

# Threat Modeling

# Overview

- Introduction
- Goals of Threat Modeling
- The approach
- Exercise
- Learning resources

# Threat Modeling Basics

- Who?
- What?
- When?
- Why?
- How?

# Who

- Building a threat model
  - Dev owns DFD (diagram)
  - Test owns ID threats (analyze)
  - PM owns overall process
- Customers for threat models
  - Your team
  - Other feature, product teams
  - Customers, via user education
  - 'External' QA resources like pen testers
  - Security Advisors

# What

- Reason about, document and discuss security in a structured way
- Threat model & document
  - The product as a whole
  - The security-relevant features
  - The attack surfaces
- Assurance that threat modeling has been done well

# Why Threat Model

- Produce software that's secure by design
  - Improve designs the same way we've improved code
- Because attackers think differently
  - Creator blindness/new perspective

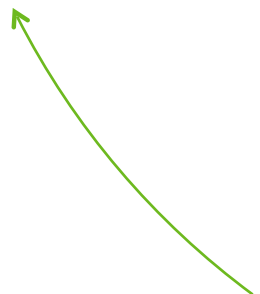
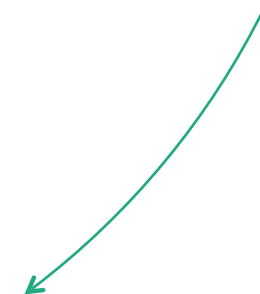
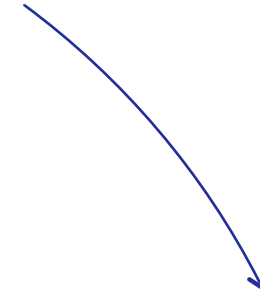
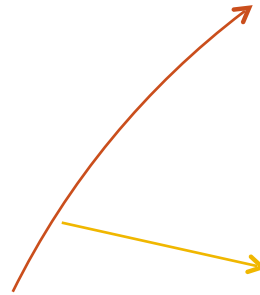
# Vision

- Diagram

- Validate

- Identify Threats

- Mitigate



# Vision

- Scenarios
  - Where do you expect the product to be used?
  - XBOX is different from Windows 7
  - xbox.com is different from XBOX
- Use cases/Use Stories
- Add security to scenarios, use cases
- Assurances/Guarantees
  - Structured way to think about “what are you telling customers about the product’s security?”



