DENIM GROUP

the leading secure software development firm

Threat Modeling for System Builders and System Breakers

Dan Cornell
@danielcornell

Dan Cornell

- Dan Cornell, founder and CTO of Denim Group
- Software developer by background (Java, .NET, etc)
- OWASP San Antonio
- 15 years experience in software architecture, development and security
- Heads Denim Group's application security team

Denim Group Background

- Professional services firm that builds & secures enterprise applications
  - *External application assessments*
    - Web, mobile, and cloud
  - *Software development lifecycle development (SDLC) consulting*
- Classroom and e-Learning for PCI compliance
- Secure development services:
  - *Secure .NET and Java application development*
  - *Post-assessment remediation*
- Deep penetration in Energy, Financial Services, Banking, Insurance, Healthcare and Defense market sectors
- Customer base spans Fortune 500
- Contributes to industry best practices through the Open Web Application Security Project (OWASP)

Agenda

- The Goals of Threat Modeling
- Understanding Threats and Risk
- Threat Modeling Activities
  - *Business*
  - *Architectural*
  - *Functional*
  - *Threat Trees*
  - *Rating Threats*
  - *Countermeasures*
- Effective Threat Modeling
- Threat Modeling Scenarios

The Goals of Threat Modeling

The Goals of Threat Modeling

- Take note we are discussing the goals, not the definition of threat modeling
- Definitions of a Threat Model differ
  - *A general list of security concerns for a given product*
  - *A set of possible application attack scenarios to consider*
  - *Structured way to look at systems and identify potential security issues*
  - *A set of attacker profiles, with attacker goals and competencies*
- These definitions are fine as definitions, but are not immediately actionable
- A threat model is difficult to define well because it does not fit a strict hierarchy or taxonomy
- We CAN identify the value in it

What Do We Want to Accomplish?

- Stay "ahead" of potential vulnerabilities
- Bring the application as planned or built in-line with security objectives
- Empower security verification activities
- Account for remediation and mitigation of problems as early as possible

- The earlier in the development / procurement process, the better
  - *The types of issues it identifies can be expensive to address after the fact.*

What Do We Want to Accomplish?

- In assessments
  - *Organize relevant observations about an application's architecture, unique features, and functional security*
  - *Attack planning for tests and review beyond the "baseline"*
- In development
  - *Predict and account for threats to the application's unique architecture and features*
  - *Estimate probability of a true exploit (DREAD)*
  - *If functional security is not defined elsewhere, define it here*
    - AuthX
    - Input validation
    - Data protection
    - etc.
  - *Justify the expense of countermeasures*

How Can a Threat Model Support These Goals?

- Clearly identify the stakes
  - *What sensitive assets does the application handle?*
  - *Who are the threat agents?*
- Identify architecture and design-level issues
  - *Data Flow*
  - *Business Logic*
- Collate functional security solutions
  - *What is your authentication solution? Validation? Data protection?*

- Threat Modeling must support your goals, not the other way around

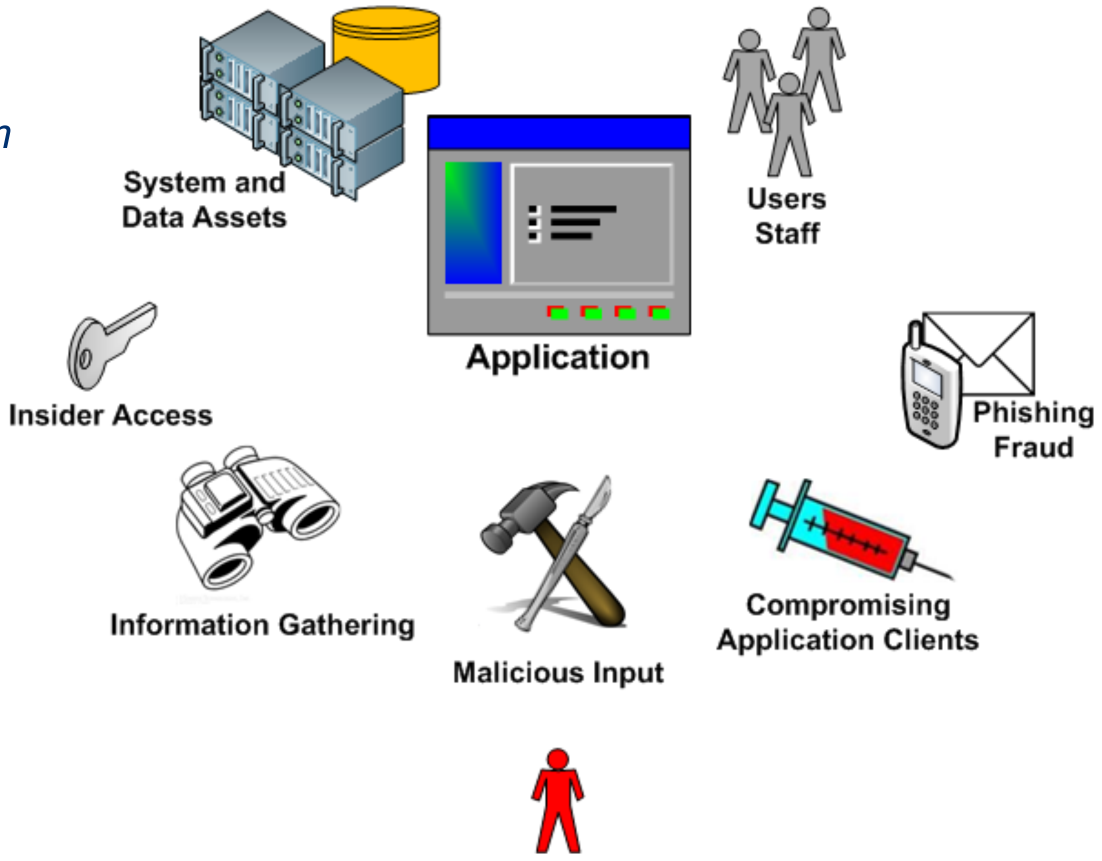
DENIM GROUP

the leading secure software development firm

Understanding Threats and Risk

Understanding Threats and Risk

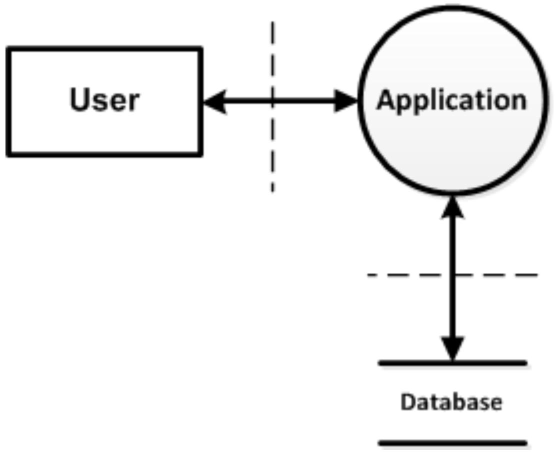
- High-Level attack vectors
  - *Defeating a security mechanism*
  - *Abusing an application feature*
  - *Exploiting the insufficient security or poor implementation*
- Remember, your application is part of a larger system



- Why is this important?

