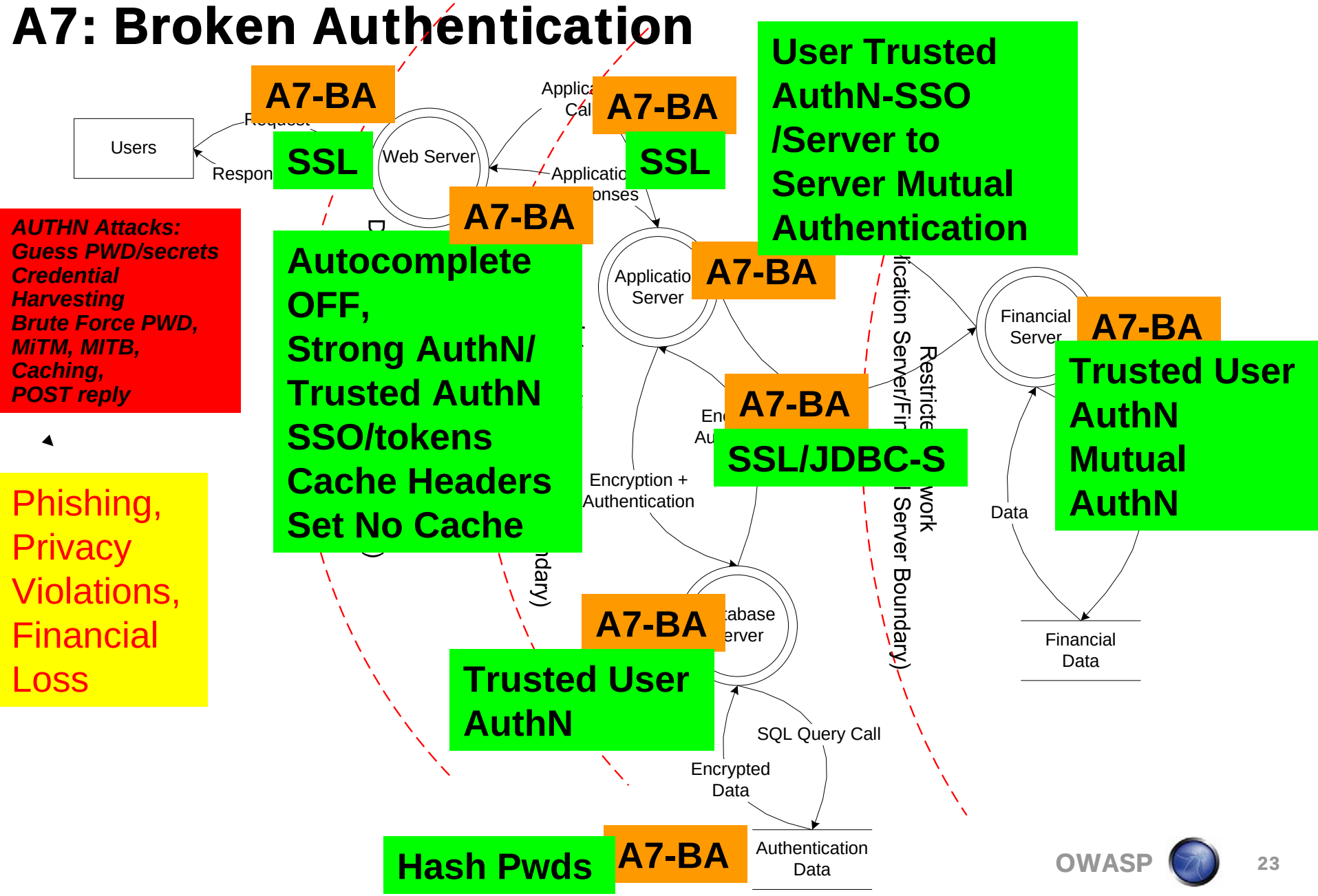
A6: Information Leakage and Improper Error Handling



A7: Broken Authentication (BA) and Session Management (SM)

- **BA Threats**
  - Flaws can lead to the spoofing of the credentials in transit, man in the middle attacks, brute forcing of password and guessing of passwords
- **BA Vulnerabilities**
  - Weak or no authentication as well in-secure password management (e.g. resets), password stored in clear in cookies, login caching, credential harvesting via error messages and social engineering, lack of mutual authentication, lack of protection of credentials in storage and transit, use of reversible encryption for passwords, use of impersonation instead of trusted authentication
- **BA Countermeasures**
  - Use authentication strength commensurate to the risk of the transaction, protect authentication credentials with encryption (e.g. SSL), use challenge/responses in password resets, userID resets, use trusted authentication (e.g. SSO) not impersonation, do not cache login-pages, enforce idle time-out

A7: Broken Authentication



A7:Broken Authentication and Session Management

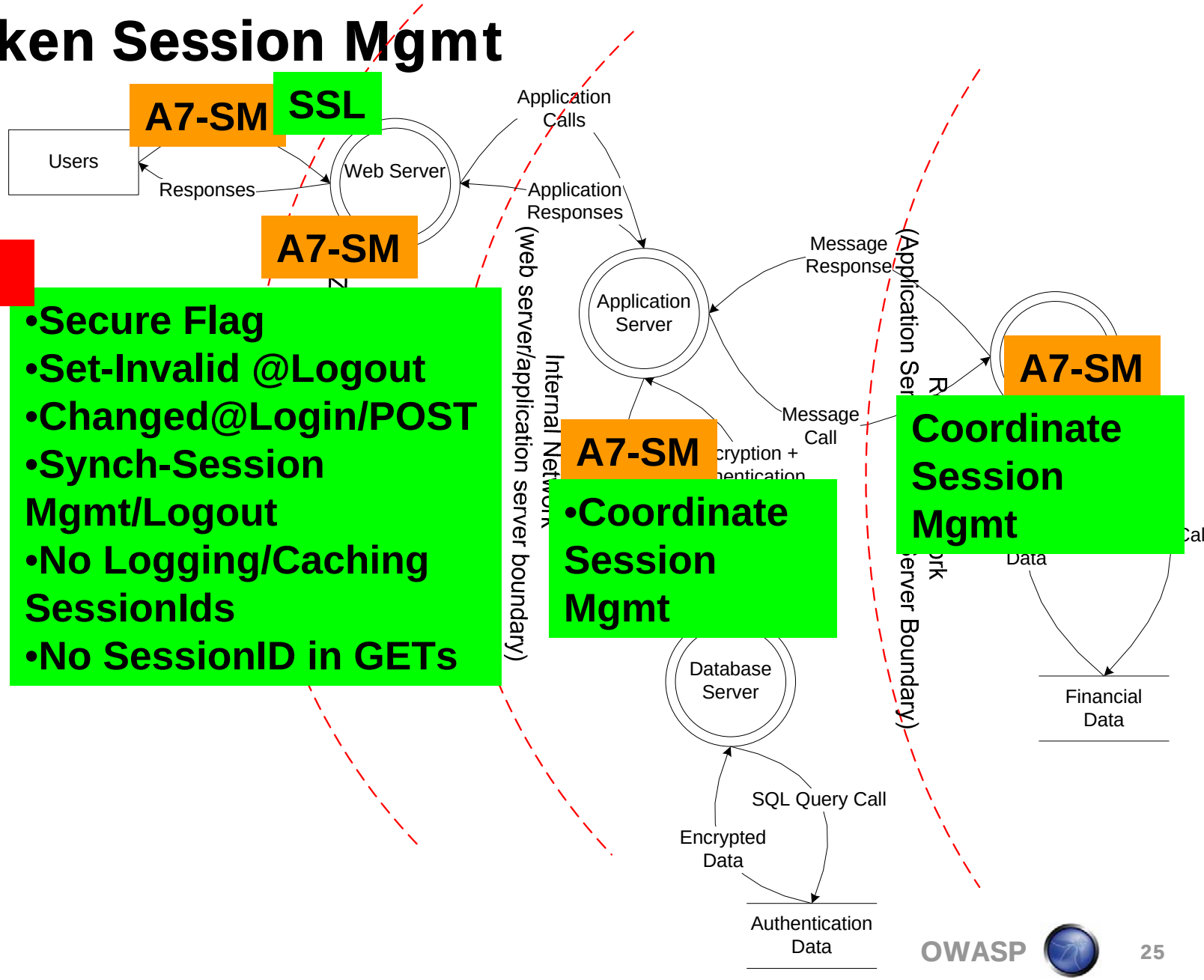
- **SM Threats**
  - Session management flaws can lead to hijacking of user accounts, user impersonation, undermine authorization and accountability controls and cause privacy violations.
- **SM Vulnerabilities**
  - Failure to protect credentials and session tokens through their lifecycle by non re-issuing after authentication and POST data, non marking secure, SessionID caching and logging, guessable SessionIDs, lack of SessionID invalidation at logout, lack of synchronized session management and single logout
- **SM Countermeasures**
  - Re-issue SessionIDs after each POST, protect SessionID from caching and logging, set secure flag, uses syncronized session management across servers, limit sessions per user, invalidate them at user/idle logouts

