Threat Modeling

Why Make a “Good” Threat Model?

- Helps communicate prioritized issues to management
  - Given we have limited time and resources, which defenses should be built first?
- Helps developers track current and future security work
- Forms a great basis for a Penetration Testing Plan - what mitigations to verify?
- Documents what security assumptions are made about a system

Essential Parts of a Good Threat Model

- Unique Threat ID
  - for tracking, rows might be re-sorted
- Assets (what's at risk)
  - data classification, $$$, reputation
- Vulnerability
  - what is the weakness?
- Threat
  - how will the weakness be exploited?

- **Likelihood / Probability** (numeric)
  - odds that bad thing will happen
- **Impact / Consequences** (numeric)
  - how bad is bad thing?
- **Total Risk** = Likelihood * Impact

- Current Mitigations & Controls
  - *could impact the likelihood
- Future Mitigations & Controls
  - with engineering cost estimate to fix
- Attacker Sophistication / Threat Actor
  - no/low/high skilled, insider, nation-state*
- Attack Surface
  - network, environment, insider, pivot*

Optional:

- Metadata tracking categories
- Link to GUS story
- Link to GRC risk acceptance

Everyone Does the Same Risk Assessment

(did not make this up)

North American Public Avalanche Danger Scale

Avalanche danger is determined by the likelihood, size and distribution of avalanches.

Danger Level		Travel Advice	Likelihood of Avalanches	Avalanche Size and Distribution
5 Extreme	4 5	Avoid all avalanche terrain.	Natural and human-triggered avalanches certain.	Large to very large avalanches in many areas.
4 High	4 5	Very dangerous avalanche conditions. Travel in avalanche terrain not recommended.	Natural avalanches likely; human-triggered avalanches very likely.	Large avalanches in many areas; or very large avalanches in specific areas.
3 Considerable	3	Dangerous avalanche conditions. Careful snowpack evaluation, cautious route-finding and conservative decision-making essential.	Natural avalanches possible; human-triggered avalanches likely.	Small avalanches in many areas; or large avalanches in specific areas; or very large avalanches in isolated areas.
2 Moderate	2	Heightened avalanche conditions on specific terrain features. Evaluate snow and terrain carefully; identify features of concern.	Natural avalanches unlikely; human-triggered avalanches possible.	Small avalanches in specific areas; or large avalanches in isolated areas.
1 Low	1	Generally safe avalanche conditions. Watch for unstable snow on isolated terrain features.	Natural and human-triggered avalanches unlikely.	Small avalanches in isolated areas or extreme terrain.

Safe backcountry travel requires training and experience. You control your own risk by choosing where, when and how you travel.

No Rating		Insufficient information to establish avalanche danger rating. Check zone forecast for local information.

Essential Parts of a Good Threat Model

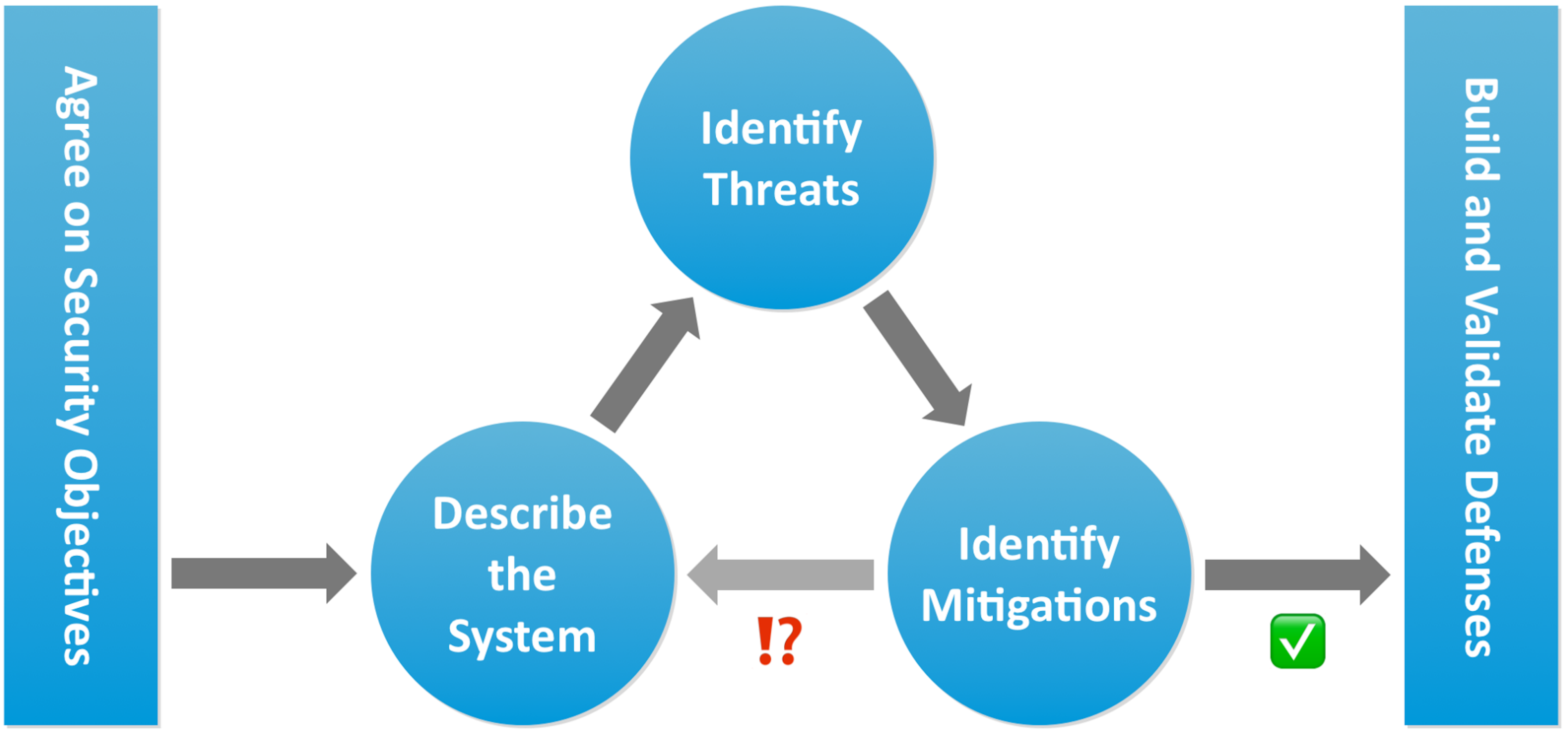
Do not let the perfect be the enemy of the good!

Write down every threat anyone thinks of. If it's unlikely, then set the Likelihood low and it will get sorted to the bottom and end up as a P2 (and no work will happen on it).
Just because you don't believe in zombies doesn't mean they're not coming...