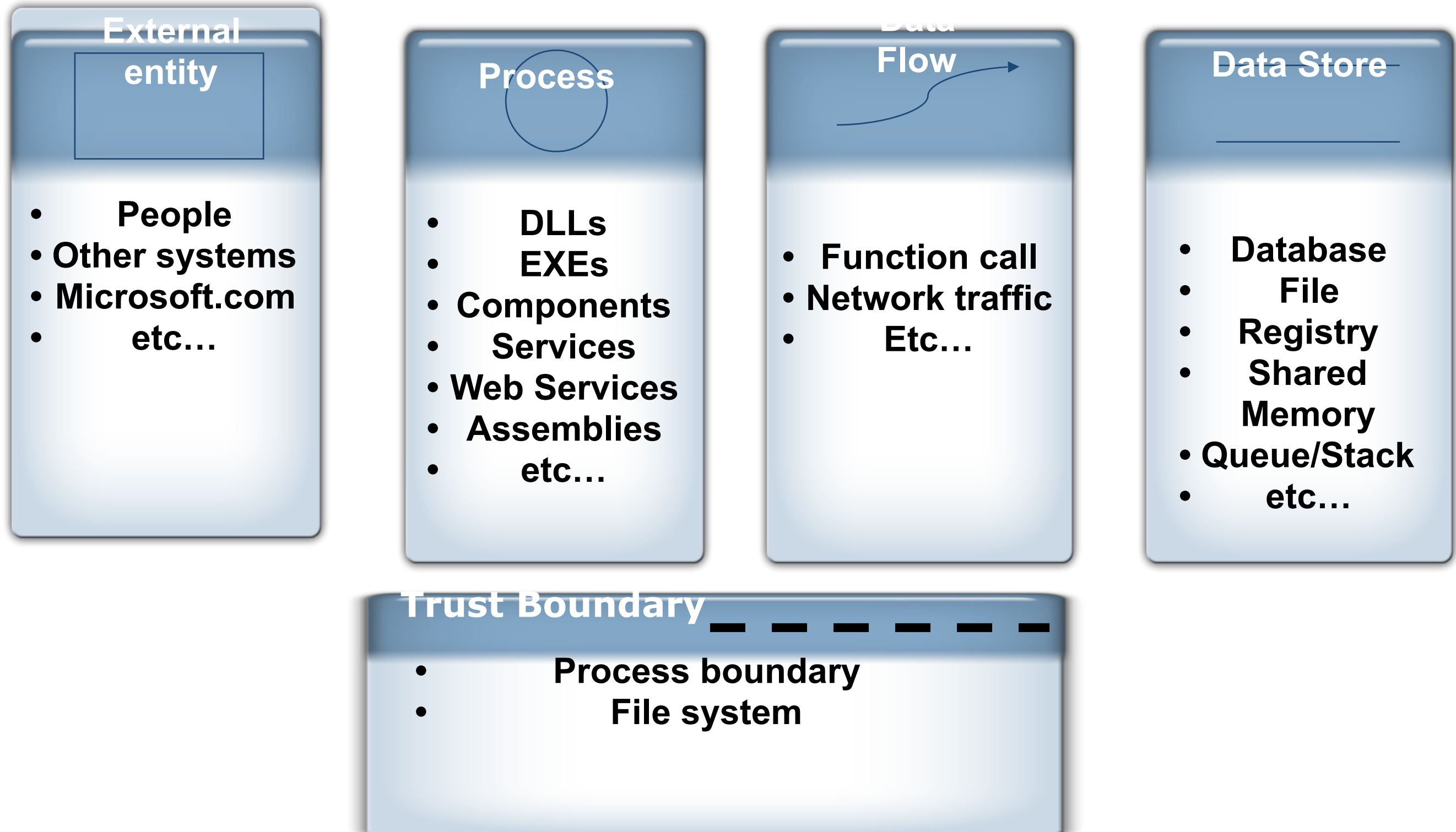
# How to Create Diagrams

- Go to the whiteboard
- Start with an overview which has:
  - A few external interactors (some use ‘actors’)
  - One or two processes
  - One or two data stores (maybe)
  - Data flows to connect them
- Check your work
  - Can you tell a story without edits?
  - Does it match reality?

# Diagramming

- Use DFDs (Data Flow Diagrams)
  - Include processes, data stores, data flows
  - Include trust boundaries
  - Diagrams per scenario may be helpful
- Update diagrams as product changes
- Enumerate assumptions, dependencies
- Number everything (if manual)

# Diagram Elements - Examples



# Diagrams: Trust Boundaries

- Add trust boundaries that intersect data flows
- Points/surfaces where an attacker can interject
  - Machine boundaries, privilege boundaries, integrity boundaries are examples of trust boundaries
  - Threads in a native process are often inside a trust boundary, because they share the same privs, rights, identifiers and access
- Processes talking across a network always have a trust boundary