A7: Broken Session Mgmt

HTTP Redirection
Session Hijacking

Phishing, Privacy Violations, Financial Loss

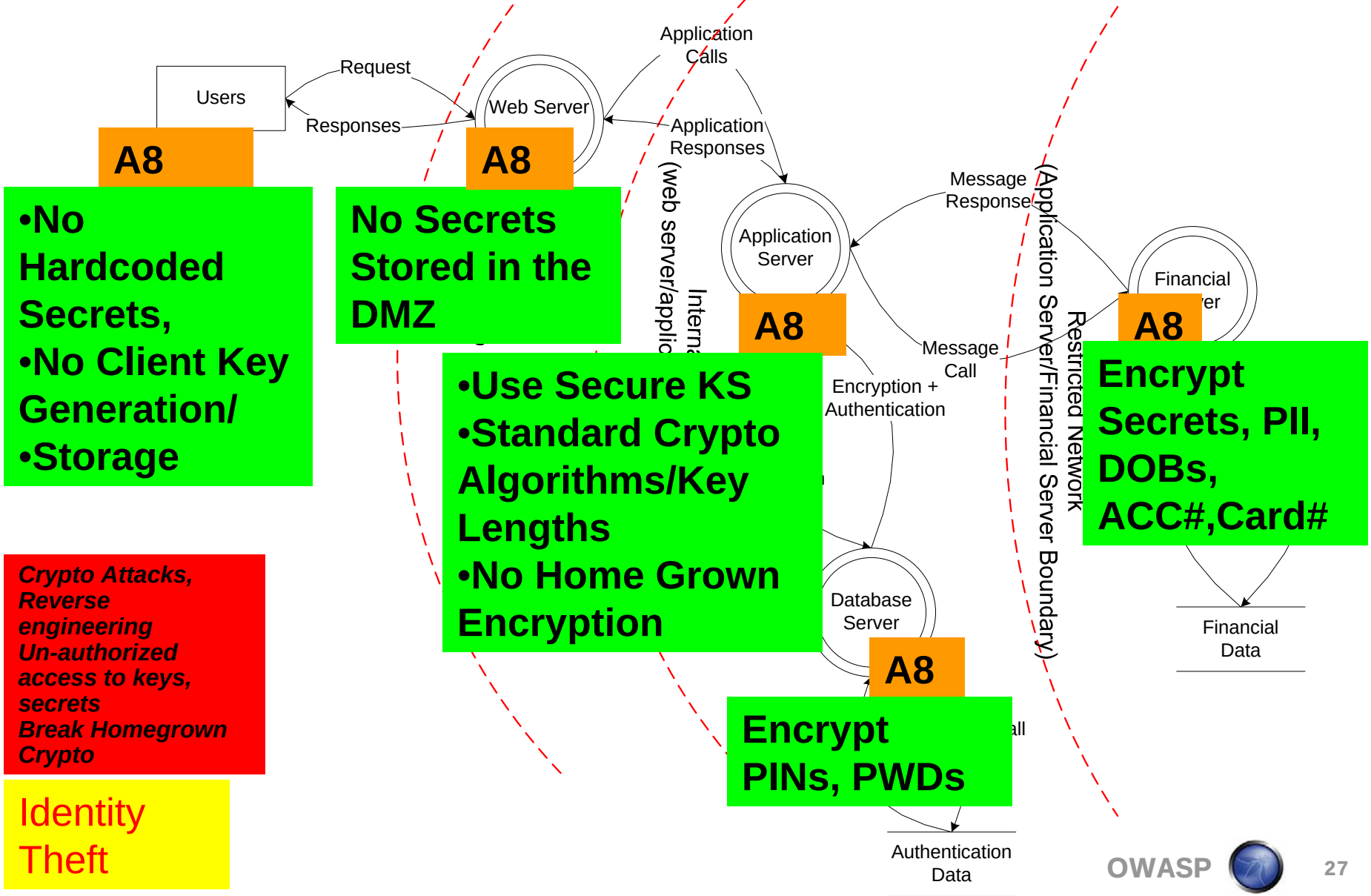
- **Secure Flag**
- **Set-Invalid @Logout**
- **Changed@Login/POST**
- **Synch-Session Mgmt/Logout**
- **No Logging/Caching SessionIds**
- **No SessionID in GETs**



A8: Insecure Cryptographic Storage

- **Threats**
  - Disclosure of customer sensitive information, authentication data to unauthorized users, secrets such as keys and challenge response answers
- **Vulnerabilities**
  - Failing to protecting sensitive data with cryptography, use of weak encryption algorithms/short encryption keys, use of home-grown encryption, unsecured secrets such as private keys via hard-coding
- **Countermeasures**
  - Encrypt authentication credentials and sensitive information in storage, use standard encryption strength algorithms and minimum key lengths, use secure keys storage and secure management