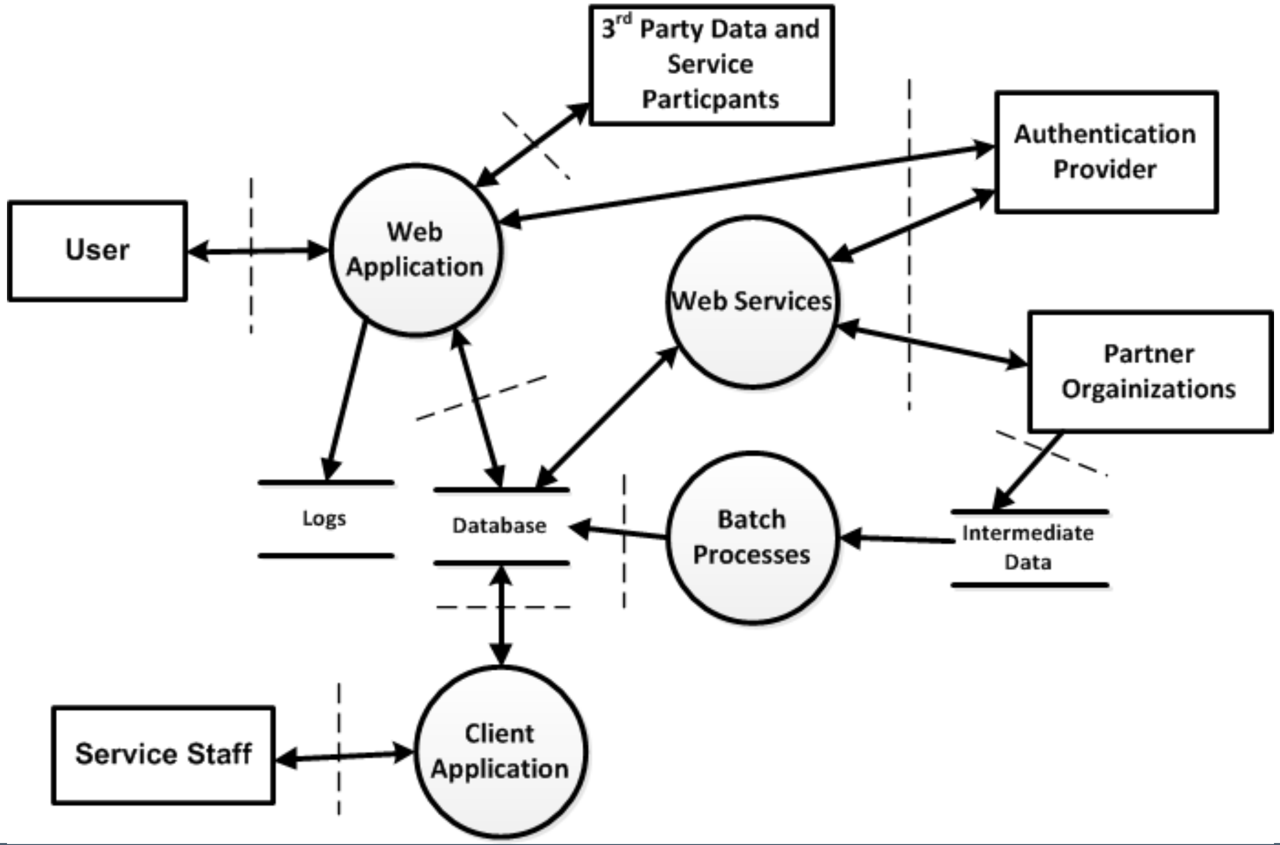
Understanding Threats and Risk

- What can seem like a simple application at first glance…



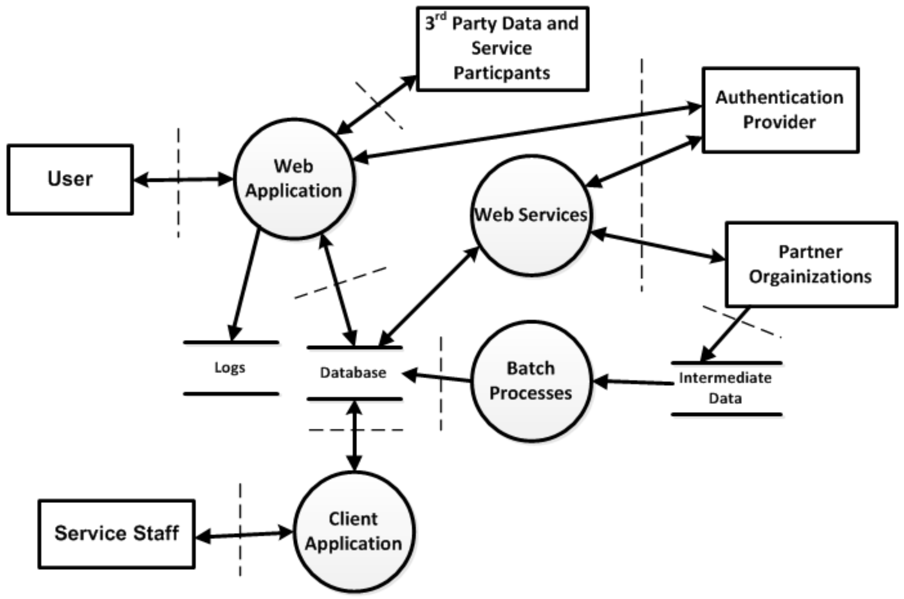
Understanding Threats and Risk

- Can quickly explode in complexity upon closer examination



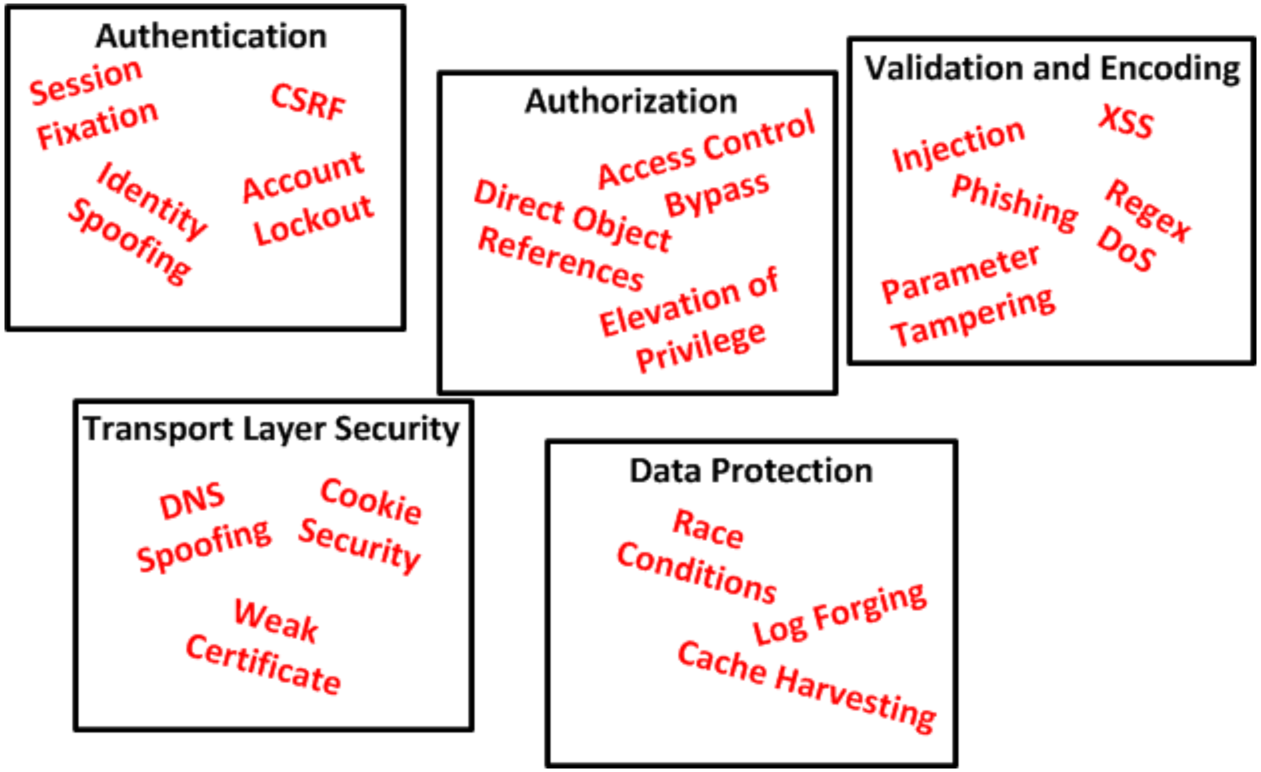
Understanding Threats and Risk

- Try not to decide the scope of an architecture review or security assessment before thinking of the big picture
- The weakest point in a system may not be what you think
- With the right information on-hand, discovering vulnerabilities can be a simple matter of Q&A



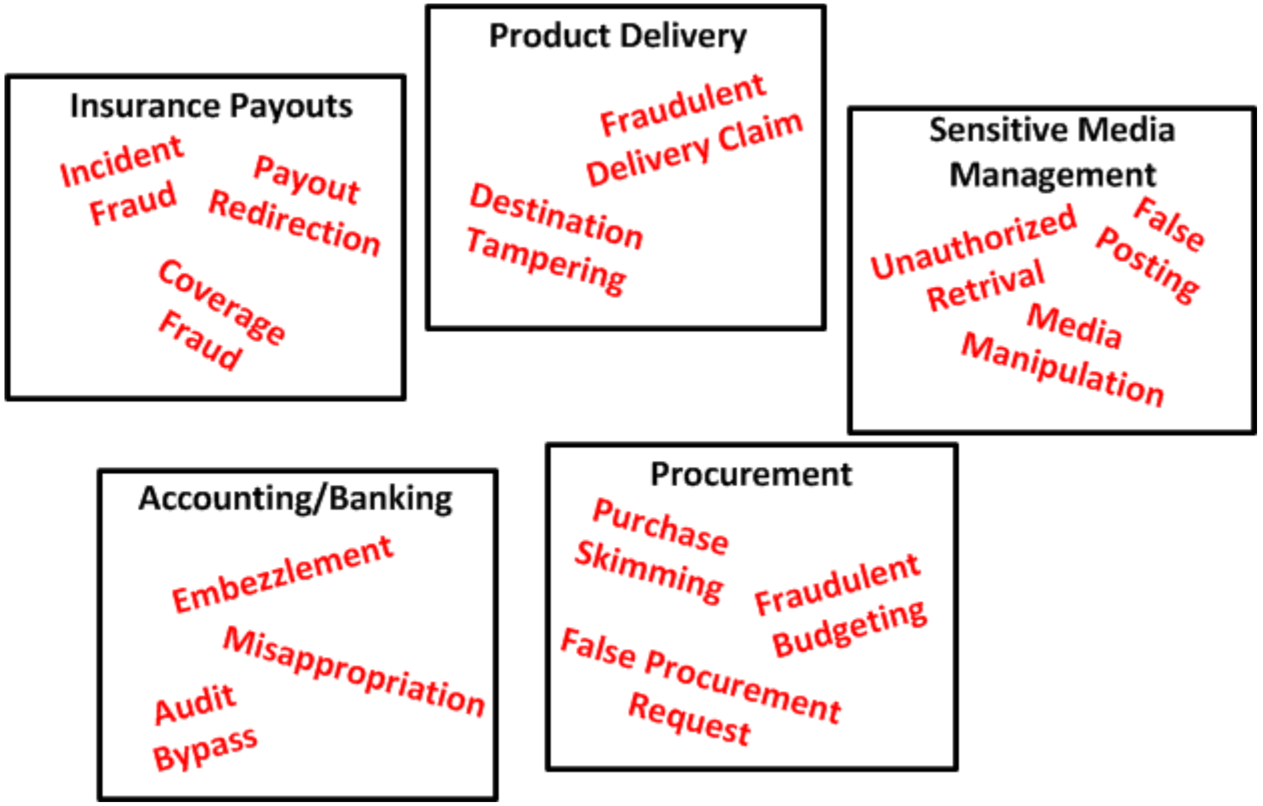
Understanding Threats and Risk

- We have seen the risks to poor functional security…



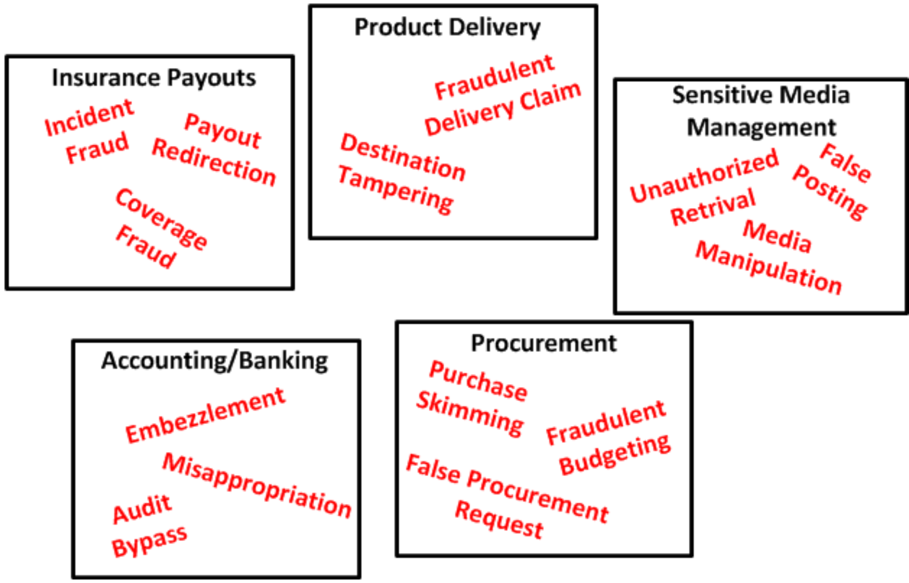
Understanding Threats and Risk

- What about the unique features of our application?