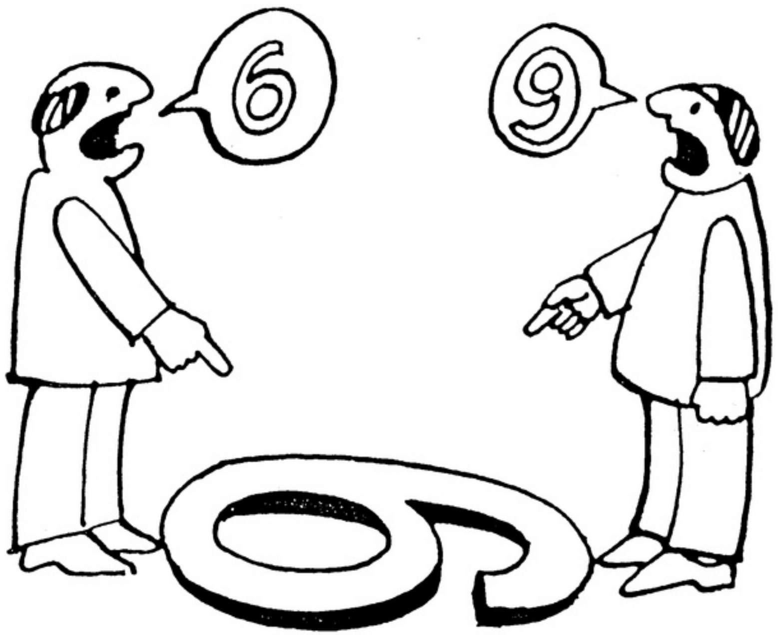
This saves other people from having to think of it all over again and re-open the debate about whether or not it should be on the model. This is a waste of time.

Threat models need to be **living documents**, so the expectation is that they will be periodically re-visited and updated. Doesn't have to be perfect the first pass, and it won't really ever be done.

Steps of Creative Threat Modeling



Use a threat identification methodology that works best for the system you are analyzing (use multiple if you can!)



How to Threat Model: Focus on what you are building

1. What are you building?

1. What can go wrong?

1. What are you going to do about it?

1. Check your work on 1-3

I’m not a security expert, why should I care?

1. Threat Modeling helps you get better at security!
2. It allows you to have constructive conversations with experts
3. It’s a structured way of figuring out if it’s OK to build something or change something already built
4. Risk assessment is a universal survival skill
5. Management can have a short attention span, so having a prioritized list colored by severity means they only need to read about three things before approving your budget for security work (it’s not SE, just efficiency…)

Process for Working With Teams

1. Initial kick-off meeting with stakeholders, explain threat modeling if needed and then go through a few rows with everyone to set the right **altitude** for threats (art != science).
   In the room:
   a. All the worriers from the dev team
   b. Security domain experts
   c. Maybe an architect?
   d. Best to keep the management and any optimists out, to save time
2. Send everyone home for a week or two and have them fill out all the threats they can think of (brainstorming-style, don't prematurely edit things out)
3. In a week or two, have another meeting with the same people and go over the combined list of everyone's threats. Check for duplicates and make sure everyone agrees on the final Risk score for each row.

***Its easy to get in the weeds !!!

Scenario

Scenario Introduction

The Product

We want to call the service “MachineSoup”. This will sit at the core of a few different products that require low-latency ingestion of data (e.g. for IoT), a **machine learning** processor to create the model, and then have the result queryable via this **innovative PostgreSQL module** that allows efficient vectorized spatial querying of the model. That might sound confusing, ***and it is***! We barely know what our Data Scientists are on to, but it sounds great and people really wants this. The exact processing that takes place will be a bit different between the different use-cases, so we want to design a secure way for Data Scientists to **upload code for these models**, to test out their models without stealing customer data, etc.

Scenario Introduction

The Environment

Assume that the environment upon which MachineSoup runs is basically a Linux-based OS that for the purposes of this exercise is “secure”.

Functional requirements

1/2

Producer sends raw data to the processing service.

- This service call is authenticated with a username and a service passphrase.
- The call also includes an orgId, modelId, and an optional model versioning label (modelId stays stable over time and the versioning label is described below)
- The orgId, modelId, and versioning label routes which actual processing service the raw data goes to.

The Processing Service will do some convolutional data-sciency stuff on the data then update the specified model in the special postgres database.

- The code run is based on the orgId, modelId and the versioning label
- Batches of raw data are stored in the postgres database as BLOB for later data-science efforts.
- BLOBs aren't indexed so can't be queried, just retrieved in relation to some timestamp, orgId, and modelId

Functional requirements

2/2

Consumers will use the query API to ask for the results.

- The Consumers are authenticated by a reusable component and the orgId and allowable modelIds + labels are known.
- If the consumer is now allowed to query a specific modelId or modelId+label, the query is denied.

Consumers with special privileges are allowed to “promote” a specific modelId + label to be the default when no label is specified for processing or querying.

- This is analogous to promoting some “staging” system into “production”.

The postgres database is queried by the query service on behalf of that end user.

- The results might be paginated/streamed via some in-memory caching system.

We need to be able to occasionally ship compiled C++ code for the PostgreSQL plugin that does the indexing.

Data Science requirements

Data scientists will want to tune the models on behalf of a customer, and so they need to be able to ship new code to the processing service

Data scientists can design new models by replaying the raw data (stored in BLOBs). This staged model is identified by modelId and mandatory label.

The Data scientist can use special credentials to access that staged model via the Consumer query API

Data scientists can stage these new models with a specific modelId for the Consumer to do their own testing

- The Data Scientists first upload a staged bit of code for the model then run code a job that replays the raw data from the BLOB

Legal requirements

Data Scientists are able to look at the raw data -- the ones stored in BLOBs for debugging purposes -- and test out models

Data Scientists cannot combine models or raw data in this type of system

Once entered, the raw data cannot leave the MachineSoup network environment, especially important to not have this data on laptops/workstations.

We need to send the customers a report of when exactly the data scientists access the data and there may be further regulation on how long sessions can be left idle.

Agreeing on Security Objectives

Agree on Security Objectives

It's can be as simple as this conversation:

What are you building?	*A blog server*
What types of assets do you need to protect?	*The blog posts, author credentials*
Are there any existing security defenses you'll be re-using?	*Yes! Our prod network, our auto-patched operating system, and our centralized logging*

Agree on Security Objectives

Can we agree that our threat actors are ***hacktivists*** and ***competing companies***, but **not** nation states?

Sure, that sounds reasonable. What about these cyborg gundam dolphins I've been hearing about?

... no, but maybe some day

**phew* alright!*

Data Classification

We have a common Data Classification standard that you can use (in low-to-high order):

- **Public** = Data that is already exposed to the public OR intended to be viewed publicly
- **Internal** = Data for *all* Salesforce employees (FTEs and contractors)
- **Confidential** = Data for a *subset* of Salesforce employees that *usually isn't* restricted by law, regulation, or company MSA. Might be under NDA. E.g., data *about* customers (including email address), employee compensation
- **Restricted** = Data for a *subset* of Salesforce employees that ***is*** restricted by law, regulation, NDA, or company MSA, e.g., PII, PCI, PHI, etc.
- **Mission Critical** = Data that is critical to the survivability and success of Salesforce, including ***Customer Data***, production data, auth data, encryption secrets, etc.

Data Classification Quick-Reference Table

Classification	Short Definition	Examples	Risk Consideration
Public	Data that is already exposed to the public OR is meant to be viewed by the public	Blog, WWW, Marketing	Reputation, Competitive
Internal	Data that is meant to be viewed by *anybody* at Salesforce	Intranet, Wiki, Org Data, Employee Lists, Internal Comms	+ Social Eng, Threat Persistence
Confidential	*The above* BUT ONLY specific roles or people.	Sales Data, Employee Compensation, Data **about** Customers (incl. email addr.)	+ Significant Security Breach
Restricted	*The above* PLUS is restricted by law, regulation, NDA, or our MSA.	PII, PHI, PCI, Financials, Pre-deal M&A, Regulated, SOX, Security Vulns	+ Breach Size, Legal / Regulatory
Mission Critical	Data that is critical to the survivability and success of Salesforce	**Customer Data**, Production Data, IT & Production Secrets	+ Company Dies

Before we start...