A8: Insecure Cryptographic Storage



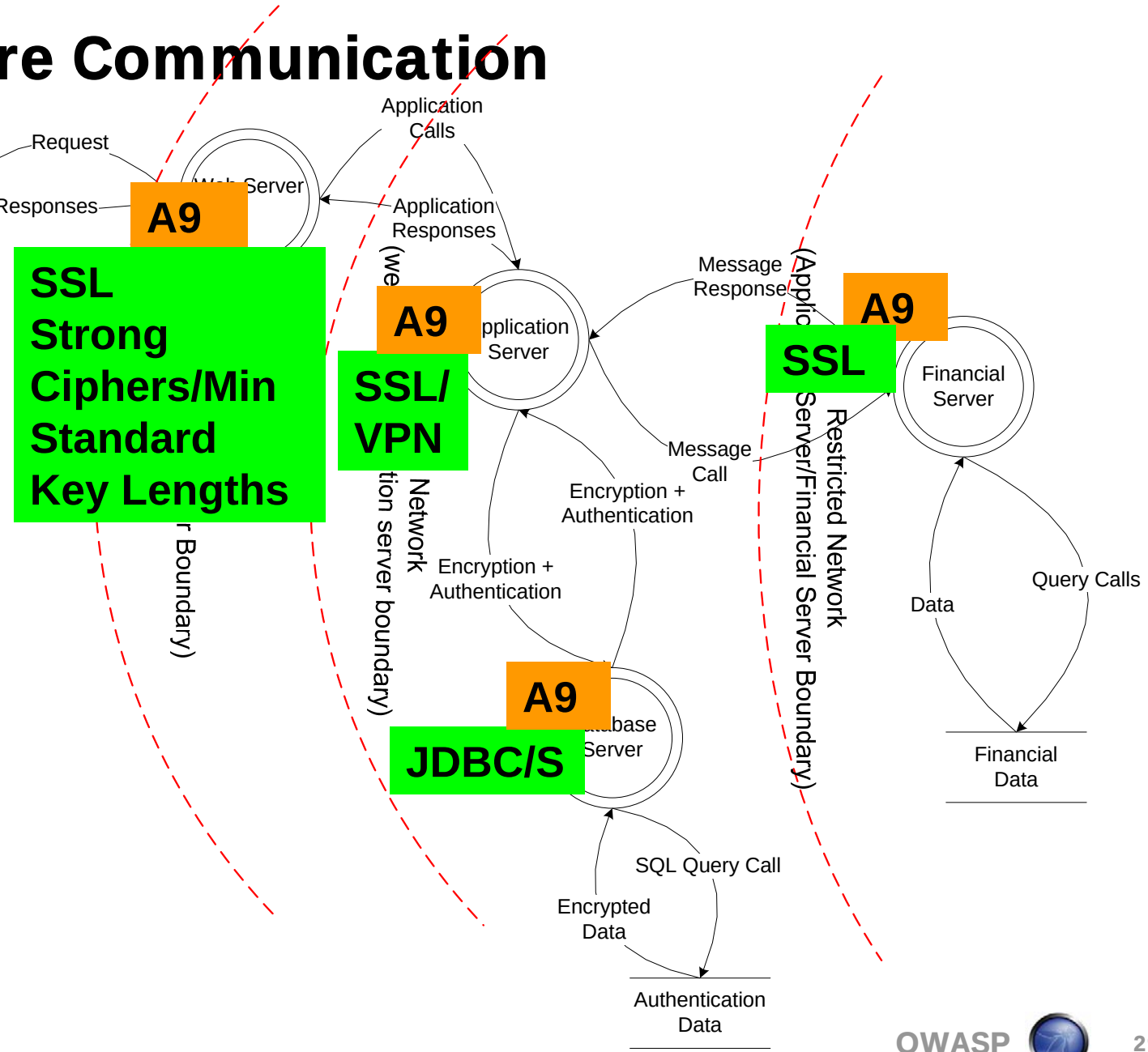
A9: Insecure Communication

- **Threats**
  - Loss of customer's PII and sensitive data for identity theft and financial fraud, non-compliance with standards, fees and law-suits
- **Vulnerabilities**
  - Failure to encrypt network traffic (e.g. SSL), failure to protect sensitive PII, sensitive and restrict data, weaknesses in SSL configuration
- **Countermeasures**
  - Use encrypted channel (e.g. SSL, VPN) for authenticated connections and while transmitting credentials, sensitive information (e.g ACC#) PII (e.g. SSN, DOB), health and other private information

A9: Insecure Communication

Crypto Attacks Against Weak Cypto/Keys Unsecure Certs, Spoofing/Sniffing clear sensitive data in transit

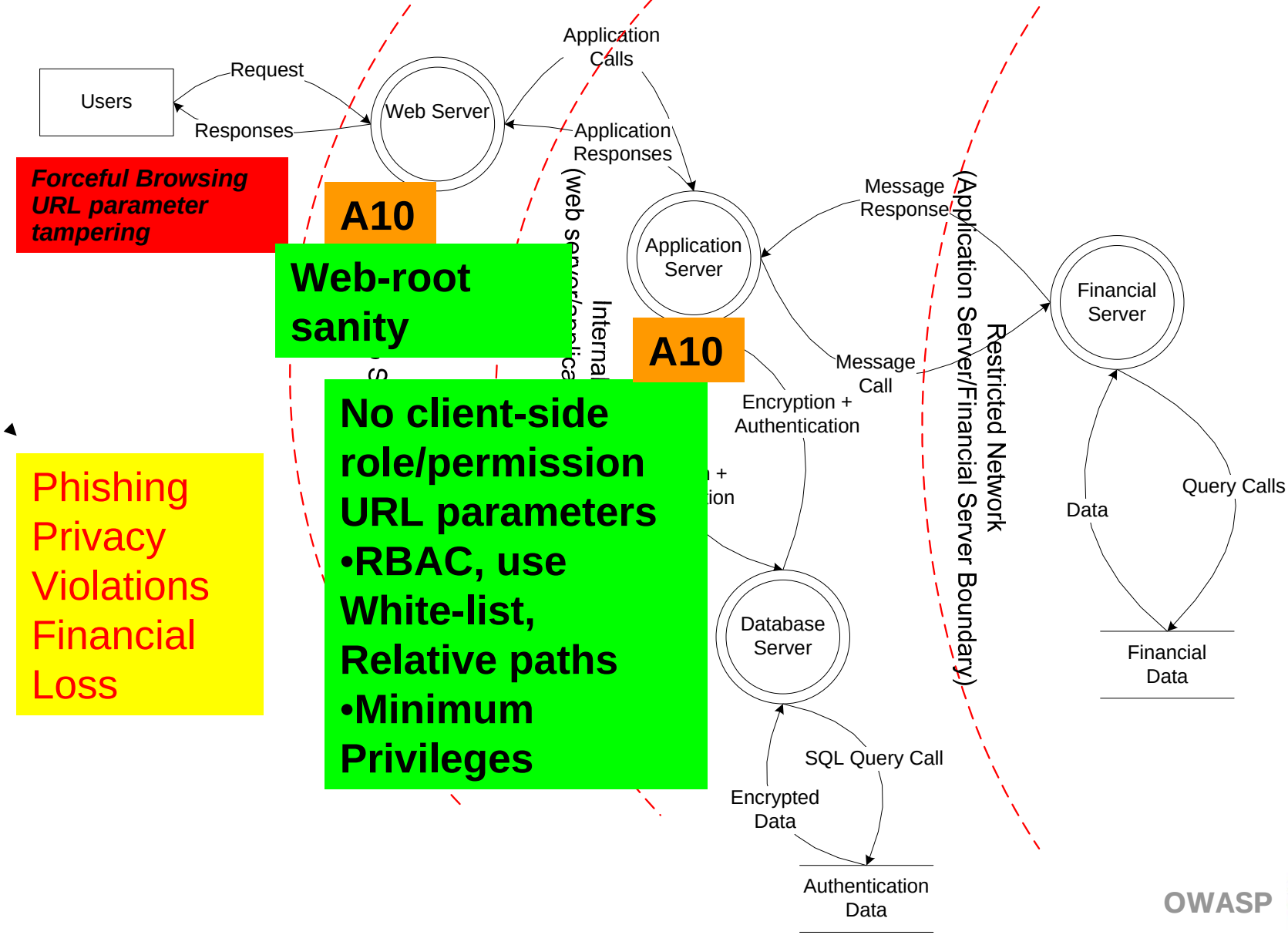
Identity Theft, Financial Loss



A10: Failure to restrict URL access

- **Threats**
  - An attacker may be able to access web pages with no authentication or bypassing authorization controls to access un-authorized functions and data
- **Vulnerabilities**
  - Lack of enforcement of authorizations for URL web page access
- **Countermeasures**
  - Ensure that RBAC is enforced on the server side to enforce which user has access to which web page, no HIDDEN parameters to enforce which web pages are accessible, only allow file types that you intend to serve, block any attempts to access log files, xml files, etc.