Understanding Threats and Risk

- Technology should not abstract business processes, but aid their efficient handling
- Application logic should not completely circumvent normal accountability



Understanding Threats and Risk

- What about functional security?
- You don’t need to be proficient with a particular technology to evaluate a security solution
  - *Is it adequate?*
  - *Do operational processes support it?*
  - *Is the solution an established, tested one or custom-mad1e?*

1. **Security Architecture**
2. **Authentication**
3. **Session Management**
4. **Access Control**
5. **Input Validation**
6. **Output Encoding/Escaping**
7. **Cryptography**
8. **Error Handling and Logging**
9. **Data Protection**
10. **Communication Security**
11. **HTTP Security**
12. **Security Configuration**
13. **Malicious Code Search**
14. **Internal Security**

Threat Modeling Activities

The Threat Modeling Process

- Let us take a clear look at what goes in and comes out of the process
  - *Business Logic*
  - *Architecture*
  - *Functional Security*
  - *Attacker's Perspective*

Threat Model Inputs and Outputs

- Business
  - *Application requirements, enterprise security strategy in*
  - *Assets, Agents, Abuse cases out*
- Architectural
  - *Application architecture and design in*
  - *Application data flow threats out*
- Functional
  - *Implementation standards or application source in*
  - *Functional security threats out*
- Threat Trees
  - *Attack surface and attacker profiles in*
  - *Threat trees out*

Approaches for Identifying Threats

- **Use Cases** for Business
  - *Useful for identifying flaws with specific application features*
- **Data Flow** for Architecture
  - *What threats can we identify looking at the application's data flow?*
  - *The whole system's data stores, services, processes, etc.*
  - *The interaction among those components*
- **Functional** Security
  - *Here are the security features. How could an attacker defeat them?*
- Attacker's Goals for **Threat Trees**
  - *If you are an attacker, what would you want to accomplish?*
  - *How would you go about achieving the malicious goal?*
  - *Useful for identifying any erroneous security assumptions*
- No one approach is perfect – these are essentially brain storming techniques

Threat Modeling Activities: Business

Threat Model Business Inputs and Outputs

- When can we do this?
  - *Any time the application is conceived*
- Business Inputs
  - *Requirements, Use Cases, other requirements-related documentation*
  - *Compliance, regulations, other strategic goals*
- Business Outputs
  - *Business Assets*
  - *Threat Agents*
  - *Abuse Cases*
- What have we accomplished?
  - *Data to scope future assessment efforts*
  - *Abuse scenarios for targeted security testing*

Identifying Threats through Abuse Cases

- This is where we catch potential flaws in business logic, customer support
- Look over each application use case
  - *What functionality fulfills that use case?*
  - *How would an attacker attempt to abuse that functionality?*
- If a use-case accounts for a user requesting a document, then the abuse case would account for a request to a document that they are not allowed to see or one that doesn't exist
- If a use-case accounts for a privileged user approving a transaction, then the abuse case would account for a lower-level user attempting to force approval for the transaction

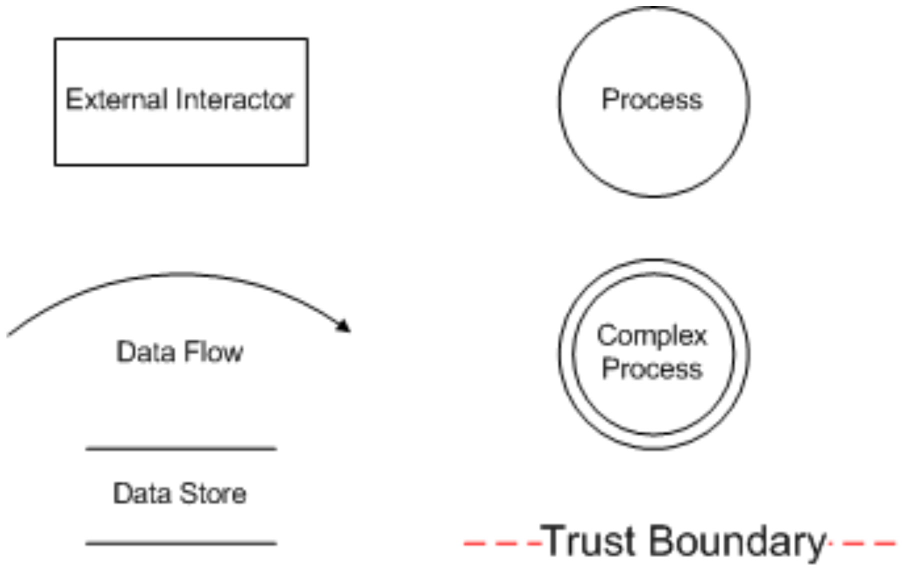
Threat Modeling Activities: Architectural

Architectural Threats

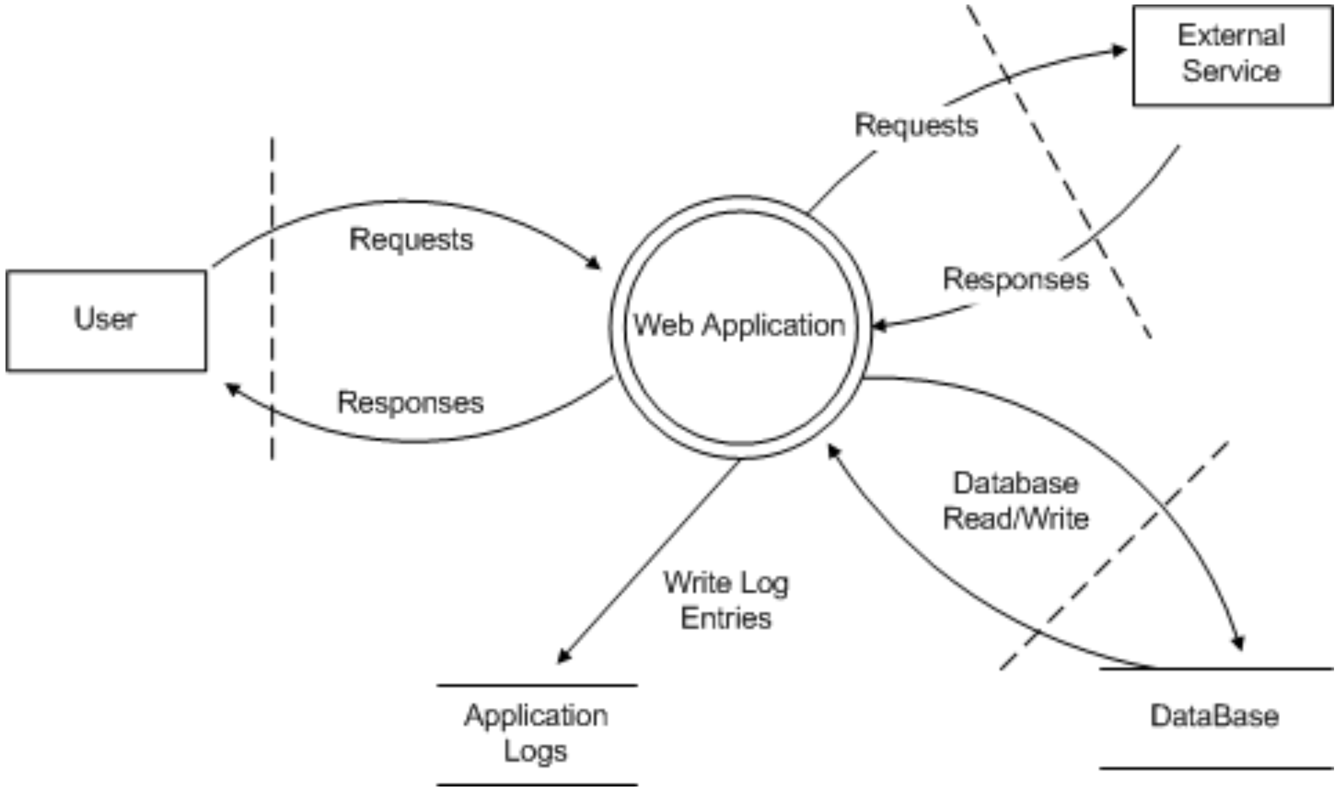
- When can we do this?
  - *When the application architecture has been conceived*
  - *It does not need to be final*
- What do we make?
  - *A Data Flow Diagram*
- We are going to do it in the Microsoft style
  - *Why? This is a good fit for many different types of systems*

Creating Data Flow Diagrams (DFDs)