Begin a new collaborative document, e.g., Google Docs, Quip

Everyone being able to type concurrently is imperative!

Durable information (i.e., not totally on a whiteboard) is also important!

Success Theme: Ensure *everyone you want to collaborate* has a seed to grow information from, permission to add to that information & verified access

Have four sections: Security objectives, data-flow diagram, threats/mitigations, and build/test strategy

For capturing threats/mitigations, use this starter table:

Location	Threat	Mitigation	Notes
[1] end-user to query api	Plaintext can be sniffed, altered, etc. while in transit from the end user	HTTPS for this connection	LoadBalancer probably does this already

Exercise: Agree on Security Objectives

Example Security Objectives

1/2

Most valuable piece of data is likely User Data (Mission Critical), with the potential for storing PII (Restricted), so we want to protect that from loss or modification. PCI data is out of scope.

Threat actors: hacktivists, organized crime, etc. (i.e., not nation-state)

There’s multi-tenancy risks.

Authentication sounds non-straightforward, let’s also focus there.

The database is using a custom component.

Example Security Objectives

2/2

The Data Science stuff has caught the attention of Legal, so there’s a lot of value in keeping that workflow safe.

The Operating System and/or Container layer and below are assumed to be secured by the infrastructure team.

The build system itself is a black box for this exercise.

Describe the System

Describe the System

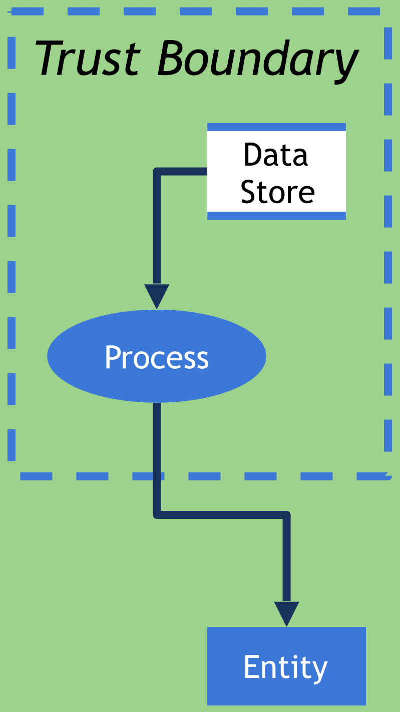
?

Recap: Data-Flow Diagrams

The users, actors, and related systems that data flows ***from*** or ***to***.

The conceptual processes that act on flows of data.

Where data "rests" (but not to die).



Collects components to the left in zones of equivalent trust.

Arrows denote direction that data is flowing. Add a label if it's ambiguous what type.

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

Trust Boundaries are Hard!

Where’s a good place to put the trust boundary? It depends.

Some good examples:

- Layered tier architecture boundaries
- Where you’d expect authentication, authorization, or “gating”
- Privilege levels, e.g., kernel vs. userspace, admin vs. regular user
- Identities, e.g., producer vs. consumer
- TPM vs. CPU vs. network
- Trusted network vs. untrusted network (e.g., the internet)