A10: Failure to restrict URL access



Strategic Approach to the OWASP T10

- **Security Engineering Processes** such as CLASP, MS SDL, TP that integrate activities such as threat modeling, secure code reviews, web application pen tests
- **Software Security Initiatives** such as BSIMM and SAMM that allow for planning of software security within the organization domains
- **Training and Awareness** in secure coding, teams work toward certifications (e.g. IC2 CSSLP, SANS GSSP) and participate to local OWASP chapter meetings
- **Software and Application Security Tools**
  - Source Code Analysis Tools (e.g. Fortify SCA)
  - Web Penetration Testing Tools (e.g. IBM Appscan)
  - Modelling Tools (e.g. VISIO, TM)
- **Software Security Standards and Guidelines**
  - Secure Coding Standards/Guides (e.g. OWASP, SafeCode, SANS T25)
  - Secure Architecture Design Patterns & Practices (e.g. MSDN, OWASP)
  - Security Testing Guides (e.g. OWASP)

Security By Design

- Objective is promote secure design and identify of potential flaws before construction phase
- Secure Architecture Design Review Process
  - Review high level design documents and verify that security controls requirements are documented
  - Provide guidance on security technology as well as secure design principles
  - Identify potential gaps of security controls in the architecture
  - Engage secure design/assessment sessions with:
    - Architects
    - Project Managers
    - Application Security Experts

The Scope Of Security By Design

- Securing The Network
  - Access Controls: Default Deny vs. Default Permit
  - Auditing and Logging (A&L)
  - IDS/IPS, Firewall/ACLs
  - Patching
- Securing The Web server
  - Hardening and Locking
  - Secure Configuration Management
  - Auditing and Logging
- Securing The Application Server
  - Mutual SSL Authentication, WS-Security, Secure XML, Secure Session Management, Auditing & Logging
- Securing The Database
  - Hardening, Extended Store Procedures, Access Privileges,

S-ODBC, Data Protection in Storage: Hashing/Encryption, A&L

Secure Architecture Requirements

- General Security Design Principles
  1. Implement Authentication With Adequate Strength
  2. Enforce Least Privilege
  3. Protect Sensitive Data In Storage, Transit And Display
  4. Enforce Minimal Trust
  5. Trace and Log User Actions And Security Events
  6. Fail Securely And Gracefully
  7. Apply Defense in Depth
  8. Apply Security By Default
  9. Design For Simplicity, KISS Principle
  10. Secure By Design, Development and Deployment
  11. Secure As The Weakest Link
  12. Avoid Security By Obscurity

Secure Design Guidelines: Authentication and Authorization

- Authentication, What, Where and How
  - Mandatory to restrict access to validated users
  - Strength depends on application risk/data classification
  - Compliant with regulations/standards
  - Provide for secure password and account management
  - Mitigates brute forcing and credentials harvesting
  - Mitigates Man In The Middle Attacks (MiTM)
  - Provides for user and host to host authentication
- Authorization Most Common Flaws
  - Flaws in Role Base Access Controls (RBAC)
  - Flaws allow for horizontal and vertical privilege escalation and forceful browsing

Secure Design Guidelines : Cryptography

- When is required and where
  - Storage and transit
  - Required for sensitive information (e.g. Credit Card No., Account No., SSN)
  - Required for authentication data
  - Session tokens should be considered sensitive
  - Key management aspects: key generation, distribution, storage, destruction and other relevant issues
- How should be implemented
  - Rely on standard solutions not home-grown
  - Reversible vs. irreversible encryption
  - Approved algorithms and minimum key lengths (e.g. 128 bit for symmetric and 1024 for asymmetric).

Secure Design Guidelines: Session Management

- Avoid common session management flaws:
  - Session cookies and authentication tokens unprotected (e.g. clear text) between client and server
  - Missing session invalidation at idle-time out and user logout
  - Missing re-issuance of new session token to prevent re-use for authentication
  - Un-secure storage in a session store in clear text
  - Lack of strong random generation of session cookies/identifiers (e.g. >128 bit)
  - Lack of coordinated session between application tiers

Secure Design Guidelines: Data Management And Validation

- What and where to validate
  - Type, format, range and length of input data
  - Wherever data crosses system or trust boundaries
  - Input and output
  - Client validation vs. server validation
- How to validate
  - Constraint, Reject, Sanitize
  - Canonical Validation
  - Output Encoding
  - Integrity Checks

Security Controls Design Guidelines: Secure Auditing And Logging, Error and Exception Handling

- Secure Auditing And Logging :
  - Provide application logs for traceability and non-repudiation for secure administrators and incident response
  - Do not log sensitive information (unless required by fraud)
  - Control access and integrity of logs
- Error And Exception Handling:
  - Avoid fail-open or insecure state
  - Avoid disclosure of customer or application sensitive information in errors
  - Avoid specific validation errors during credential validation change and recovery

Secure Architecture an Design Patterns

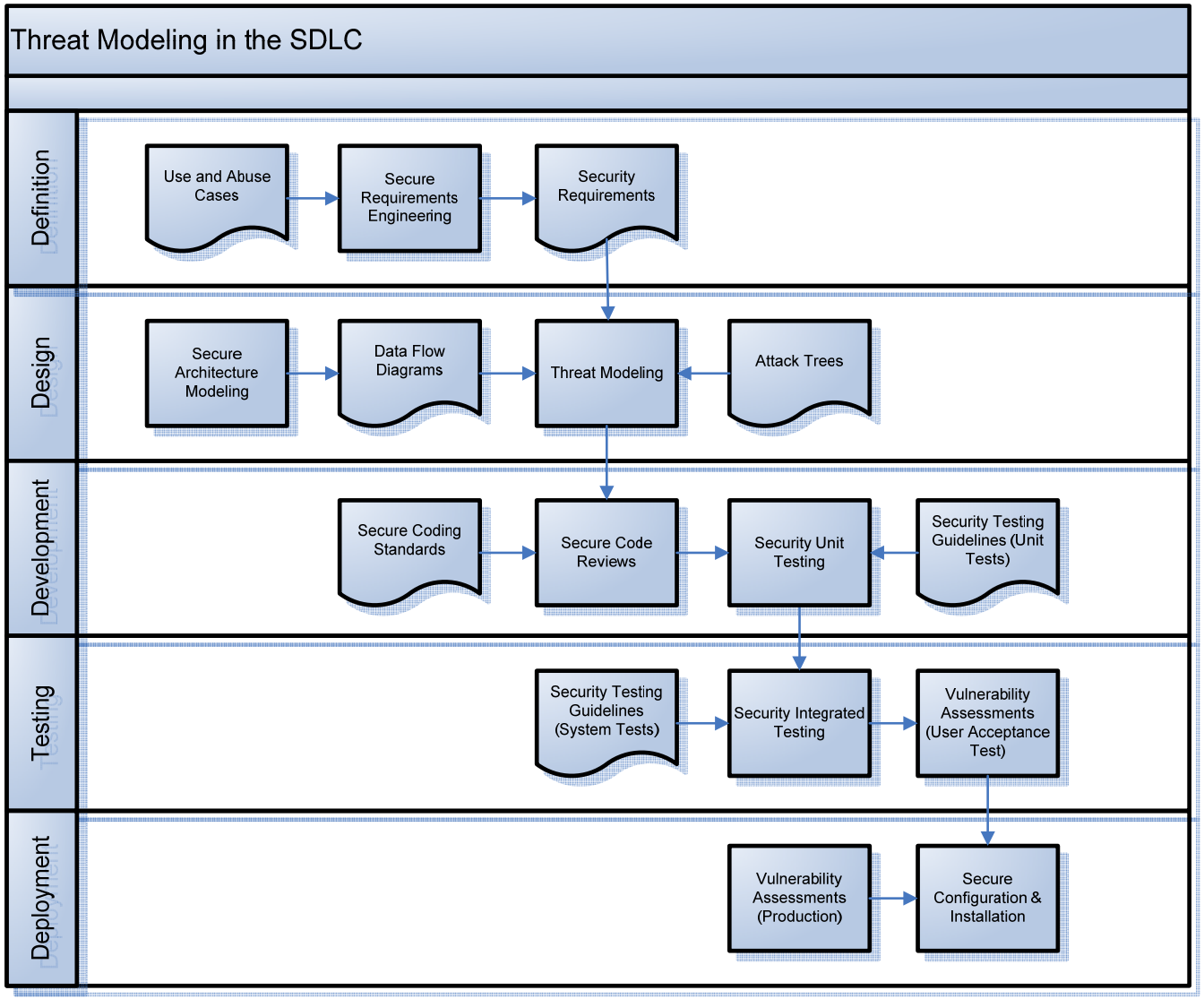
- Architecture Patterns
  - Structural organization or schema for software systems.
  - High level and conceptual
    - MVC, ASP.NET
    - Apache Struts JAVA-OBJ(M), JSP(V)-ActionForm(C)
- Design Patterns
  - Scheme for refining the subsystems or components of a software system, or the relationships between them.
  - Code Based
    - Factory-AUTH, Prototype-RBAC, Singleton-Logs, Adapter-IV, Façade-KISS, Strategy-ENC