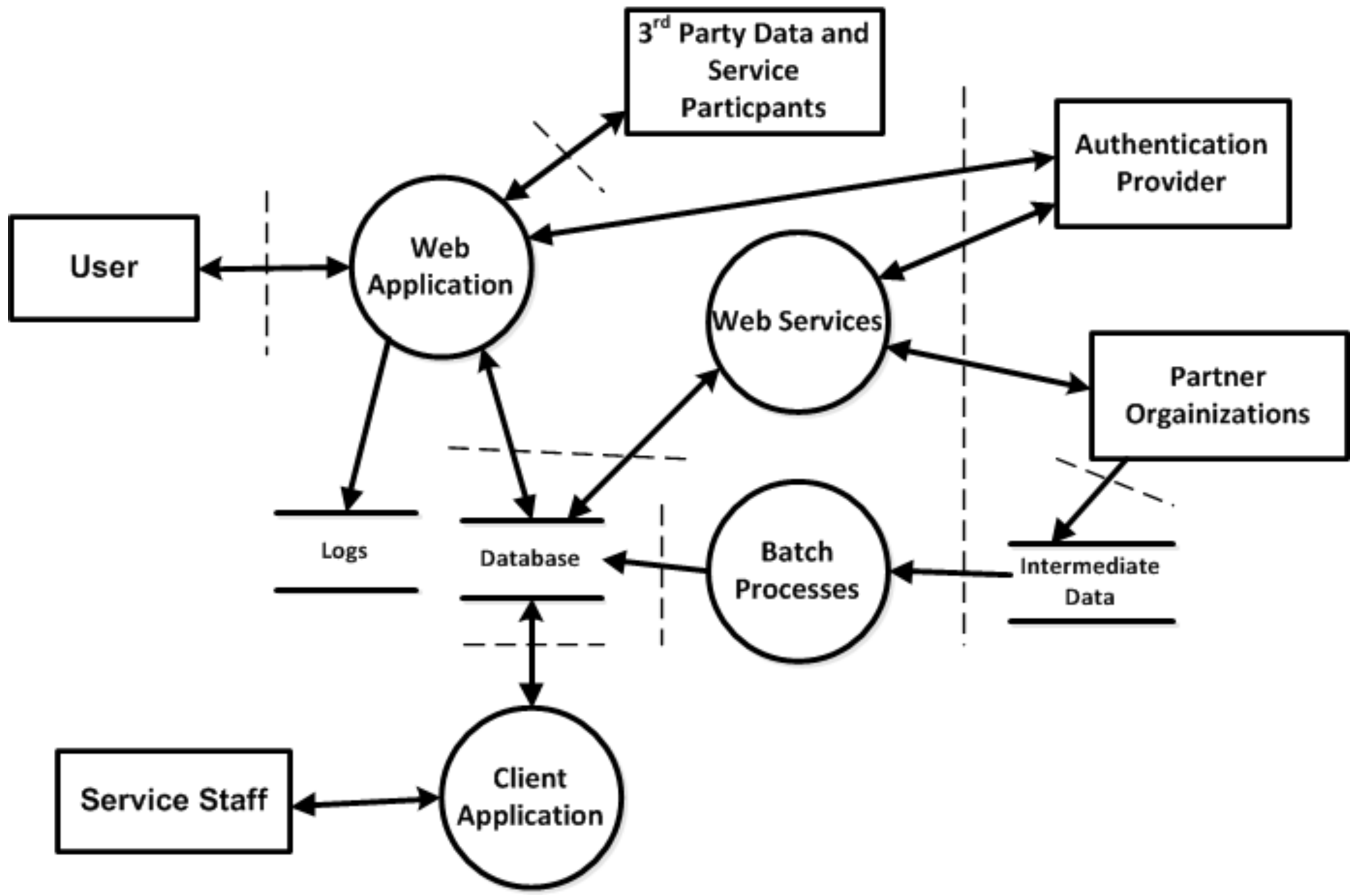
- Decompose the system into a series of processes and data flows
- Explicitly identify trust boundaries



Example Data Flow Diagram



Deriving Threats from the Data Flow



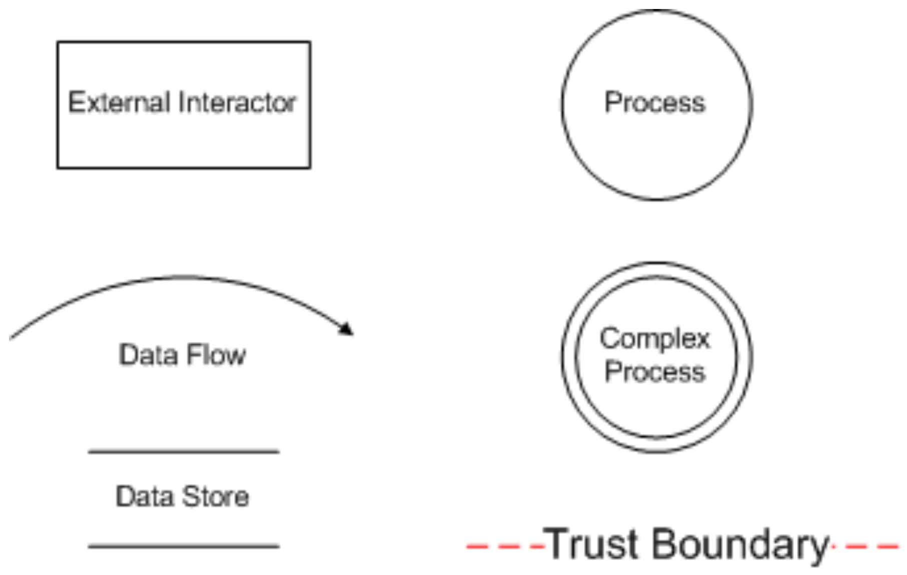
Data-Flow Based Threats

- This is where a Threat Model is uniquely beneficial
- Identify potential issues in a structured, repeatable manner
- Looking at asset types in the context of STRIDE

Identifying Threats from the Data Flow

- STRIDE is expansion of the common CIA threat types
  - *Confidentiality*
  - *Integrity*
  - *Availability*
- STRIDE
  - *Spoofing Identity*
  - *Tampering with Data*
  - *Repudiation*
  - *Information Disclosure*
  - *Denial of Service*
  - *Elevation of Privilege*