Example DFD

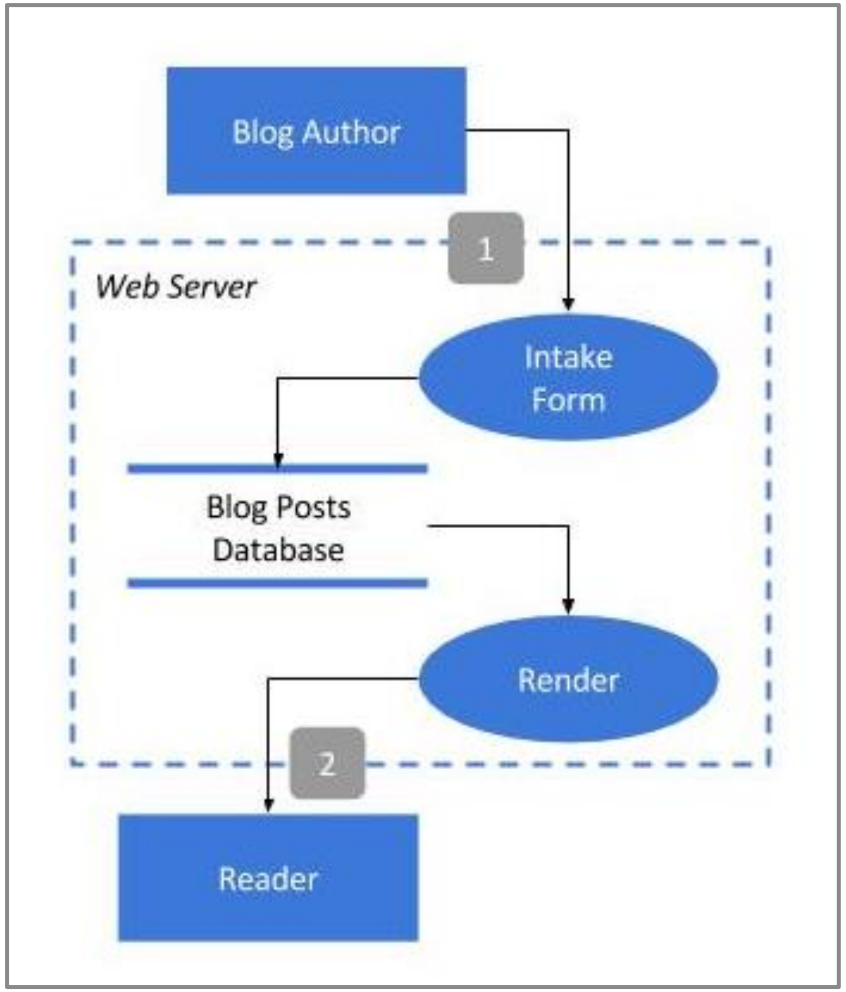

Blog Author
1
Web Server
Intake
Form
Blog Posts
Database
Render
2
Reader

Diagram Elements - Examples

External entity

- People
- Other systems
- Microsoft.com
- etc…

Process

- DLLs
- EXEs
- Components
- Services
- Web Services
- Assemblies
- etc…

Data Flow

- Function call
- Network traffic
- Etc…

Data Store

- Database
- File
- Registry
- Shared Memory
- Queue/Stack
- etc…

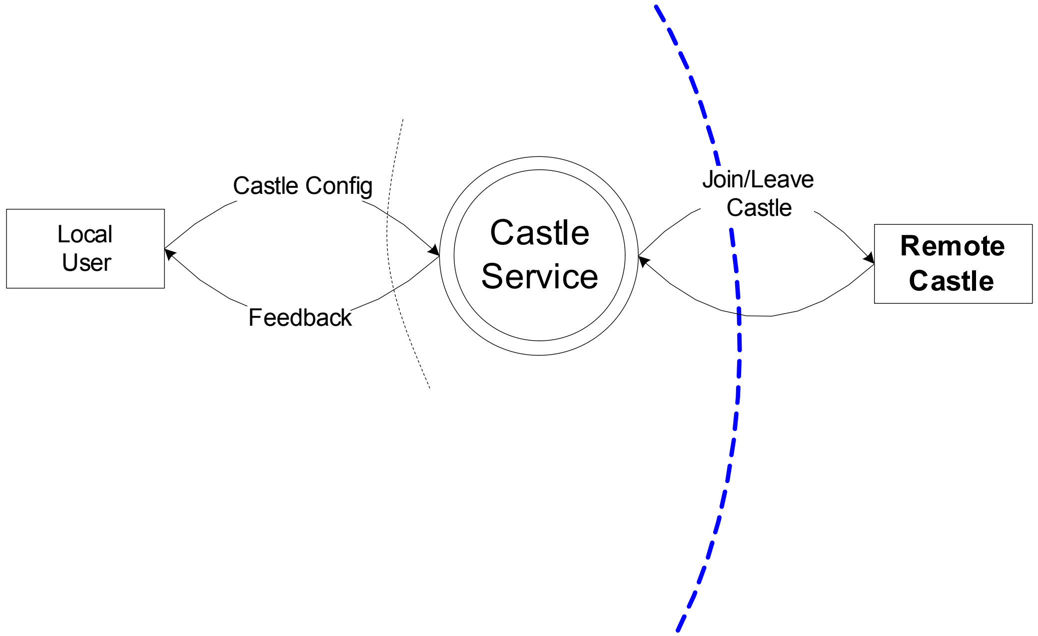
Trust Boundary

- Process boundary
- File system

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)

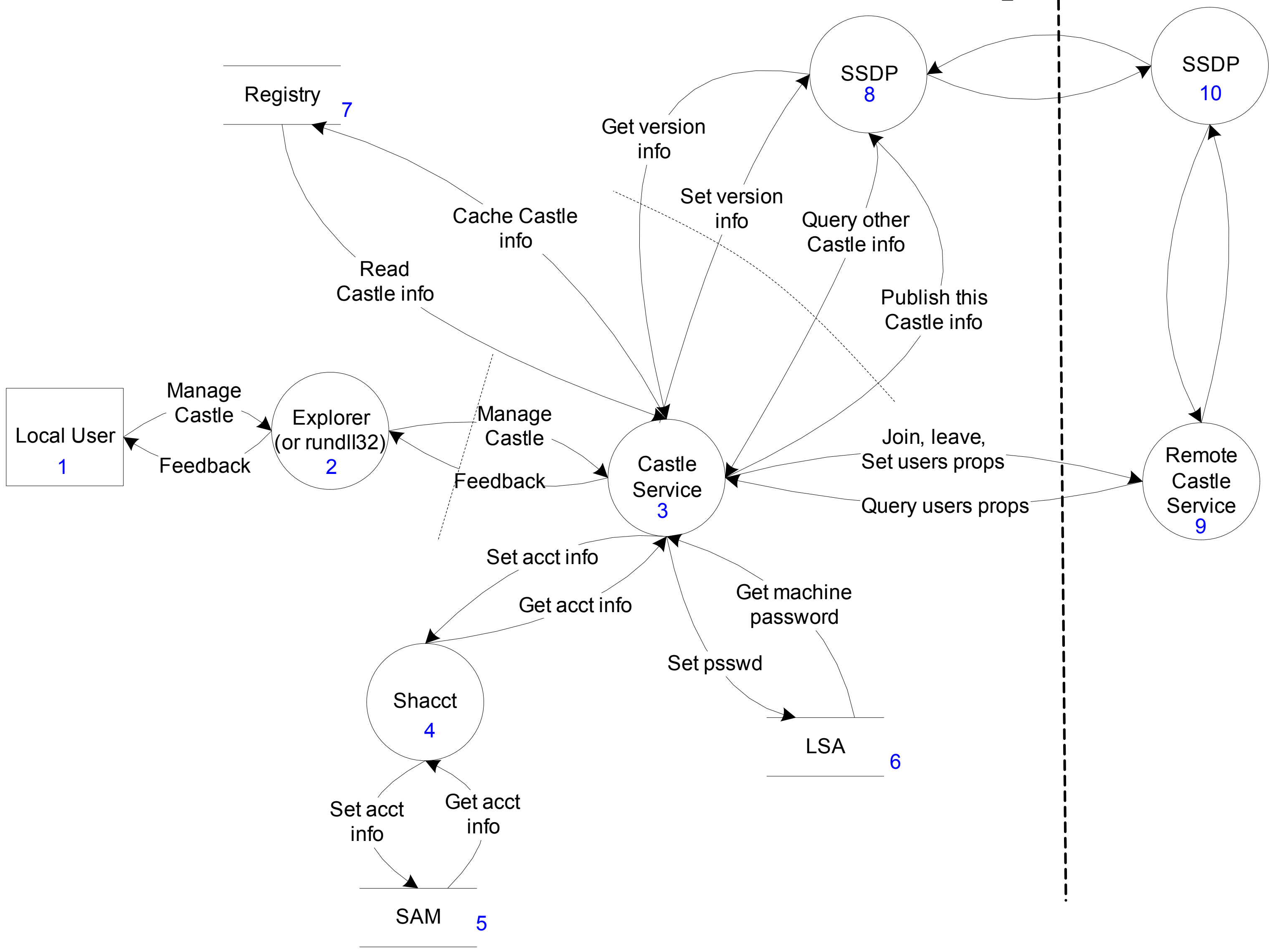


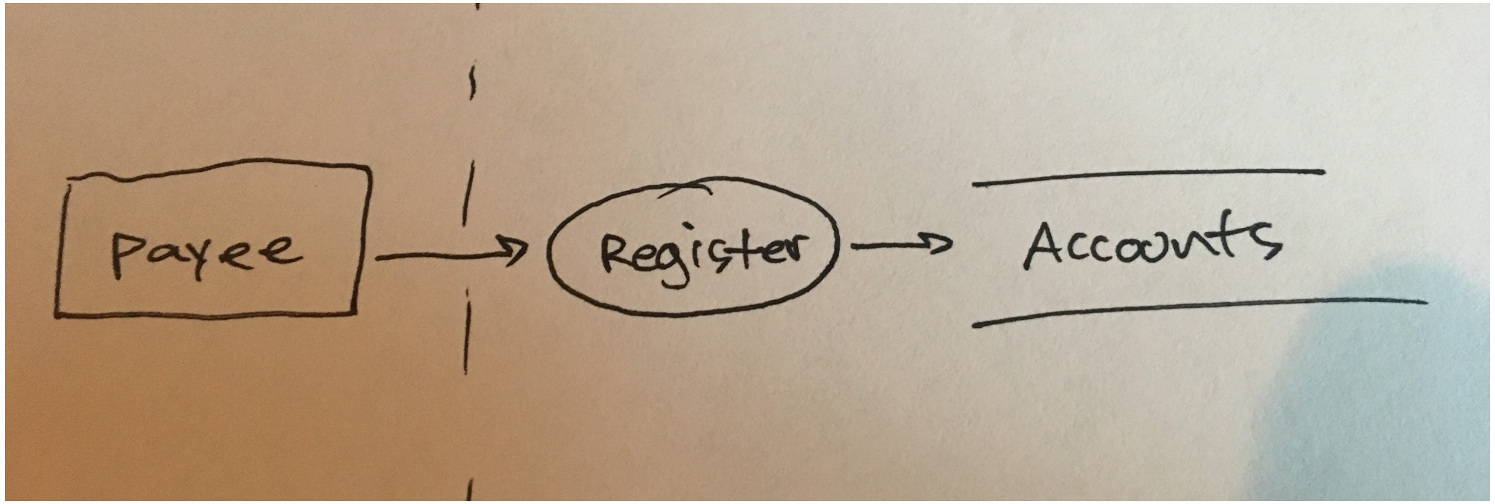
Diagram Iteration

- 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

Drawing By Hand?