Conclusions

- The OWASP T10 is just a checklist
- A checklist is not actionable without process, training and tools
- A tactical approach is to look at OWASP T10 root causes (e.g. security bugs and security flaws)
- Security flaws can be identified by looking at threats and countermeasures in the application design architecture
- A strategic approach is to secure by design by documenting secure application design requirements

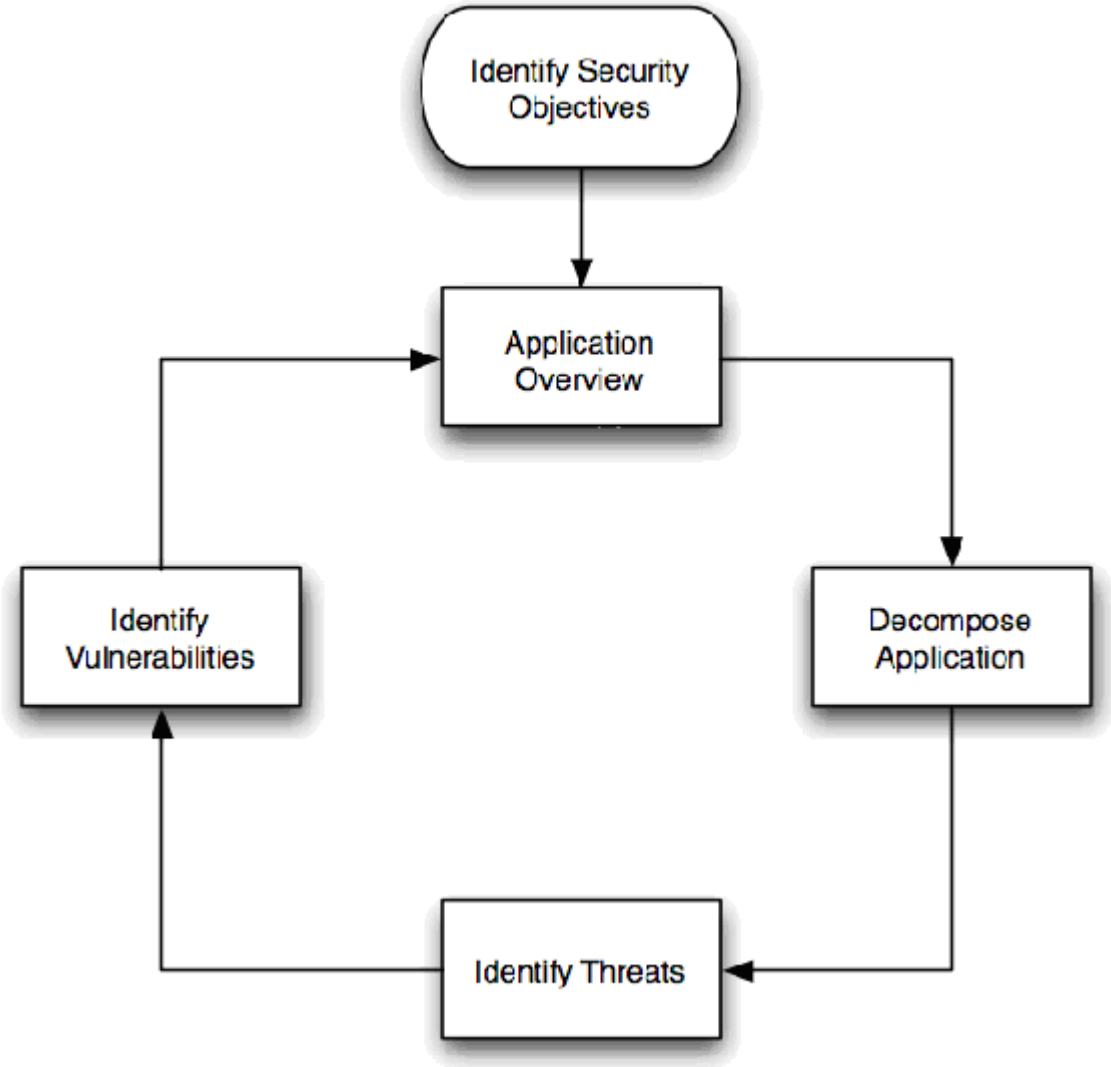
QUESTIONS ANSWERS

APPENDIX: Application Threat Modeling



From Insecure Magazine, http://www.net-security.org/dl/insecure/INSECURE-Mag-17.pdf

OWASP Threat Risk Modeling Cycle



OWASP Threat Risk Modeling http://www.owasp.org/index.php/Threat_Risk_Modeling

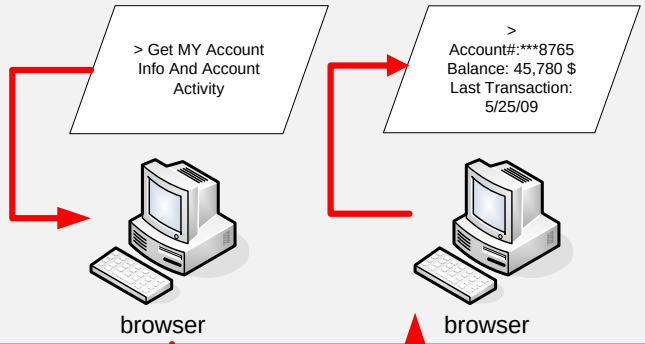
Step 1: Identify Security Objectives

- Tactical Security Assessments
  - Identification of security flaws, the threats that can exploit them and the mitigations
- Secure Architecture Design
  - Gap analysis on security requirements
  - Review of architecture and security controls
  - Design of countermeasures that mitigate threats
- Application Risk Management
  - Technical risk and business impact
  - Compliance risks
  - Support for risk mitigation strategy
    - Mitigate, Transfer, Accept it