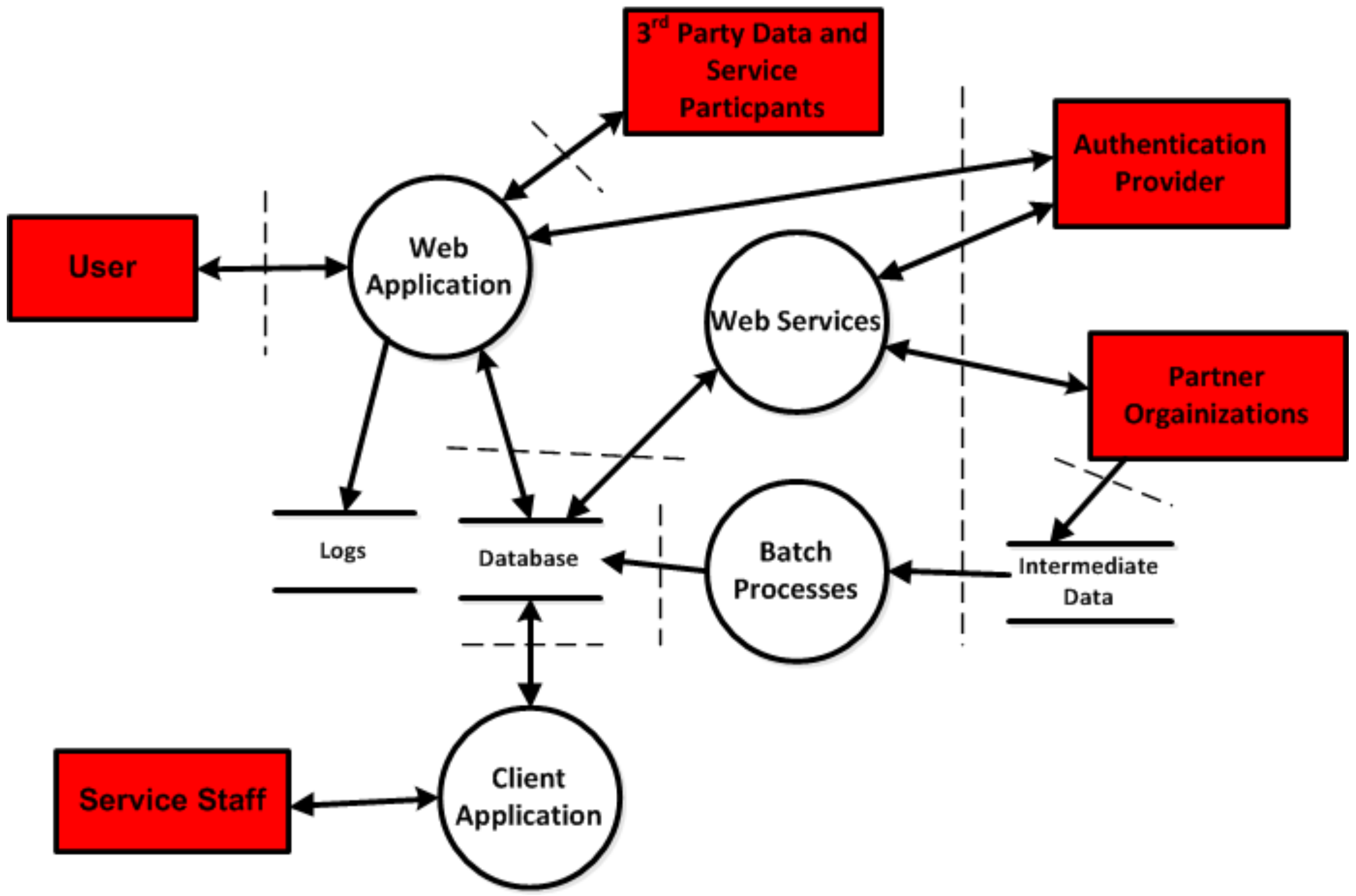
Asset Types


External Interactor
Process
Data Flow
Complex Process
Data Store
Trust Boundary

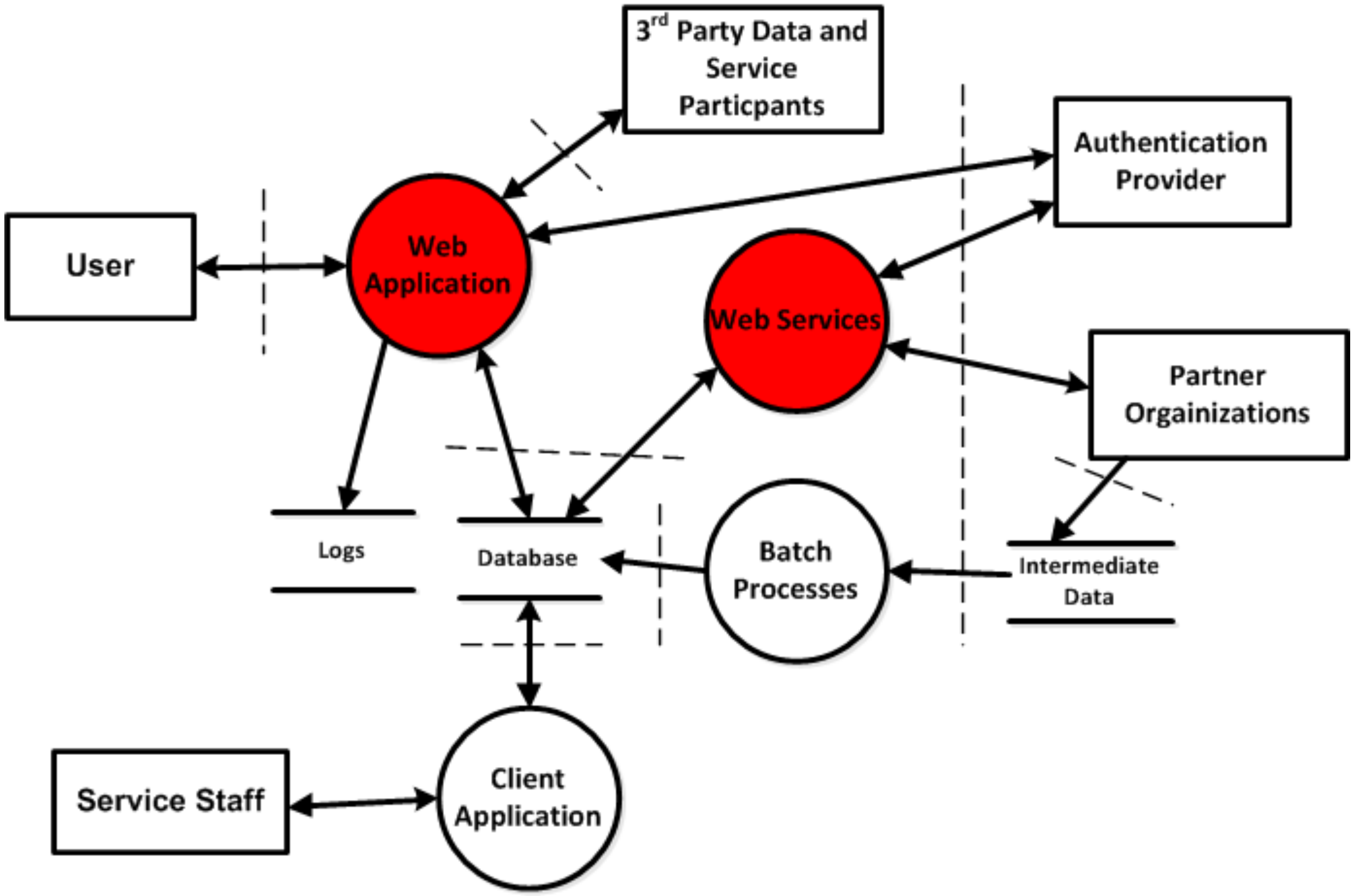
Mapping Threats to Data Flow Asset Types

Threat Type	External Interactor	Process	Data Flow	Data Store
S – Spoofing	Yes	Yes		
T – Tampering		Yes	Yes	Yes
R – Repudiation	Yes	Yes		Yes
I – Information Disclosure		Yes	Yes	Yes
D – Denial of Service		Yes	Yes	Yes
E – Elevation of Privilege		Yes		