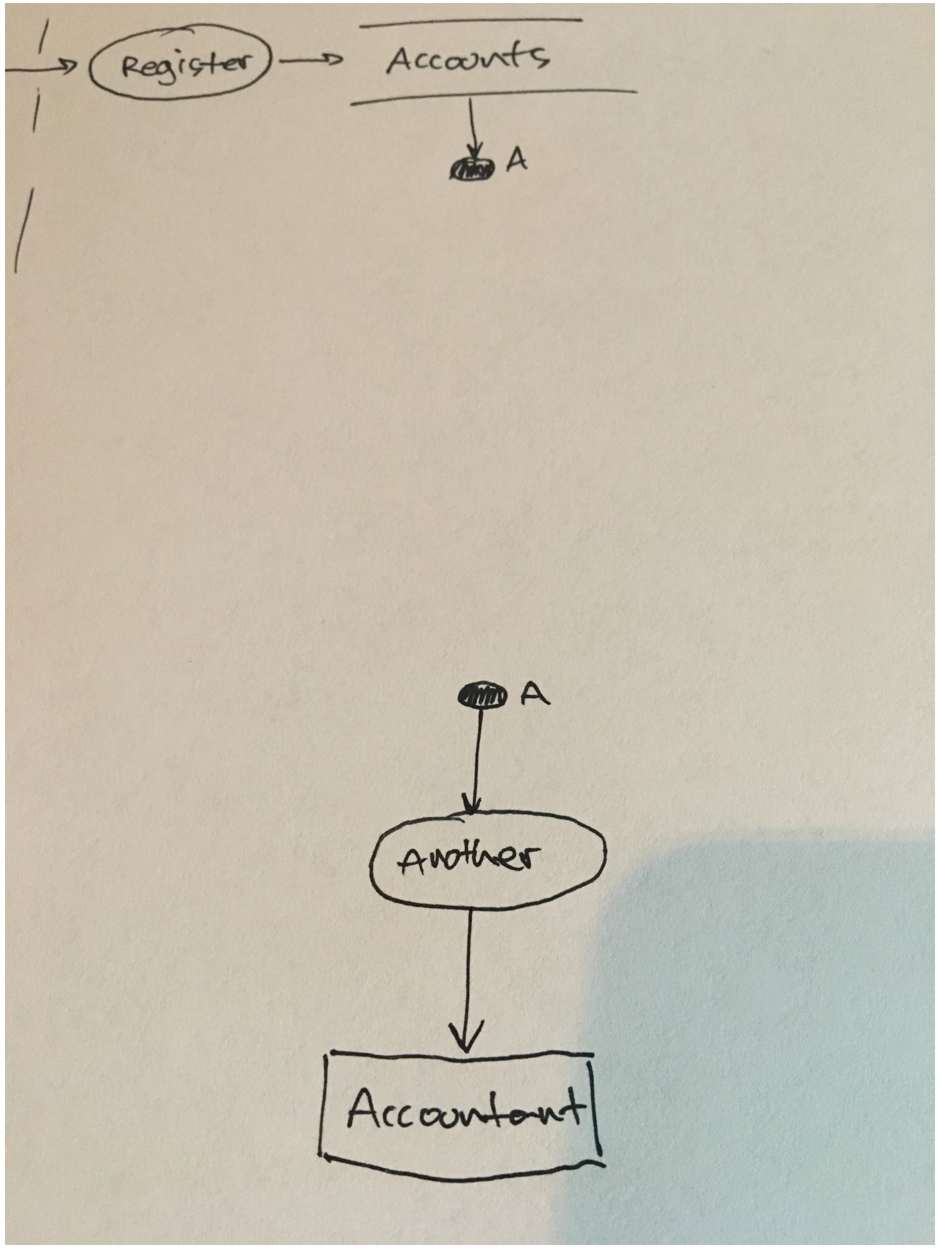
Use similar shapes

Try to write clearly!

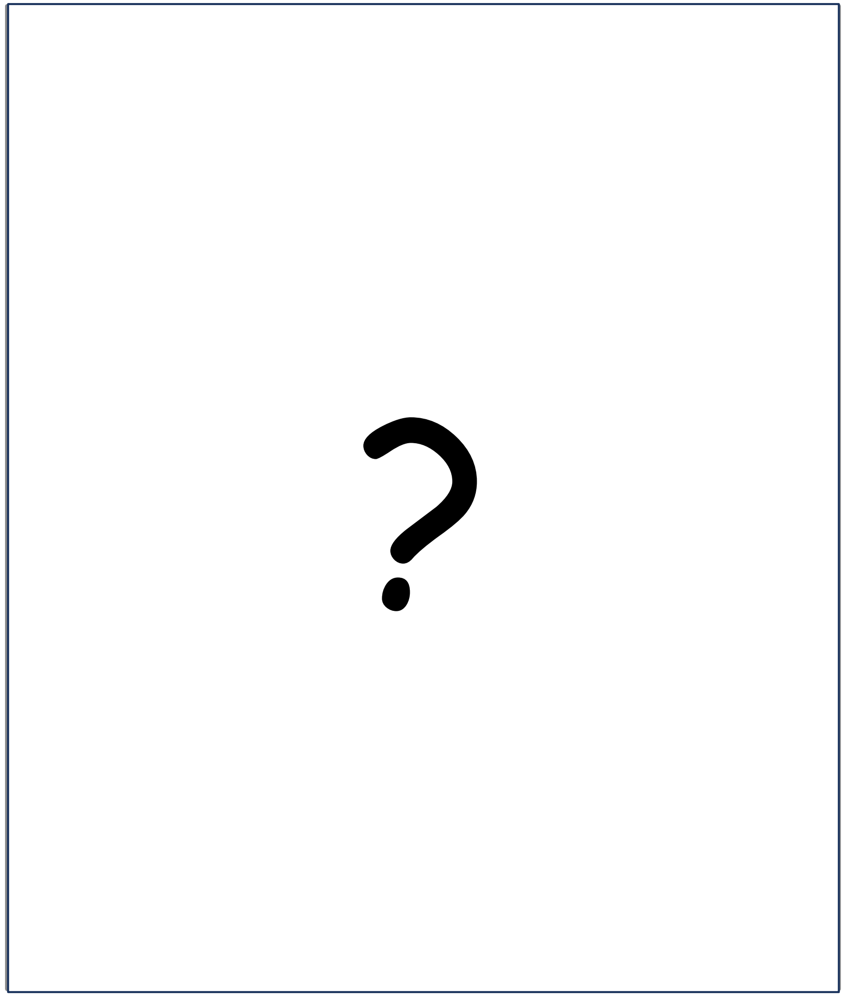


Drawing By Hand?

Pro Tip: use “wormholes” for connectors to route data-flows around other parts of the diagram



Where to start?