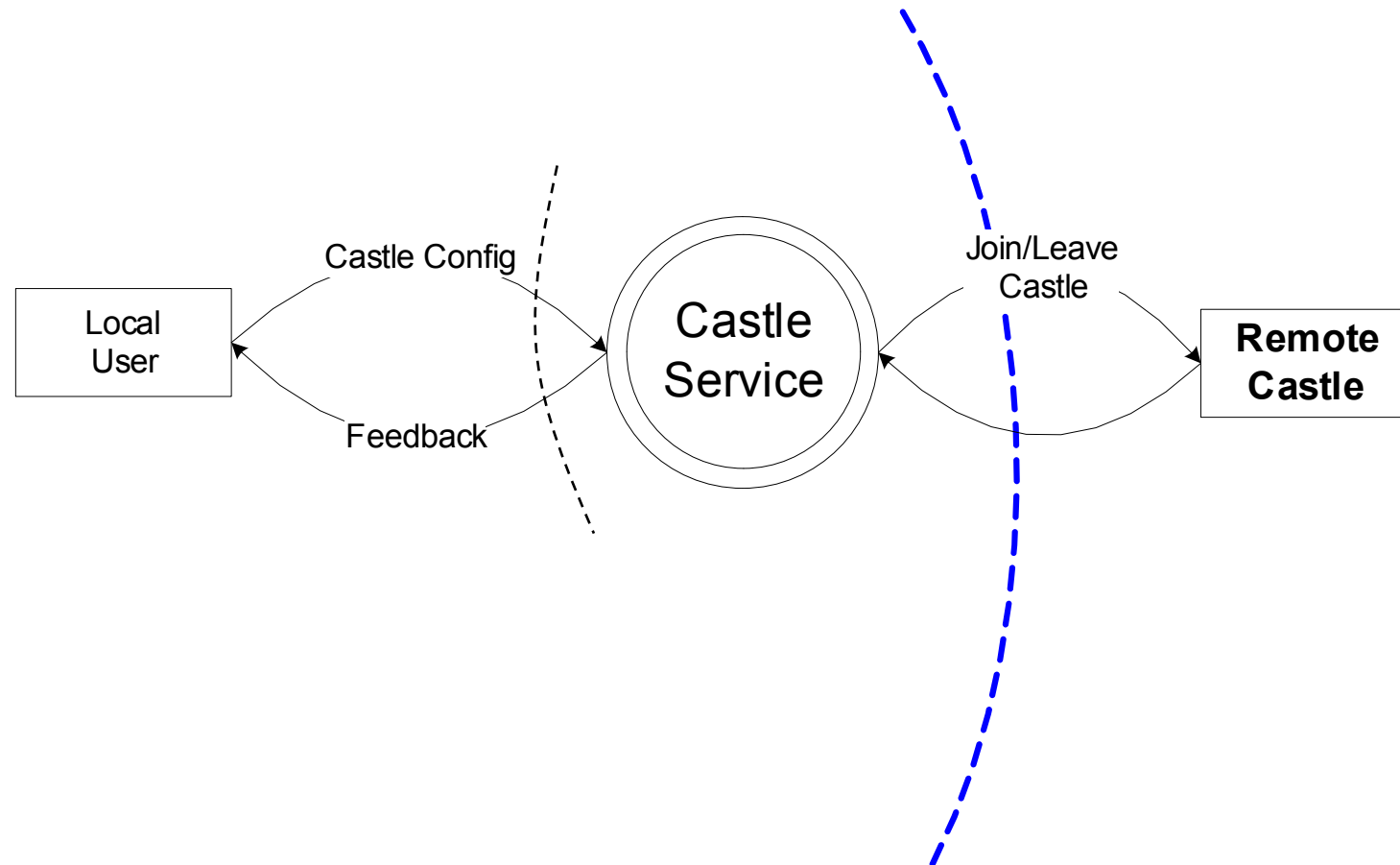
# Diagram Iteration

- Iterate over processes, data stores, and see where they need to be broken down
- How to know it “needs to be broken down?”
  - More detail is needed to explain security impact of the design
  - Object crosses a trust boundary
  - Words like “sometimes” and “also” indicate you have a combination of things that can be broken out
    - “Sometimes this datastore is used for X”...probably add a second datastore to the diagram