Step 2: Application Overview

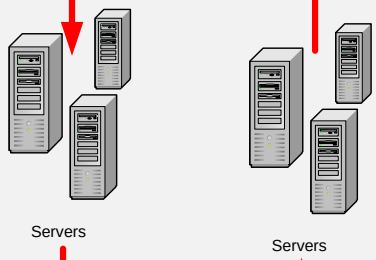
Presentation Tier

Represents the top most level
of the application.
The purpose of this tier is to translate
commands from the user interface
into data for processing to other tiers and
present back the processed data



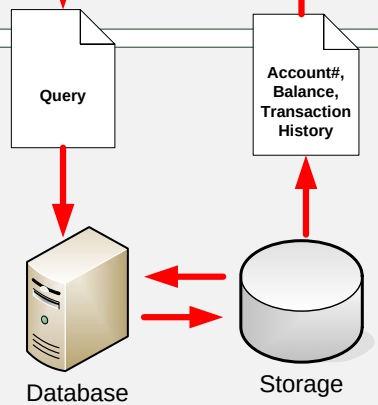
Logic Tier

This layer processes commands and
makes decisions based upon
the application business logic
It also moves and processes data
between the presentation and the data tier



Data Tier

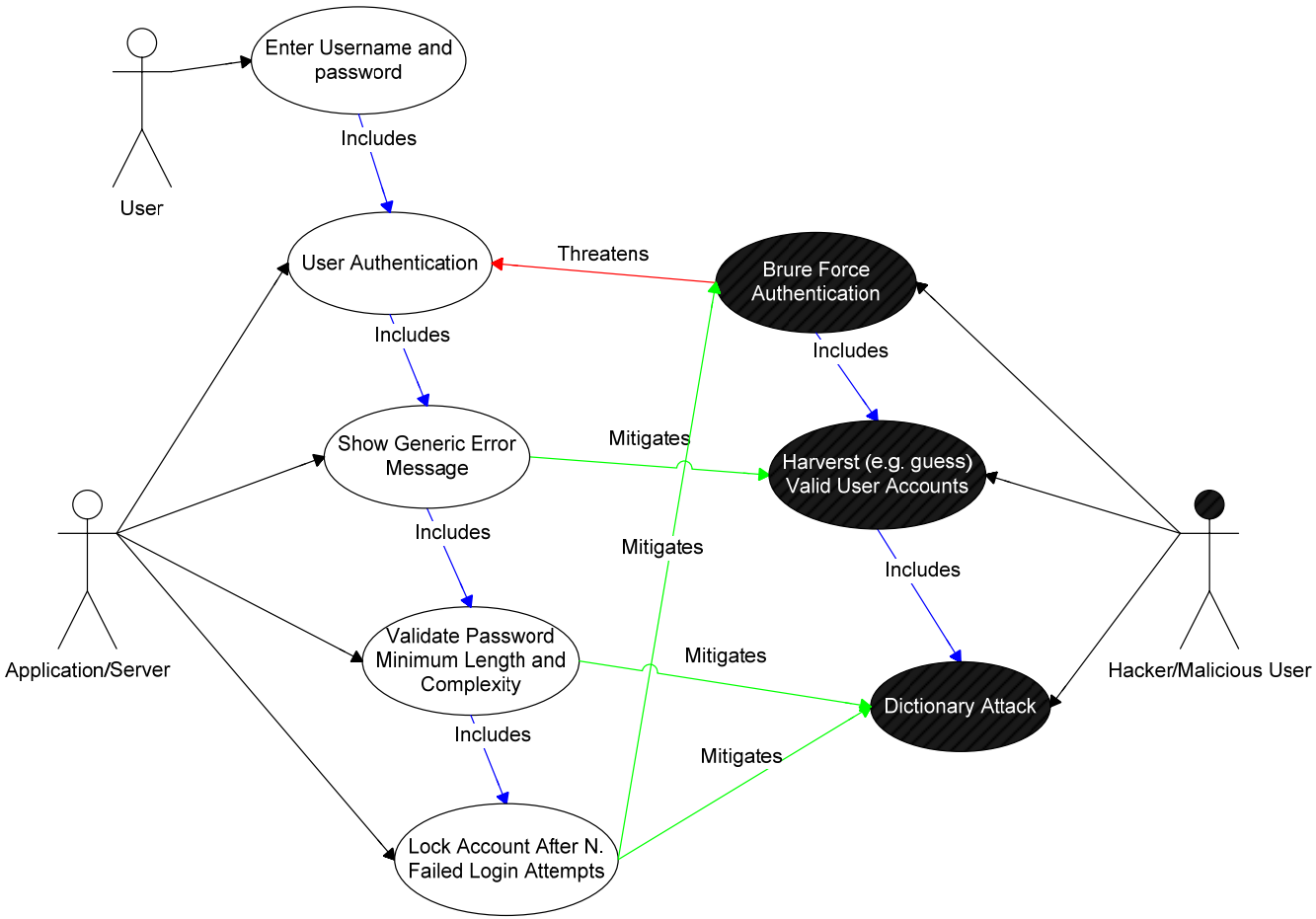
Is the layer responsible for data storage and
retrieval from a database or file system
Query commands or messages are processed
by the DB server, retrieved from the datasource
and passed back to the lo the logical tier for
processing before being presented to the user



Step 3: Decompose the Application

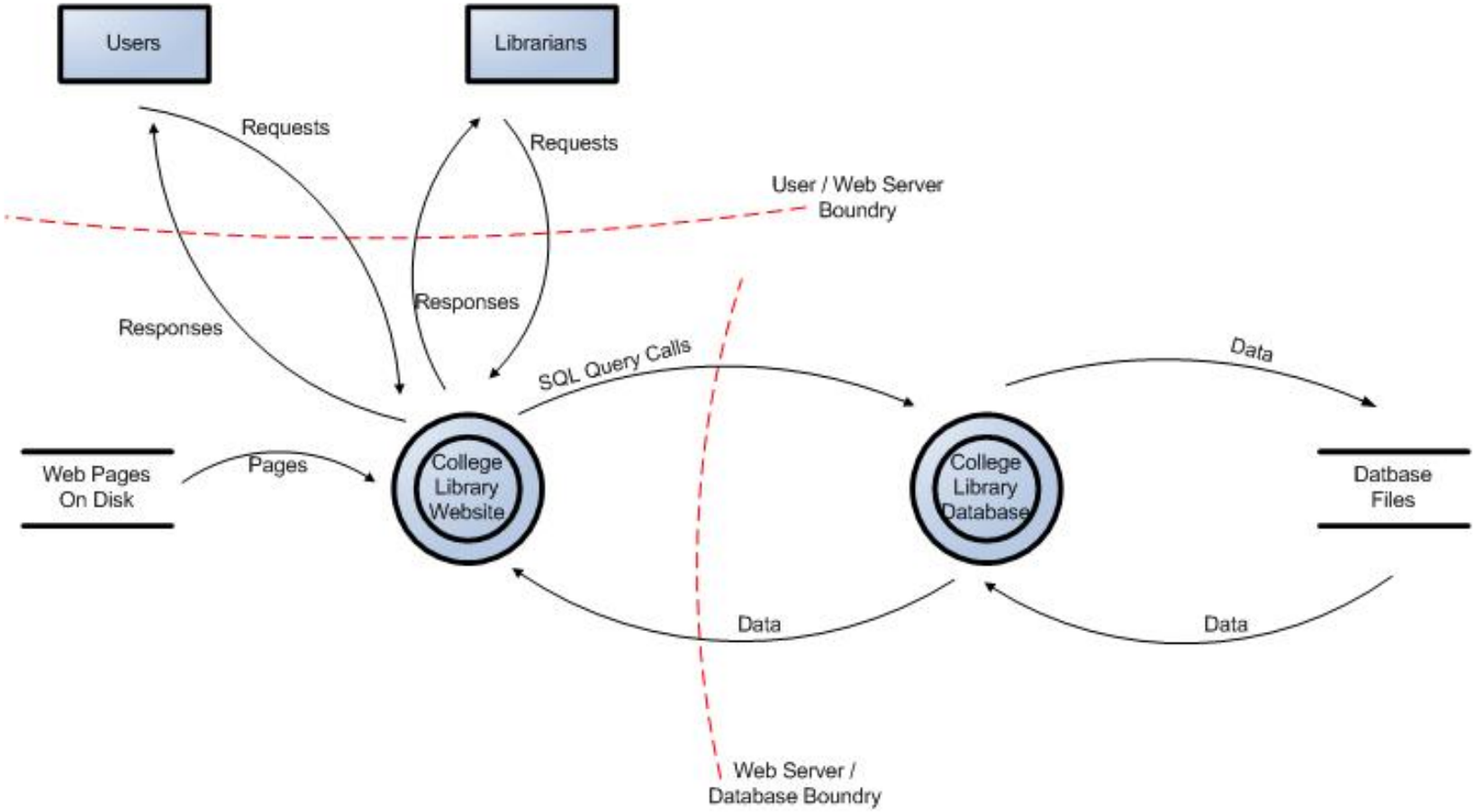
- Objective: understand the application and the interaction with external and internal entities by deriving:
  - Use and Misuse Cases
  - Entry/Exit Points
  - Trust Boundaries
  - Assets
  - Data Flows and Communication Channels
- Outcome: Data flow diagrams (DFD) for the application show the different paths through the system highlighting the privilege boundaries.