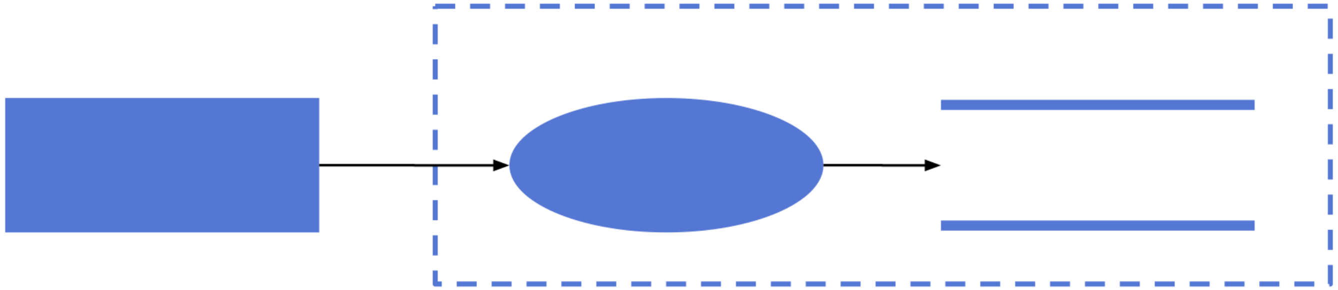


The "Seed" DFD

Avoid the Blank Canvas problem with a seed DFD

Identify one of each:

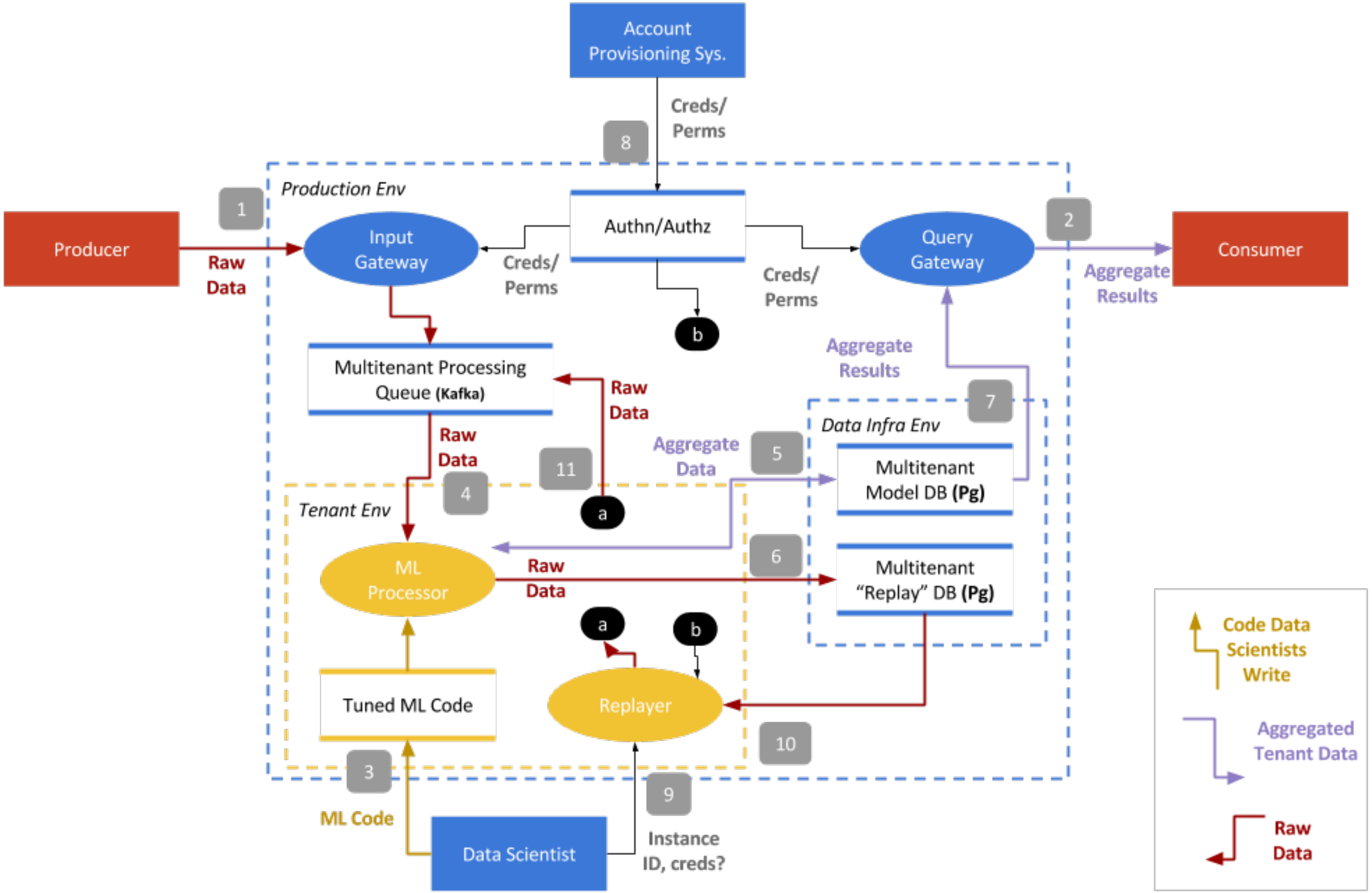
- Entity
- Process
- Data Store
- Trust Boundary



When is a DFD Complete?

- ✓ Are Processes and Entities are all named?
- ✓ Overall, is this an accurate and complete system representation?
- ✓ Have you identified all the Trust Boundaries?
- ✓ Does all data flowing into a Data Store flow out again? Is all produced data eventually consumed?
  - ✓ Are there any incomplete flows? Either complete or remove them.
  - ✓ Do flows go out of scope? Just note the other system as an Entity.
- ✓ Have the appropriate flow arrows been added?
- ✓ Do Interaction Points have labels?

Exercise: Drawing Data-flow Diagrams


Account Provisioning Sys.
Creds/ Perms
8
Production Env
1
Producer
Raw Data
Input Gateway
Creds/ Perms
Authn/Authz
Creds/ Perms
Query Gateway
2
Consumer
Aggregate Results
b
Multitenant Processing Queue (Kafka)
Raw Data
Aggregate Results
7
Data Infra Env
Raw Data
Aggregate Data
5
11
Multitenant Model DB (Pg)
4
a
Tenant Env
6
ML Processor
Raw Data
Multitenant "Replay" DB (Pg)
a
b
Tuned ML Code
Replayer
10
3
9
ML Code
Data Scientist
Instance ID, creds?
Code Data Scientists Write
Aggregated Tenant Data
Raw Data

Analyzing for Threats and Mitigations

Analyze For Threats

Analyze how **Entities**, **Processes**, and **Data Stores** interact

- To start, you may wish to use our collection of "Common Attacks" on the security wiki, OWASP e-Commerce Cornucopia, or the EoP deck from Microsoft
- When you get good at this, use the TIDES or STRIDE mnemonic

Pay special attention to where the **Data-Flows** go across **Trust Boundaries**

Write down all possible attacks (even if you think it might not be important)

Identify Mitigations

Review all the identified threats and figure out a way to mitigate them

Analyze *all the things*; leave no stone unturned

If there's too many problems, maybe it's time to iterate and try another design

Does every threat need to be 100% mitigated?

It depends...

If there's something impossible or impractical to fix, it's **residual risk**

Threat modeling enhances risk acceptance; it affords for **informed** and **rational** instead of *misinformed* and *irrational* decision making!

Pro-tip: Take A Break

Stuck thinking of threats? **Take a break!**

- Take everyone's mind off it for a while
- Schedule a date & time to reconvene

Plan your initial threat modeling session accordingly to allow space for this to happen.

Can’t think of any attacks?