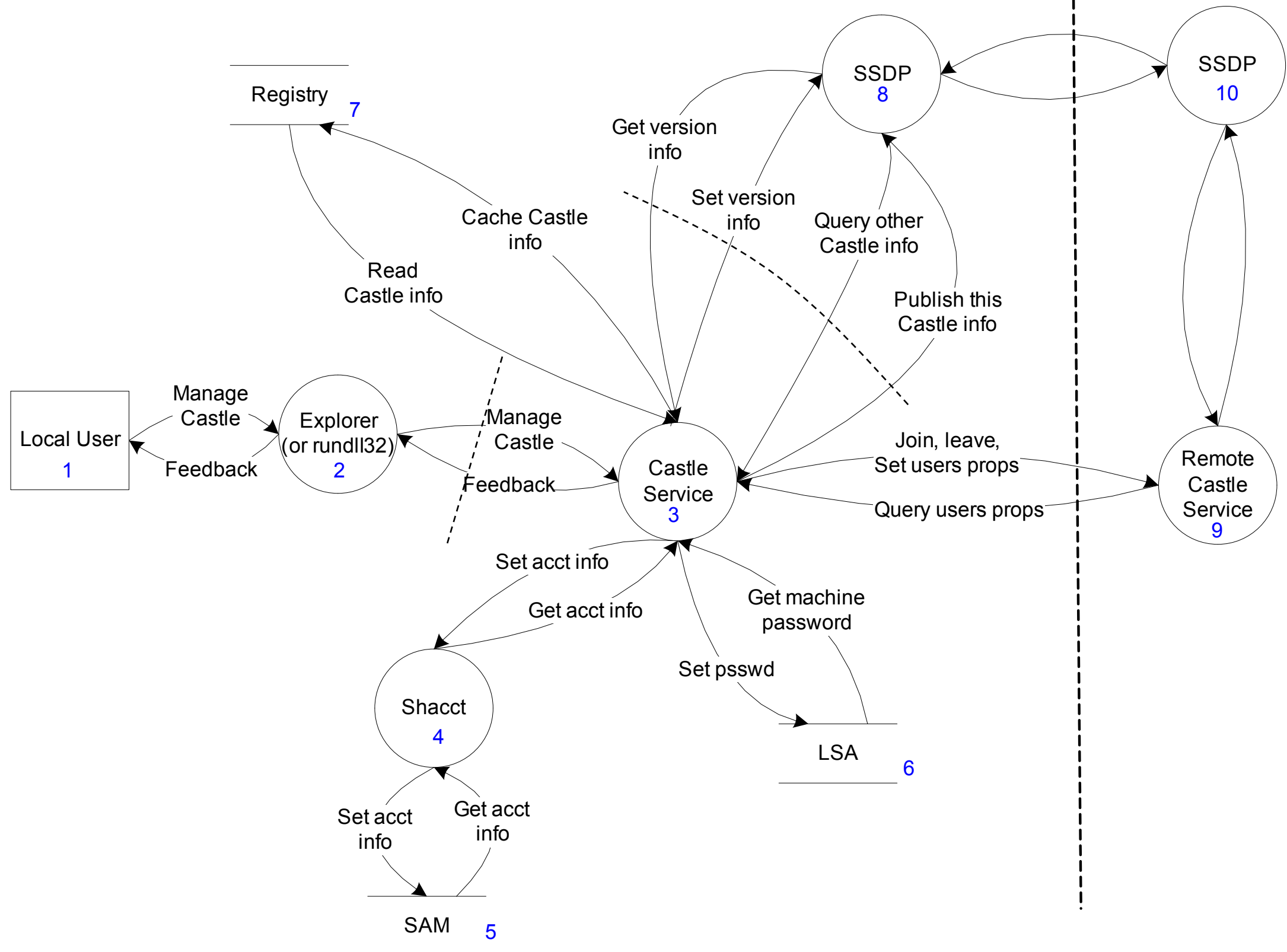
# Diagram layers

- Context Diagram
  - Very high-level; entire component / product / system
- Level 1 Diagram
  - High level; single feature / scenario
- Level 2 Diagram
  - Low level; detailed sub-components of features
- Level 3 Diagram
  - More detailed
  - Rare to need more layers, except in huge projects or when you're drawing more trust boundaries