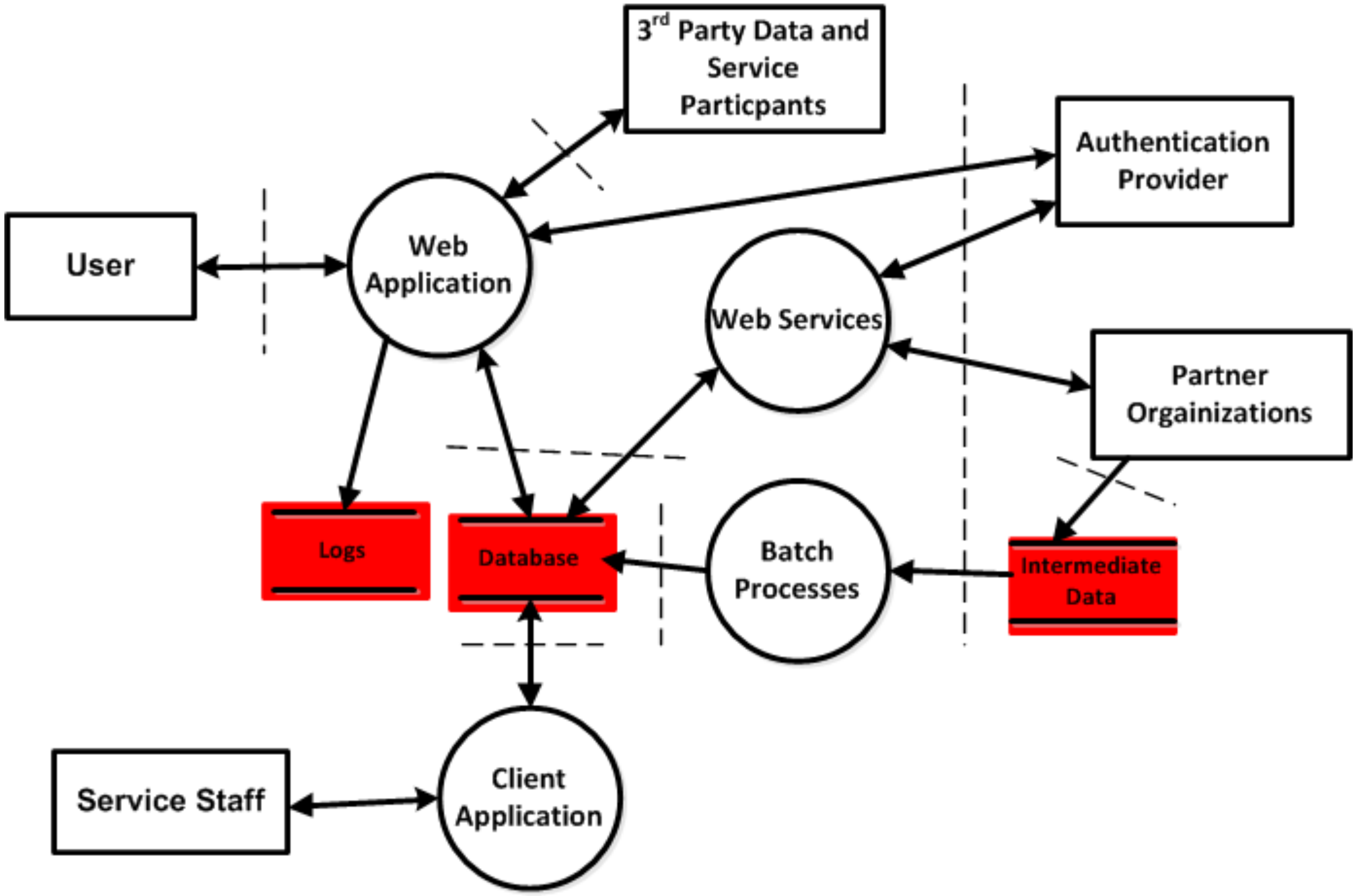
Spoofing: External Interactors



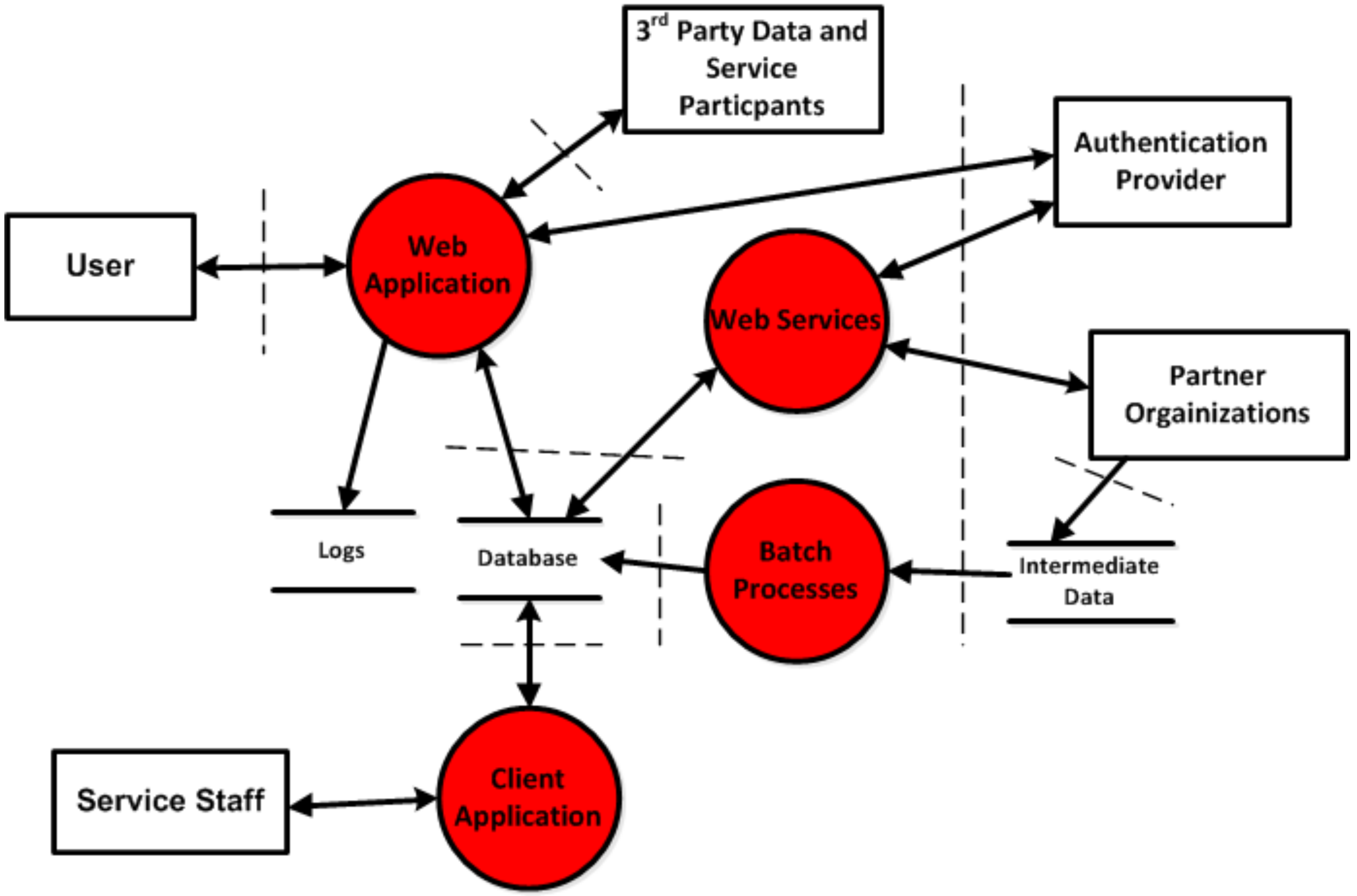
Spoofing: Remote Applications



Tampering: Data Stores

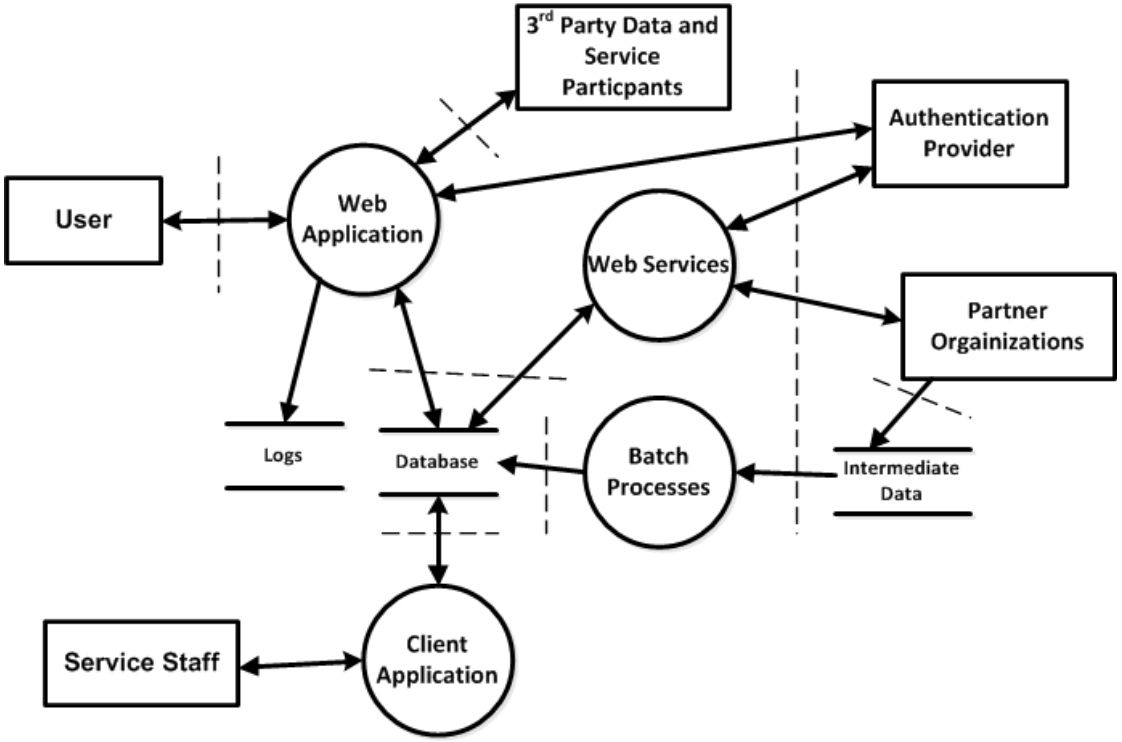


Tampering: Applications

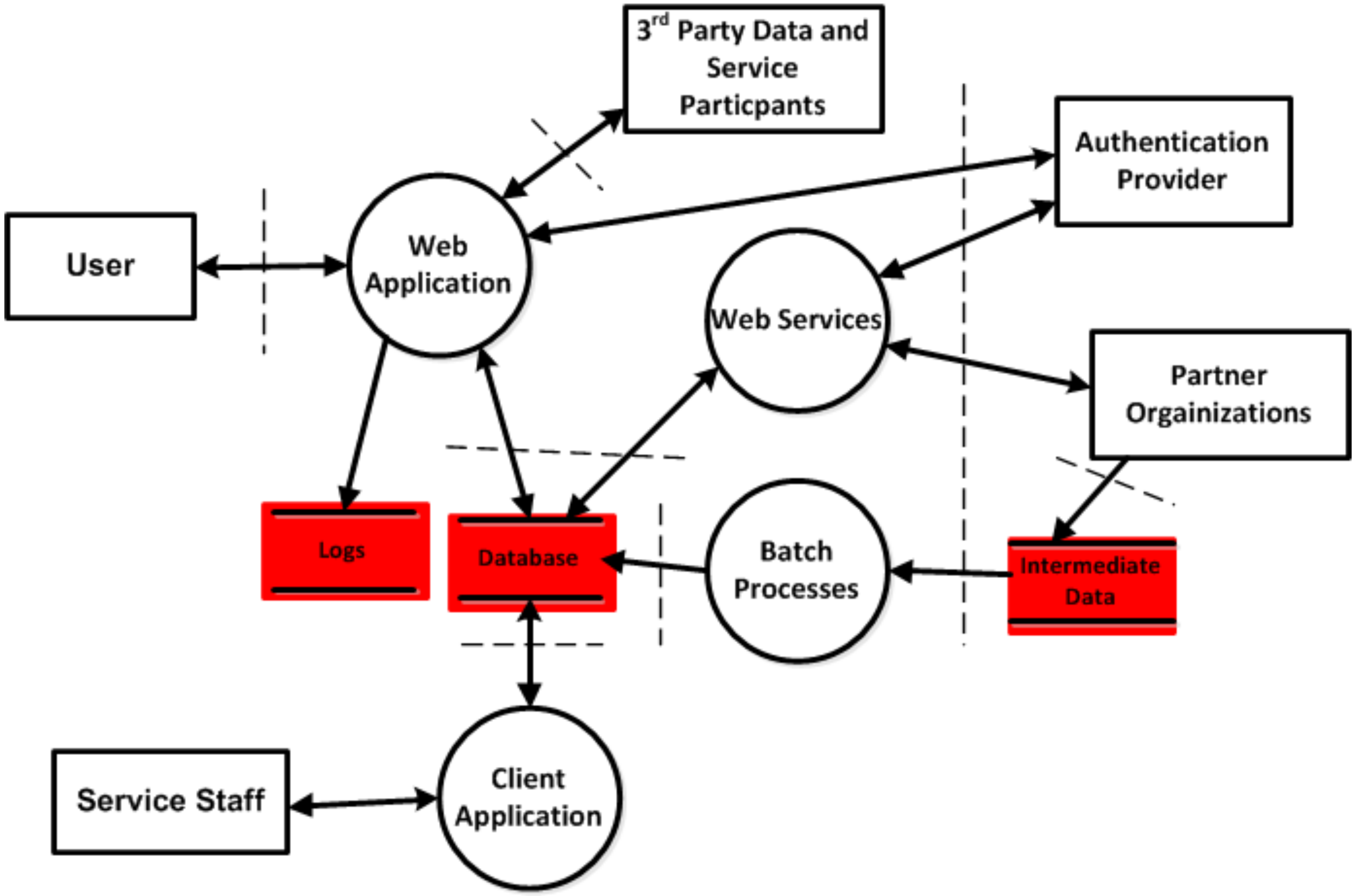


Repudiation

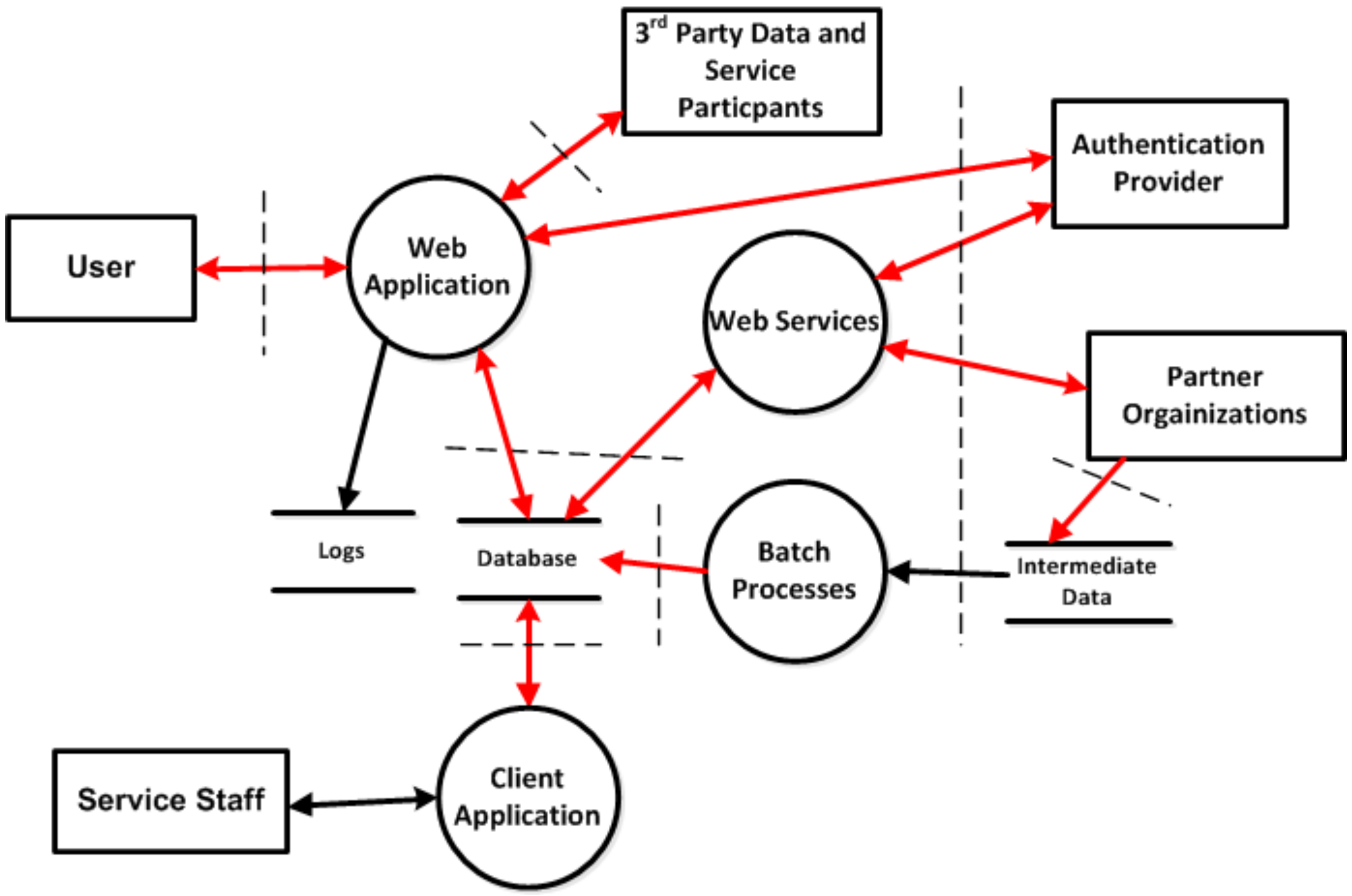
- What data reflects business transactions?
  - *Logs?*
  - *Database?*
  - *Other intermediate files?*
  - *External entities?*
- Is this data adequate?
  - *Safeguard against fraud?*
  - *Incident response?*