- Security Cards from University of Washington

https://securitycards.cs.washington.edu/

The **Security Cards** encourage you to think broadly and creatively about computer security threats. Explore with 42 cards along 4 dimensions (suits):

HUMAN IMPACT

ADVERSARY'S MOTIVATIONS

ADVERSARY'S RESOURCES

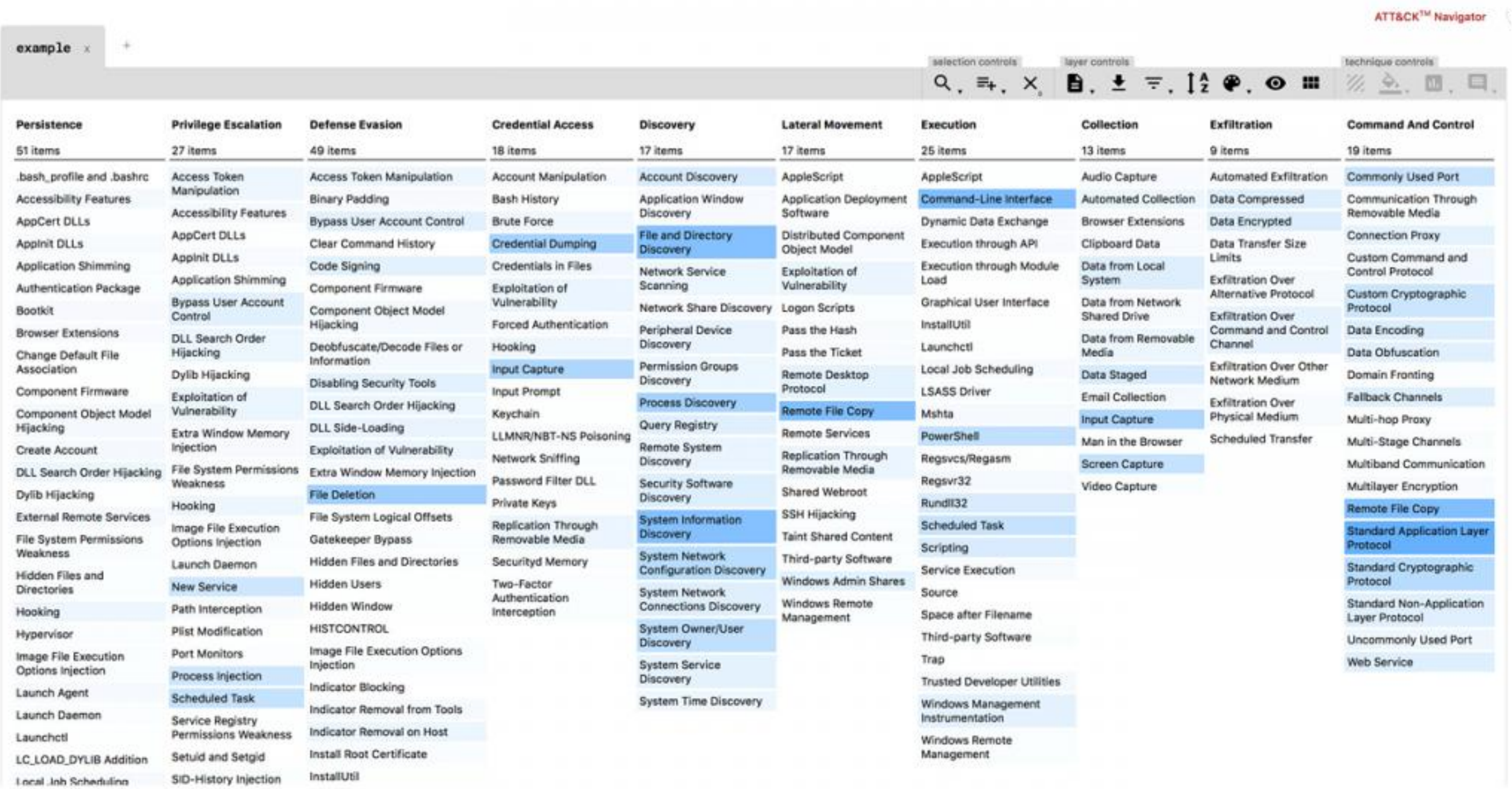
ADVERSARY'S METHODS

Use the included templates to make your own custom card.

Can’t think of any attacks?