# A Real Context Diagram (Castle)



# A Real Level-0 DFD (Castle)





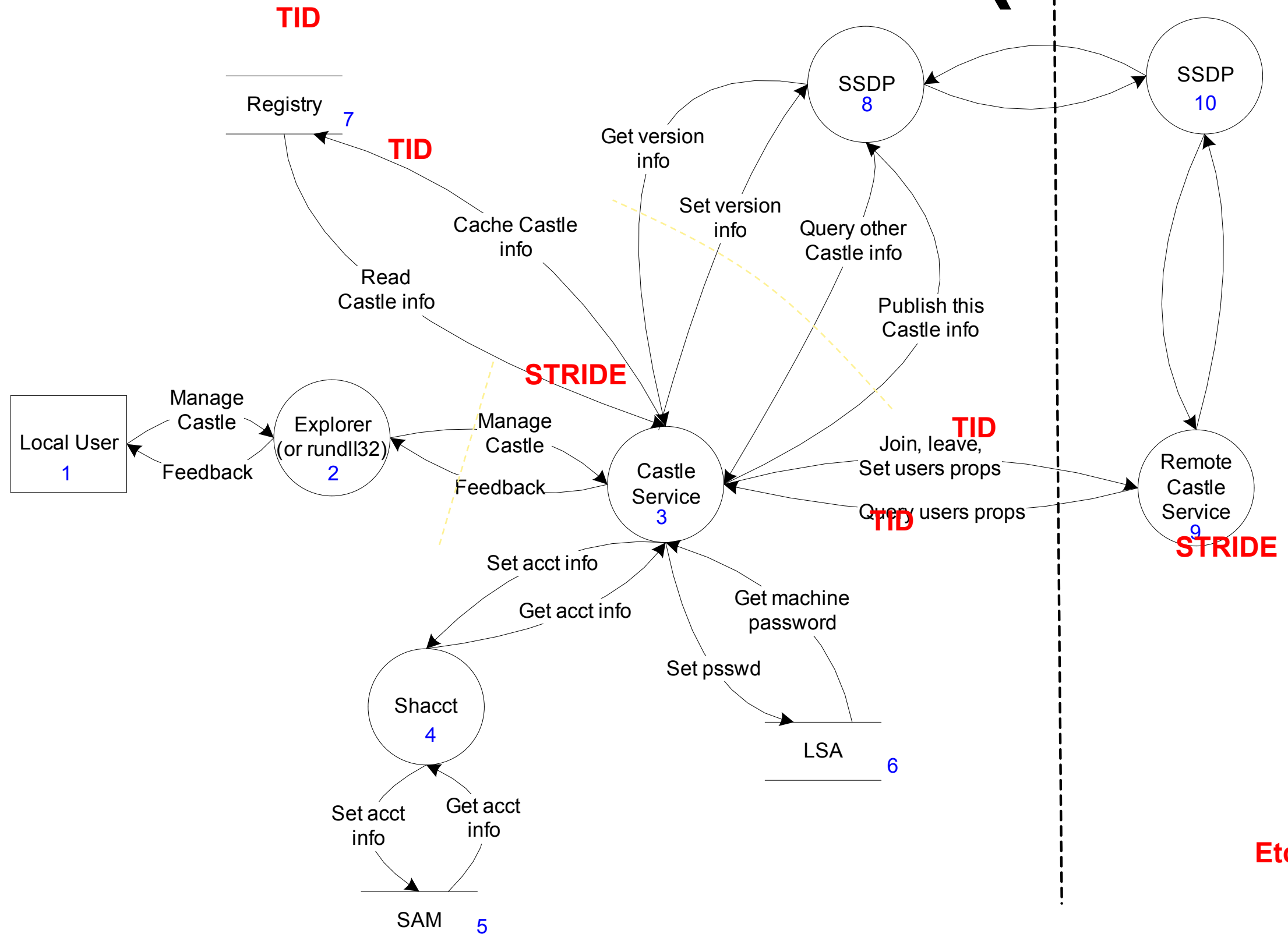
# Understanding the threats

Threat	Property	Definition	Example
<b>Spoofing</b>	Authentication	Impersonating something or someone else.	Pretending to be any of billg, xbox.com or a system update
<b>Tampering</b>	Integrity	Modifying data or code	Modifying a game config file on disk, or a packet as it traverses the network
<b>Repudiation</b>	Non-repudiation	Claiming to have not performed an action	“I didn’t cheat!”
<b>Information Disclosure</b>	Confidentiality	Exposing information to someone not authorized to see it	Reading key material from an app
<b>Denial of Service</b>	Availability	Deny or degrade service to users	Crashing the web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole
<b>Elevation of Privilege</b>	Authorization	Gain capabilities without proper authorization	Allowing a remote internet user to run commands is the classic example, but running kernel code from lower trust levels is also EoP

# Apply STRIDE Threats To Each Element

- For each thing on the diagram:
  - Apply relevant parts of STRIDE
  - External Entity: SR
  - Process: STRIDE
  - Data Store, Data Flow: TID
    - Data stores which are logs: TID+R
  - Data flow inside a process:
    - Don't worry about T,I or D
- Number things so you don't miss them

# A Real Level-0 DFD (Castle)



# Use the trust boundaries

- Trusted/high code reading from untrusted/low
  - Validate everything for specific uses
- High code writing to low
  - Make sure your errors don't give away too much

# Mitigation is the point of threat modeling

- Mitigation:
  - To address or alleviate a problem
- Protect customers
- Design secure software
- Why bother if you:
  - Create a great model
  - Identify lots of threats
  - Stop
- So find problems and fix them
  - File bugs to track them

# Mitigate

- Address each threat
- Four ways to address threats:
  - Redesign to eliminate
  - Apply standard mitigations
  - Invent new mitigations
    - Riskier
  - Accept vulnerability in design
- Address each threat!