Use and Abuse Cases



From OWASP Security Testing Guide
https://www.owasp.org/index.php/Testing_Guide_Introduction#Security_Requirements_Test_Derivation

Understanding the Application: Data Flow Diagrams



OWASP Application Threat Modeling https://www.owasp.org/index.php/Application_Threat_Modeling

Step 4: Threat Identification

- Objective: Use a systematic approach to identify security control gaps by applying threats to the decomposed application. Determine potential attacks and threats using:
  - Generic Threat-Attack/Controls Lists (STRIDE, ASF)
  - Attack Trees
  - Attack Libraries
- Outcome: List of threats to which the application component (e.g. data, entry point, trust boundary, server/process etc) is exposed to

STRIDE Threat Categorization

STRIDE Threat List		
Type	**Examples**	**Security Control**
Spoofing	Threat action aimed to illegally access and use another user's credentials, such as username and password	Authentication
Tampering	Threat action aimed to maliciously change/modify persistent data, such as persistent data in a database, and the alteration of data in transit between two computers over an open network, such as the Internet	Integrity
Repudiation	Threat action aimed to perform illegal operation in a system that lacks the ability to trace the prohibited operations.	Non-Repudiation
Information disclosure.	Threat action to read a file that they were not granted access to, or to read data in transit.	Confidentiality
Denial of service.	Threat aimed to deny access to valid users such as by making a web server temporarily unavailable or unusable.	Availability
Elevation of privilege.	Threat aimed to gain privileged access to resources for gaining unauthorized access to information or to compromise a system.	Authorization

OWASP Application Threat Modeling https://www.owasp.org/index.php/Application_Threat_Modeling

Threat Categorization: ASF Threat List

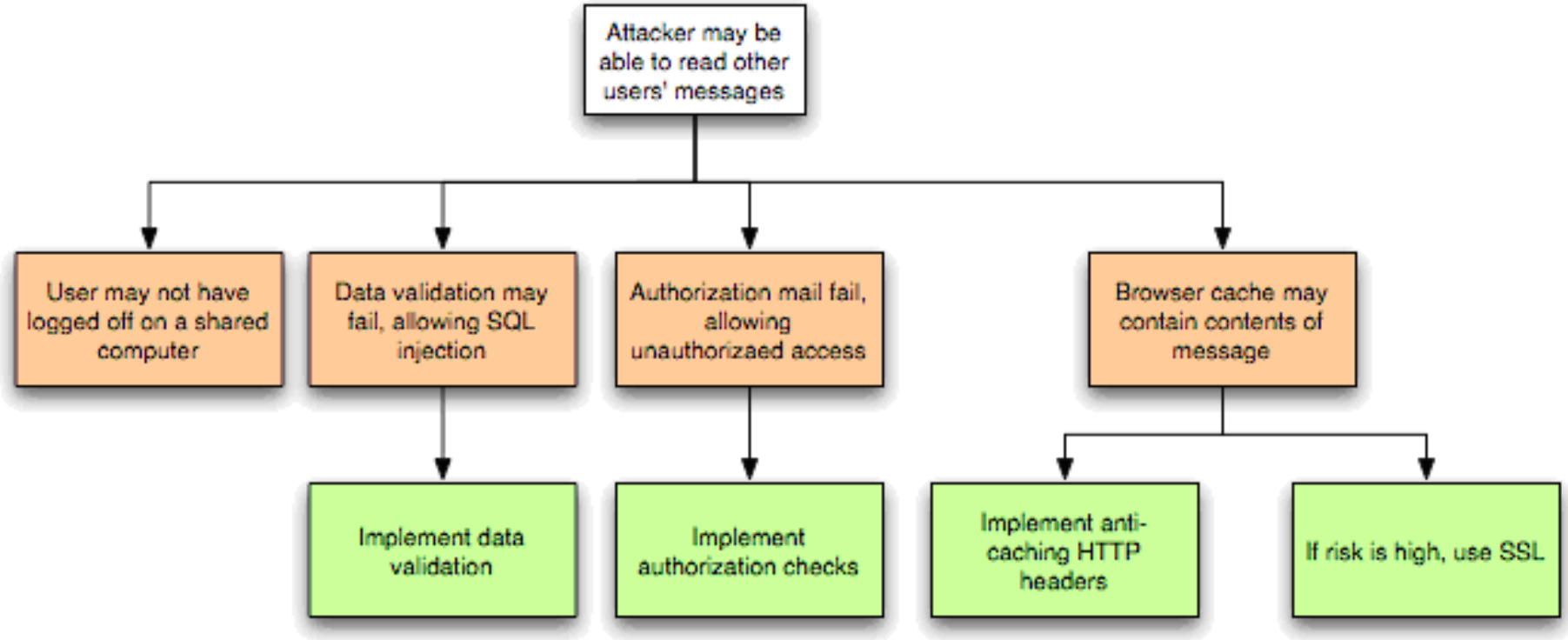
Type	Description	Attack Examples
Auditing & Logging	Threats caused by failure to maintain detailed and accurate application logs that can allow for traceability and non-repudiation and provide enough information for administrators to identify security issues and for incident response specialists to trace an attack. An attacker can use logs to obtain critical information about the system as well as tamper them to clear his traces after an attack.	1. Non repudiation 2. Cleaning the logs to remove evidence 3. Inserting of faked logging entries
Authentication	Threats caused by lack of strong protocols to validate the identity of a user to access a system or component outside the trust boundary. An attacker can obtain illegitimate access to the system or its individual components.	1. Impersonation 2. Spoofing attacks 3. Brute force attacks 4. Dictionary attacks 5. Token/Cookie Reply attacks 6. Man in The Middle attacks(MiTM)
Authorization	Threats caused by lack of a mechanisms to enforce access control on protected resources within the system. An attacker can get access to resources that should not have privileges to access to.	1. Escalation of privileges 2. Luring attacks 3. Forceful browsing
Configuration Management	Threats caused by insecure deployment and administration. An attacker can get access to system/user data and gain unauthorized access to system functionality	1. Elevation of privileges 2. Unauthorized access to configuration data 3. Unauthorized access to administration interfaces
Data Protection in Storage and Transit	Threats caused by the implementation of encryption and lack of adequate protection for secrets and other data in storage or transit. An attacker can compromise user/system confidentiality and data integrity by bypassing the cryptographic mechanisms used by the system	1. Access to sensitive data in storage 2. Access to secrets (e.g. encryption keys) 3. Eavesdropping of data during transmission 4. Data tampering 5. Brute force attacks to crack encryption