- Mitre Attack Navigator - https://github.com/mitre/attack-navigator

- Mitre Attack Matrix - https://attack.mitre.org/wiki/Main_Page



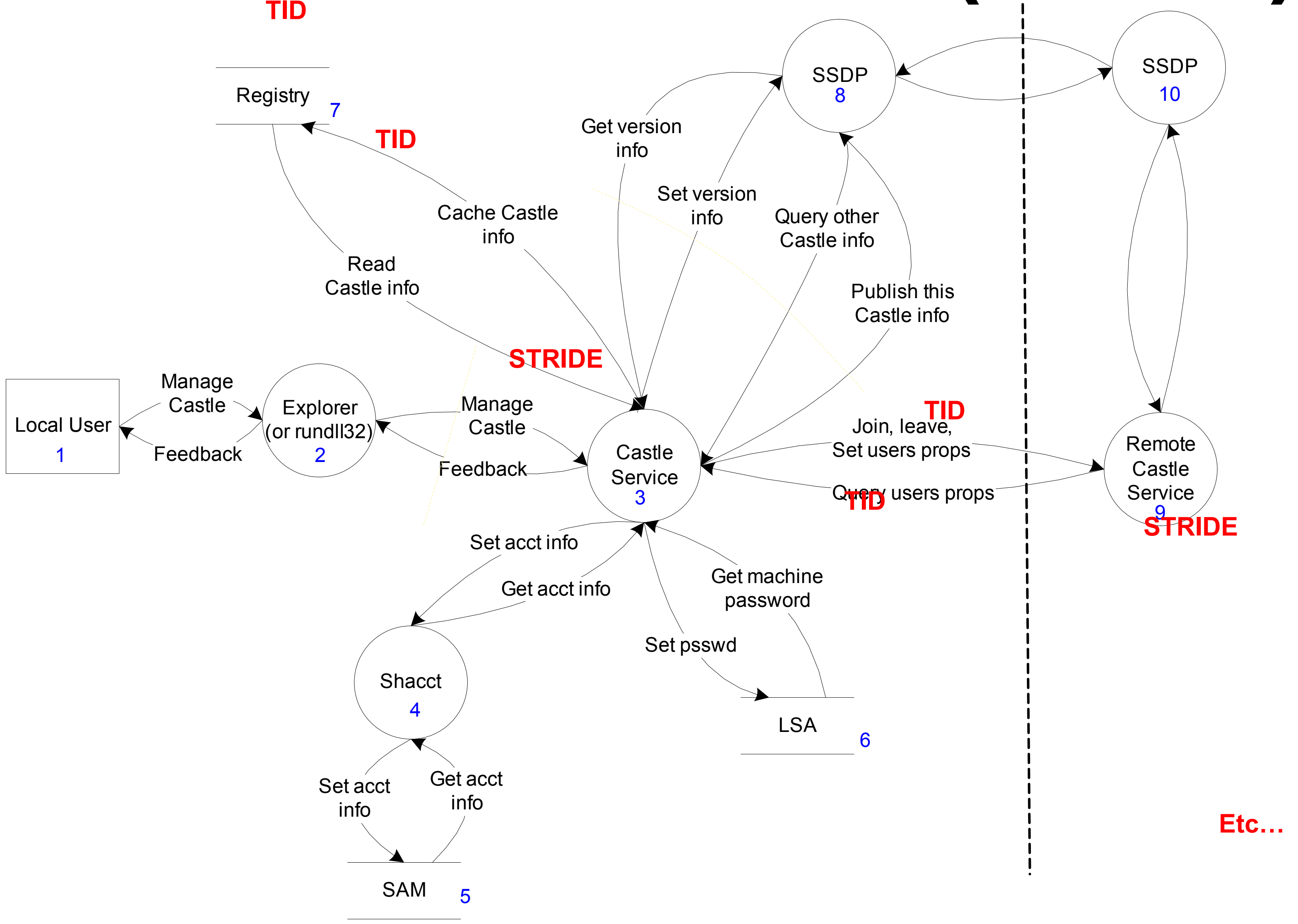
Understanding the threats

Threat	Property	Definition	Example
Spoofing	Authentication	Impersonating something or someone else.	Pretending to be any of billg, xbox.com or a system update
Tampering	Integrity	Modifying data or code	Modifying a game config file on disk, or a packet as it traverses the network
Repudiation	Non-repudiation	Claiming to have not performed an action	“I didn’t cheat!”
Information Disclosure	Confidentiality	Exposing information to someone not authorized to see it	Reading key material from an app
Denial of Service	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
Elevation of Privilege	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)



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 ☺️

Exercise: Identifying Threats and Mitigations

Example Threats & Mitigations

Location	Threat	Mitigation	Notes
1	Unencrypted data in transit can be modified or stolen	Use HTTPS or TLS transport	EBS? Encryption to perimiter
1	Producer can spoof the ID of some other org, poisoning data for that org	Authenticate and authorize the producer against the intended tenant ID. Don't rely on ID just in the data (if it's attached to the data)	
2 & 7	Query process suffers from an injection flaw that allows querying data from an orgId other than the one authorized	Use a prepared statement when querying the database. Ensure that the custom plugin doesn't concatenate strings in an unsafe way - **might need security expert review since this would be a pretty critical flaw; find SQLi expert?.**	Also namespaces and auth connections to a namespace?

Location	Threat	Mitigation	Notes
Tenant Env	If one tenant env is able to communicate with another (i.e., poor isolation), code from one ML processor could attack another	**Partial**: process isolation **Partial**: (better) linux container isolation per tenant **Full**: (best) VM/hypervisor isolation per tenant	AWS: - EC2 is about the same as VM isolation. - Could enhance with machine roles and AMI permissions

Location	Threat	Mitigation	Notes
3	Malicious Data Scientist(DS) pushes code to an instance they're not permitted to	Authenticate and authorize that the DS has permission to write code to this this tenant	AWS: - EC2 machine role / EBS role has tenant id - Give data scientist access to that tenant ID in IAM
3	Malicious Data Scientist writes malware code, runs lateral movement attacks	? Scan uploaded code? Build pipeline security? Is this shell commands to upload this or some sort of code promotion? Is logging a sufficient defense?	**This might be pretty dangerous. Need ideas. Sandboxing? Outbound proxy to control access**

Location	Threat	Mitigation	Notes
ML Processor	Malicious Data Scientist writes processing code to exfiltrate raw data or model to some external website	Disallow all outbound web connections from ML processor (firewall -- partial: host, full: network)	AWS: - Security groups are a good firewall-like defense, could be defined per tenant to have an allow-list via http proxy?
ML Processor	Data Scientist uses shell access (e.g., SSH) to ML Processor machine to exfiltrate data back to the Data Scientist's host	**Full**: no shell access **Partial**: log access, monitor connection for traffic anomalies,	Do they need shell access?

Location	Threat	Mitigation	Notes
4	Processor is able to subscribe to a topic that it doesn’t have permission to, stealing that tenant’s data	Name topics after tenant/model IDs, write an authorization extension for Kafka (Java) that checks the inbound connection is authorized to read.	Could also be a rogue actor on the network, so pays to have good authentication here.
5	Processor can query or modify existing model data for another tenant	Authorize that the Processor has permission to access this tenant’s data (Data Store’s responsibility!)	Could use PostgreSQL namespaces to isolate tenants? Does persistent connection pooling make this difficult?
6	Processor can query or modify existing replay data for another tenant	(same as other postgres authorization cases)	(see elsewhere)

Location	Threat	Mitigation	Notes
8 & Authn/ Authz datastore	Class of attacks: central point of authority	Re-use best practices here. Isn't there a federated auth solution that our peers built that we could re-use here? JWTs? SAML? Kerberos?	
9	Data Scientists causes data from one tenant to be replayed into another tenant	Ensure that the source replay database and Kafka topic being written to belong to the same tenant **Better**: the Input Gateway can symmetrically encrypt the data upon receipt with a tenant specific key? This would systematically prevent access to another tenants data, solve data-at-rest issues, etc. Processor and Replay processes can request decryption key for that tenant	Shared responsibility of Kafka (auth the connecting Replay process) and Replay Database (auth the replay process) **For the encryption mitigation**: Is this a MadDog/K4A/etc. Use case? If in AWS, can we leverage machine roles and KMS (i.e., only permit an EC2 instance access to Keys of the same tenancy)

Location	Threat	Mitigation	Notes
10	Data Scientist causes Replay process to obtain data from another tenant’s replay datastore	(see above for #9 - Pg needs to auth access on a per tenant basis, ideally)	
Production Env	Network attacker could sniff traffic and steal customer data	Machine to machine connection encryption? Mesh with authorization somehow?	*Probably need an exception for this unless the infra team has encrypted SDN (software defined networking) or something?*

Strategizing Build & Verifying

Before you proceed...

CAPTURE EVERYTHING THAT’S ON A WHITEBOARD

Build and Verify

Simple!

1. Build (code, network, etc.)
2. Verify that it works correctly according to the threat model

On Correctness

- Functionality is verified to be correct
- Security should be verified the same sort of way!
- Use this as an opportunity to use Static Code Analysis
- Re-use secure “substrate” & security control libraries
- Follow your coding standards!

So, to finish up we need to

1. Identify things to watch out for during building
2. Identify any “secure substrate” that we can re-use to give us better velocity
3. Start a test plan that includes techniques that hackers themselves would use against these defenses

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

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

Exercise: Build and Verification Strategy

Example Build Strategies

Considerations before building:

- Tenancy authentication and authorization is pretty important, but amounts to a lot of work. What Library/framework/protocol can we re-use for this?
- Does database connection pooling work with authenticating/authorizing tenancy at the database layer?

The security of the processing subsystem hinges on the Data Scientist

- Contrast: how much of the overall security depends on sysops not abusing shells/packages?
- Do we have to trust them, but implement some partial mitigations to keep them honest?
- Might want to look at non-Repudiation threats in that case.
- Might be able to make some trade-offs, or lower residual risk, if data scientists can use hardened workstations of some sort?

Example Test Strategies

Things to attack:

- Kafka authentication / spoofing, tampering, etc.
  - Kafka’s relatively immature... wire protocol fuzzing? abuse TLS implementation?
- Data format flaws - converting between input and output formats
  - e.g., try JSON injection with superfluous quote characters
- If using JWTs, cover the JWS “none” algorithm and the “Switch RS256 signatures to HS256” protocol flaws
- Backdoors: database backups/replicas, open shell sessions from data scientists
- Build pipeline for data scientists - see how hard it is to abuse this from certain “insider threat” postures
- Pretend to be one tenant and try to attack another tenant

Remember...

1. What are you building? -> Data flow diagram

1. What can go wrong? -> Identify threats

1. What are you going to do about it? - Identify mitigations

1. Check your work on 1-3

Thanks