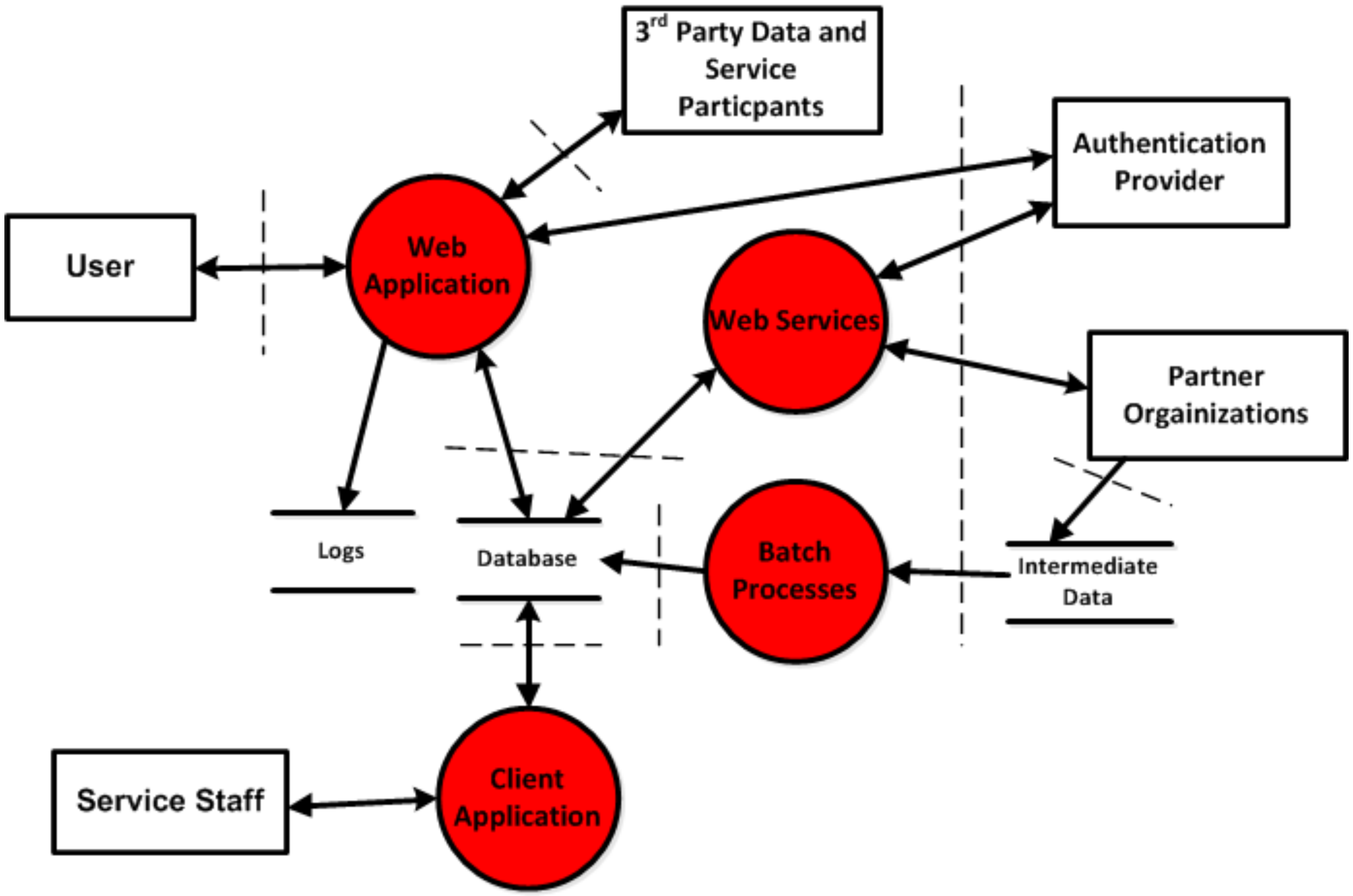
Information Disclosure: Data Stores



Information Disclosure: Transport Layer

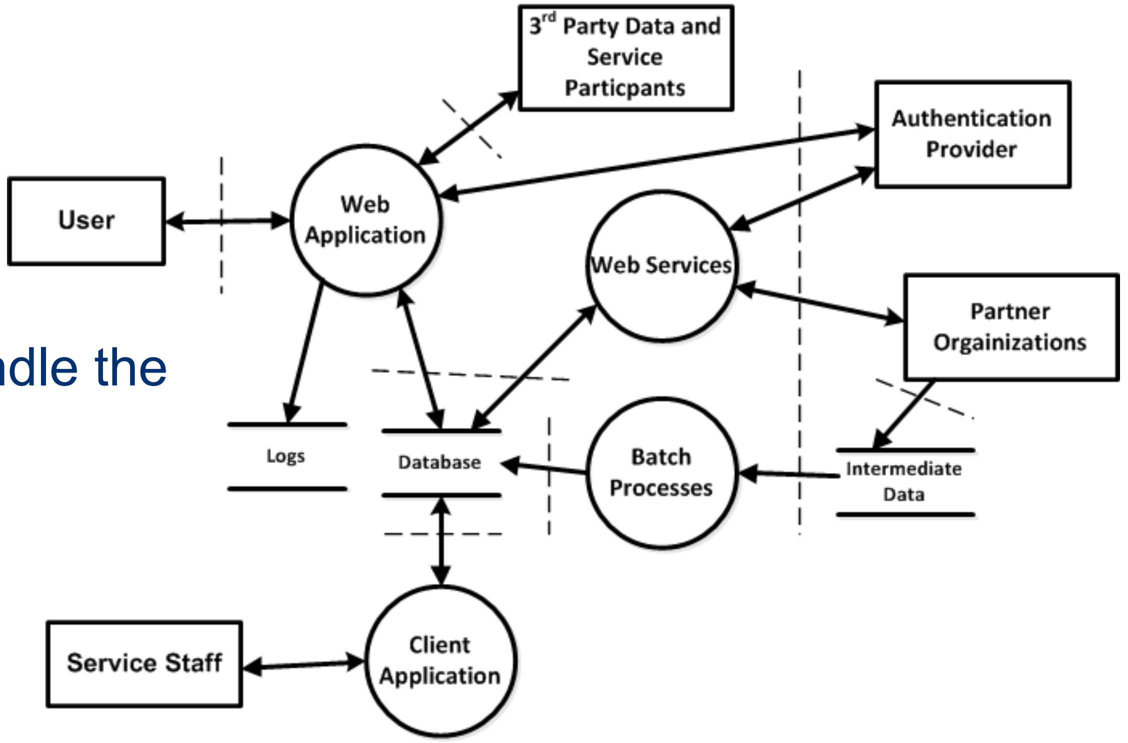


Information Disclosure: Technical and Operations

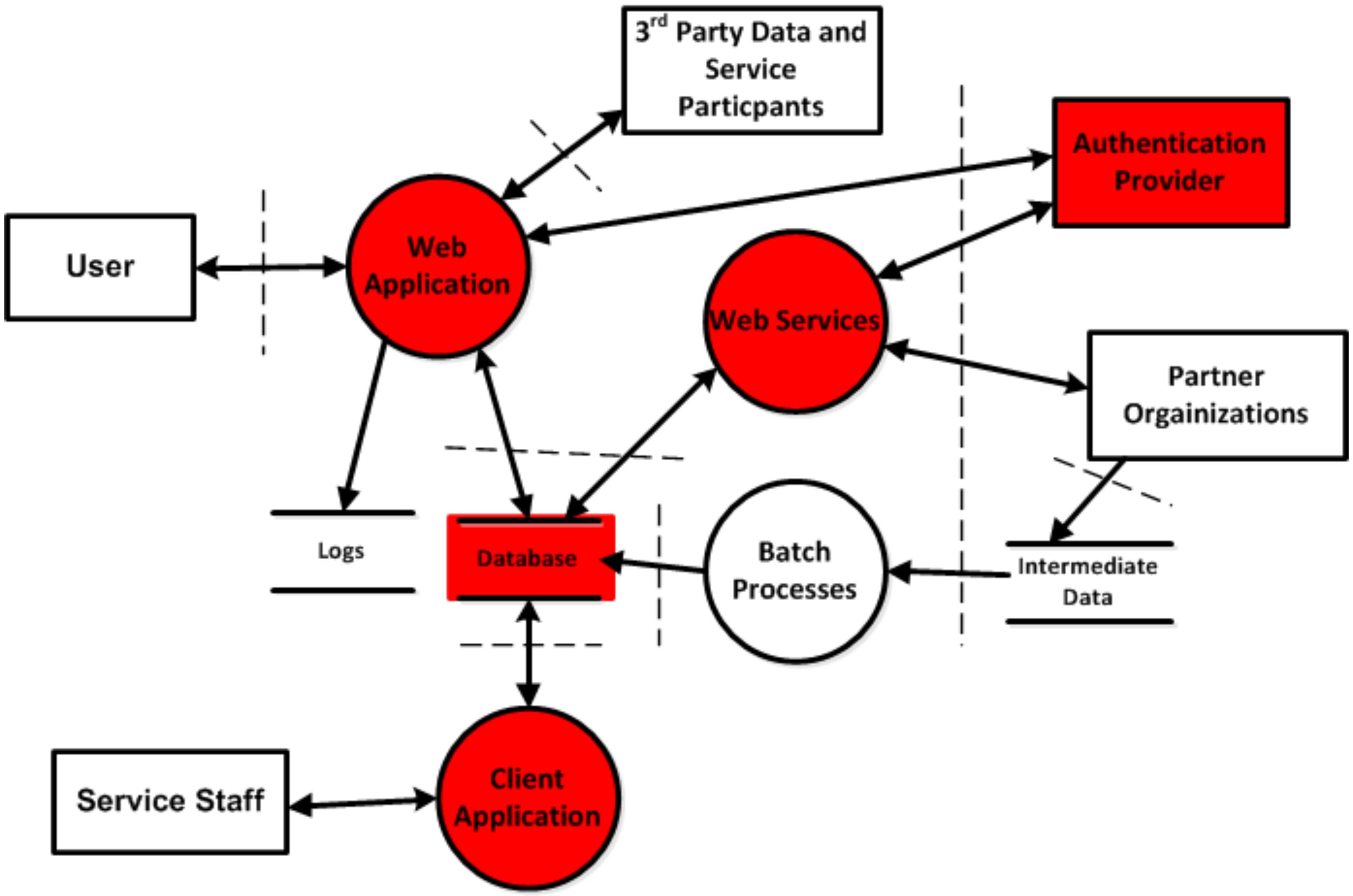


Information Disclosure: Identified Data Assets

- What are the touch points for sensitive data?
  - *Data Stores?*
  - *Applications?*
  - *Transport Layer?*
- How do these entities handle the data?
  - *In memory?*
  - *At rest?*