https://www.owasp.org/index.php/Application_Threat_Modeling

Step 5: Vulnerability Identification

- Objective: Identify vulnerabilities due to un-mitigated threats using:
  - Vulnerabilities are threats with no countermeasures
  - Use threat-countermeasures lists (STRIDE, ASF)
  - Use threat trees
- Outcome: a threat profile describing the security flaws of the application in terms of:
  - The threat and the impacted network, host, server/tier, component, data
  - The vulnerability being exploited by the threat
  - The impact (e.g. disruption/denial of service)
  - The severity of the vulnerability (e.g. High, Medium, Low)
  - The recommended mitigation control (e.g. countermeasure)

Threats, Vulnerability Conditions and Mitigations: Threat Trees



OWASP Application Threat Modeling https://www.owasp.org/index.php/Application_Threat_Modeling

STRIDE Threat And Countermeasures

STRIDE Threat & Mitigation Techniques List	
Threat Type	**Mitigation Techniques**
Spoofing Identity	1. Appropriate authentication 2. Protect secret data 3. Don't store secrets
Tampering with data	1. Appropriate authorization 2. Hashes 3. MACs 4. Digital signatures 5. Tamper resistant protocols
Repudiation	1. Digital signatures 2. Timestamps 3. Audit trails
Information Disclosure	1. Authorization 2. Privacy-enhanced protocols 3. Encryption 4. Protect secrets 5. Don't store secrets
Denial of Service	1. Appropriate authentication 2. Appropriate authorization 3. Filtering 4. Throttling 5. Quality of service
Elevation of privilege	1. Run with least privilege

OWASP Application Threat Modeling https://www.owasp.org/index.php/Application_Threat_Modeling

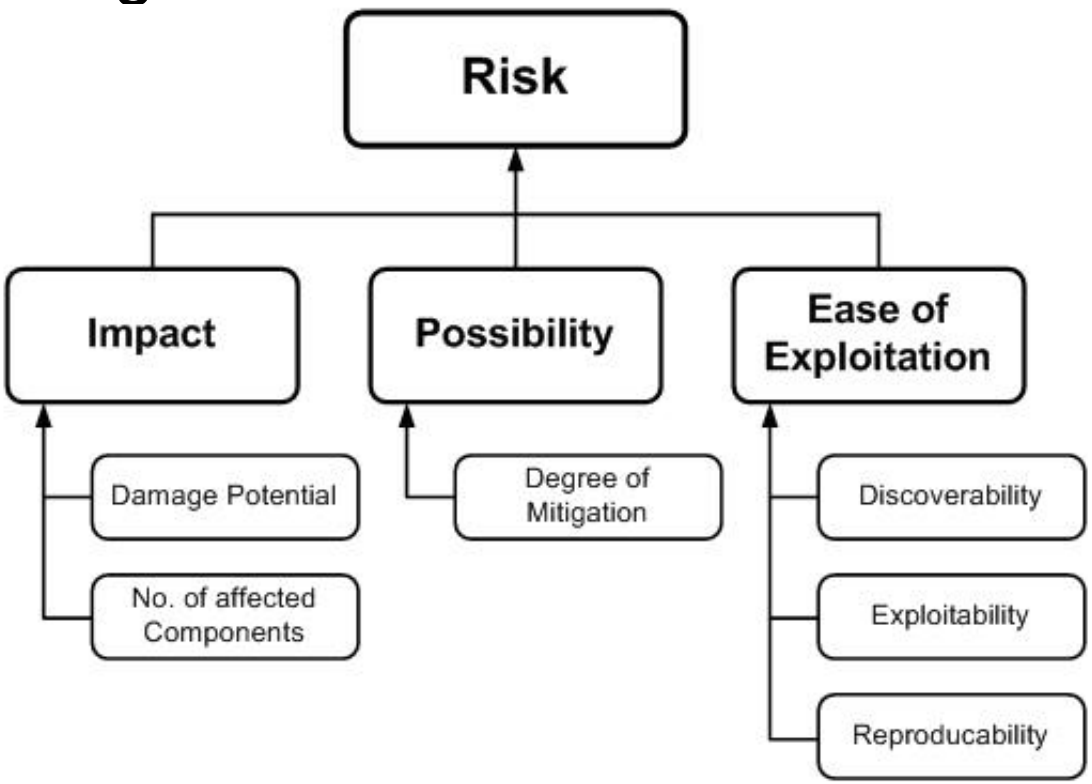
Countermeasure Identification

ASF Threat & Countermeasures List	
Threat Type	**Countermeasure**
Authentication	1. Credentials and authentication tokens are protected with encryption in storage and transit 2. Protocols are resistant to brute force, dictionary attack and replay attacks 3. Strong password policies are enforced 4. Trusted server authentication is used instead of SQL authentication 5. Passwords are stored with salted hashes 6. Password resets do not reveal password hints and valid usernames 7. Account lockouts do not result in a denial of service attack
Authorization	1. Strong ACLs are used for enforcing authorized access to resources 2. Role-based access controls are used to restrict access to specific operations. 3. The system follows the principle of least privilege for user and service accounts 4. Privilege separation is correctly configured within the presentation, business and data access layers.
Configuration Management	1. Least privileged processes are used and service accounts with no administration capability 2. Auditing and logging of all administration activities is enabled 3. Access to configuration files and administrator interfaces is restricted to administrators
Data Protection in Storage and Transit	1. Standard encryption algorithms and correct key sizes are being used 2. Hashed message authentication codes (HMACs) are used to protect data integrity 3. Secrets (e.g. keys, confidential data) are cryptographically protected both in transport and in storage 4. Built-in secure storage is used for protecting keys 5. No credentials and sensitive data are sent in clear text over the wire

https://www.owasp.org/index.php/Application_Threat_Modeling

Risk Factors

- Threats can be classified (e.g. ranked) according to risk factors to allow informed decisions on risk mitigation



OWASP Application Threat Modeling https://www.owasp.org/index.php/Application_Threat_Modeling

Threats and Risk Models

- Qualitative Risk Models
  - **Risk = Probability (Degree of Mitigation, Exploitability) * Impact (*Damage Potential*)**
- Another method for determining risk is by ranking threats according to DREAD model:
  - Damage potential – How great is the damage if the vulnerability is exploited?
  - Reproducibility – How easy is it to reproduce the attack?
  - Exploitability – How easy is it to launch an attack?
  - Affected users – As a rough percentage, how many users are affected?
  - Discoverability – How easy is it to find the vulnerability?
    - **Risk = Min(D, (D+R+E+A+D) / 5)**

Risk Mitigation Strategies

- Threats can be resolved by:
  - Risk Acceptance - doing nothing
  - Risk Transference - pass risk to an externality
  - Risk Avoidance - removing the feature/component that causes the risk
  - Risk Mitigation - decrease the risk
- Mitigation strategies should be
  - Examined for each threat
  - Chosen according to the appropriate technology
  - Prioritized according to risk level and the cost of mitigations