# Standard Mitigations

Spoofing

Authentication

- To authenticate principals:
  - Basic & Digest authentication
  - LiveID authentication
  - Cookie authentication
  - Windows authentication (NTLM)
  - Kerberos authentication
  - PKI systems such as SSL/TLS and certificates
  - IPSec
  - Digitally signed packets
- To authenticate code or data:
  - Digital signatures
  - Message authentication codes
  - Hashes

Tampering

Integrity

- Windows Mandatory Integrity Controls
- ACLs
- Digital signatures
- Message Authentication Codes

Repudiation

Non Repudiation

- Strong Authentication
- Secure logging and auditing
- Digital Signatures
- Secure time stamps
- Trusted third parties

Information Disclosure

Confidentiality

- Encryption
- ACLS

Denial of Service

Availability

- ACLs
- Filtering
- Quotas
- Authorization
- High availability designs

Elevation of Privilege

Authorization

- ACLs
- Group or role membership
- Privilege ownership
- Permissions
- Input validation

# Inventing Mitigations is Hard

- Mitigations are an area of expertise like networking, databases, or cryptography
- Amateurs make mistakes, so do pros
- Mitigation failures will appear to work
  - Until an expert looks at them
  - We hope that expert will work for us
- When you need to invent mitigations, get expert help
  - We will try to talk you off the ledge 😊



# Validating Threat Models

- Validate the whole TM
  - Does diagram match final code?
  - Are threats enumerated?
  - Minimum: STRIDE per element that touches a trust boundary
  - Has Test reviewed the model?
    - Created appropriate test plans
    - Tester approach often finds issues with TM, or details
- Is each threat mitigated?
  - Are mitigations done right
- Did you check these before FSR?
  - Shipping will be more predictable

# Validate Quality of Threats & Mitigations

- Threats
  - Describe the attack
  - Describe the context
  - Describe the impact
- Mitigations:
  - Associate with a threat
  - Describe the mitigation(s)
  - File a bug
  - Fuzzing is a test tactic, not a mitigation

# Validate Information Captured

- Dependencies
  - What other code are you using?
  - What security functions are in that other code?
  - Are you sure?
- Assumptions
  - Things you note as you build the threat model
  - “HTTP.sys will protect us against SQL Injection”
  - “LPC will protect us from malformed messages”
  - CryptGenRandom will give us crypto-strong randomness

# Effective Threat Modeling Meetings

- Start with a DFD walkthrough
- Identify most interesting elements
  - Assets (if you identify any)
  - Entry points/trust boundaries
- Walk through STRIDE against those
- Threats that cross elements/recur
  - Consider library, redesigns