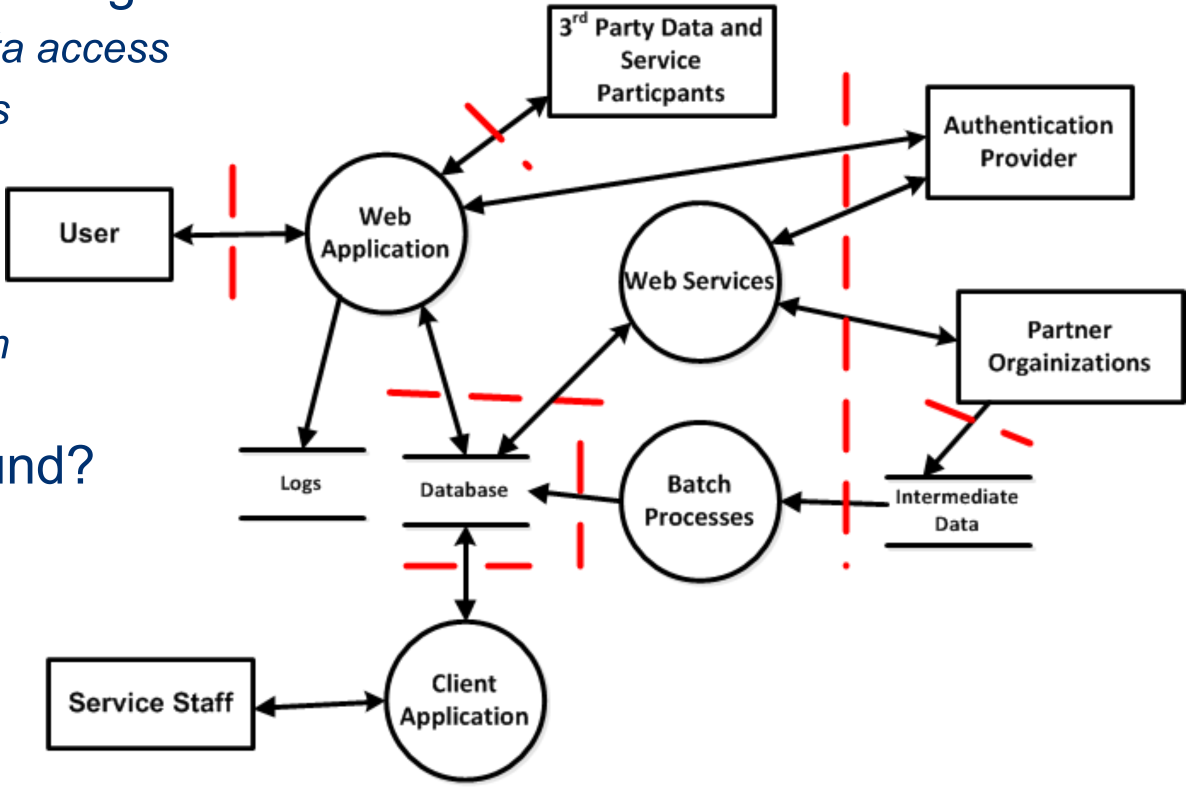


Denial of Service



Elevation of Privilege

- What entities manage privileges?
  - *Apps manage feature/data access*
  - *Apps maintain credentials*
  - *Network trusts app*
  - *Batch processes access file system*
  - *Batch processes maintain credentials*
- Are these privileges sound?
  - *Too permissive?*
  - *Prone to disclosure?*
  - *Poor operational management?*



Threat Modeling Activities: Functional

Functional Security Threats

- When can we do this?
  - *The team uses coding standards*
  - *The application is released or well into development*
- Functional Inputs
  - *Technology Stack*
  - *Implementation Standards*
- Functional Outputs

Functional Security

- Even organizations that track functional security do not often have it collated
  - *Authentication*
  - *Session Management*
  - *Input Validation*
  - *Data Protection*
  - *Error Handling*
  - *Etc.*
  - *I may or may not know what your coding standards are, but I want to know how you actually implement them*
- Abuse cases for each domain of functional security
  - *How would an attacker look for gaps?*
  - *How could the mechanism be abused or circumvented?*
- This does go over a lot of the assessment "baseline", but the value is in having it together

Functional Security Taxonomy

The OWASP Application Security Verification Standard is a good fit for this

1. **Security Architecture**
2. **Authentication**
3. **Session Management**
4. **Access Control**
5. **Input Validation**
6. **Output Encoding/Escaping**
7. **Cryptography**
8. **Error Handling and Logging**
9. **Data Protection**
10. **Communication Security**
11. **HTTP Security**
12. **Security Configuration**
13. **Malicious Code Search**
14. **Internal Security**

One important item NOT in ASVS

- **Least Privilege**

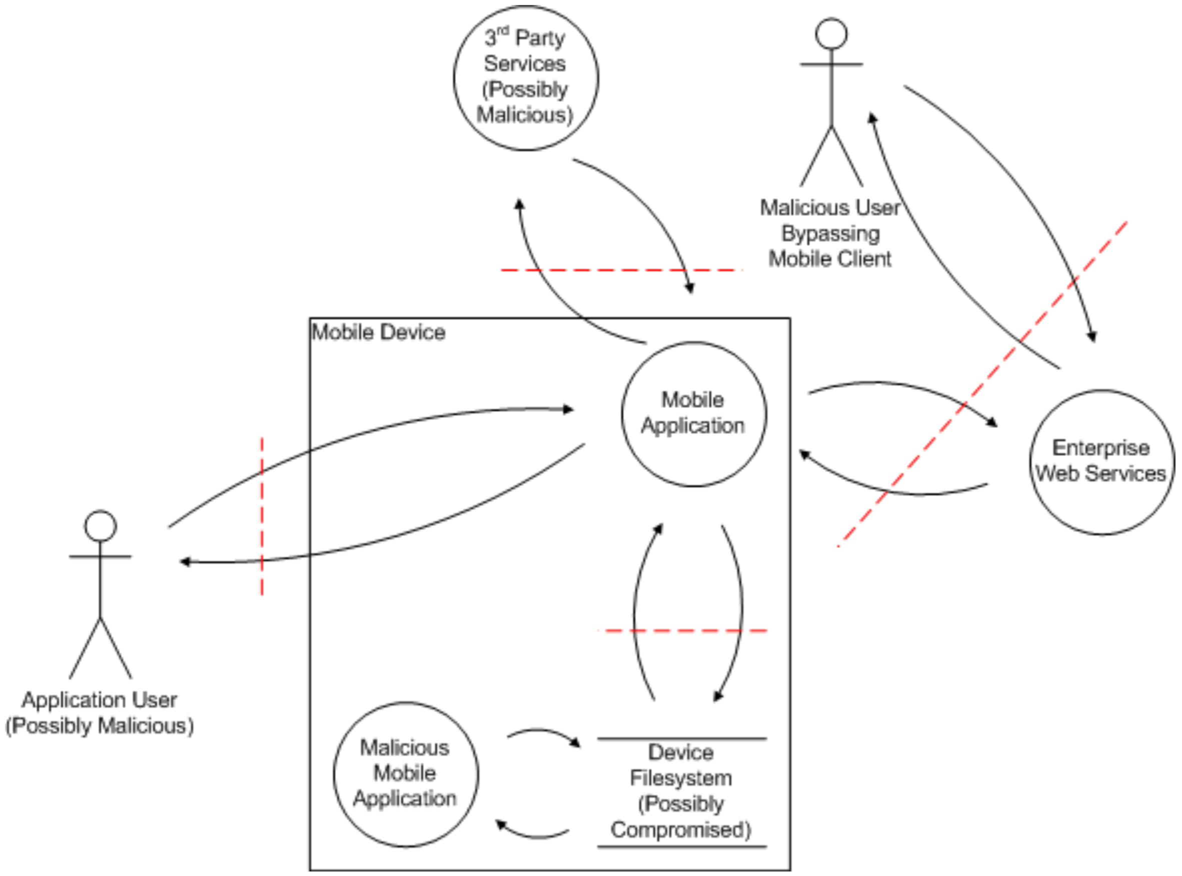
Threat Modeling Activities: Countermeasures

Countermeasures

- Do nothing
- Remove the feature
- Turn off the feature
- Warn the user
- Counter the threat with Operations
  - *Accountability*
  - *Separation of Duties*
- Counter the threat with Technology
  - *Change in Design*
  - *Change in Implementation*

- There is no “catch all” countermeasure

Generic Mobile Application Threat Model


3rd Party Services (Possibly Malicious)
Malicious User Bypassing Mobile Client
Mobile Device
Mobile Application
Enterprise Web Services
Application User (Possibly Malicious)
Malicious Mobile Application
Device Filesystem (Possibly Compromised)

Questions / Contact Information

Dan Cornell

Principal and CTO

dan@denimgroup.com

Twitter @danielcornell

(210) 572-4400

www.denimgroup.com

blog.denimgroup.com