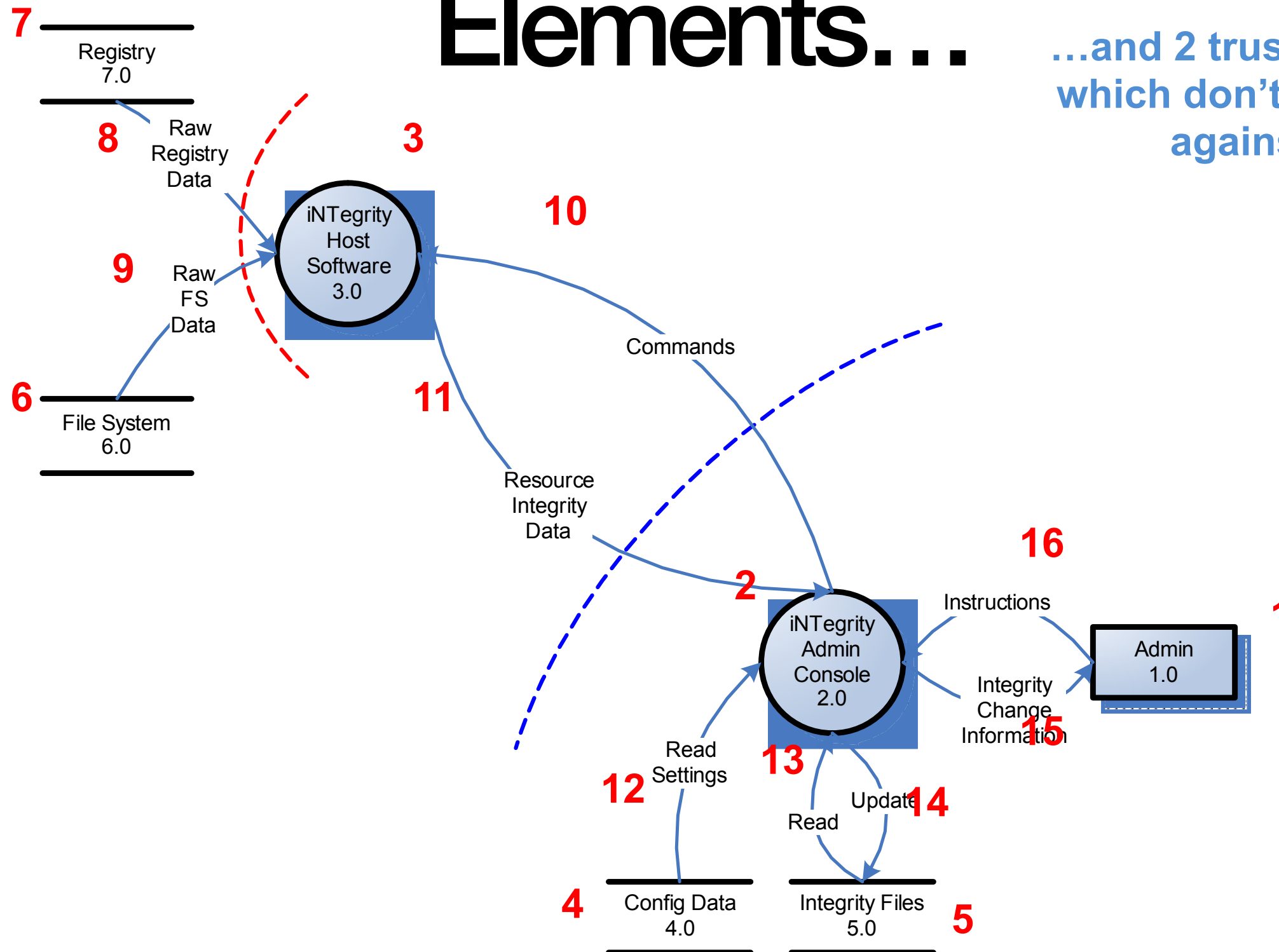
**PAUSE FOR QUESTIONS  
BEFORE EXERCISE**

# Exercise

- Work in teams to:
  - Identify all diagram elements
  - Identify threat types to each element
  - Identify at least 3 threats
  - Identify first order mitigations
-

# Identify all Elements: 16 Elements...

...and 2 trust boundaries,  
which don't have threats  
against them



# Identify Threats!

- Specific
- Understand threat and impact
- Identify 1st order mitigations



**End exercise**

# Call to Action

- Threat model your work!
  - Start early
  - Track changes
- Work with your Security Advisors!
- Talk to your “dependencies” about security assumptions
- Learn more
  - <http://blogs.msdn.com/sdl>

# Learning Resources

- MSDN Magazine
  - Uncover Security Design Flaws Using the STRIDE Approach <http://msdn.microsoft.com/msdnmag/issues/06/11/ThreatModeling/default.aspx>
  - <http://msdn.microsoft.com/en-us/magazine/cc700352.aspx>
  - Getting Started with the SDL TM Too
  - <http://msdn.microsoft.com/en-us/magazine/2009.01.securitybriefs.aspx>
- Lots more SDL: Training and Resources
  - <http://msdn.microsoft.com/en-us/security/cc448120.aspx>
- Books: lots of info which drove evolution of better processes