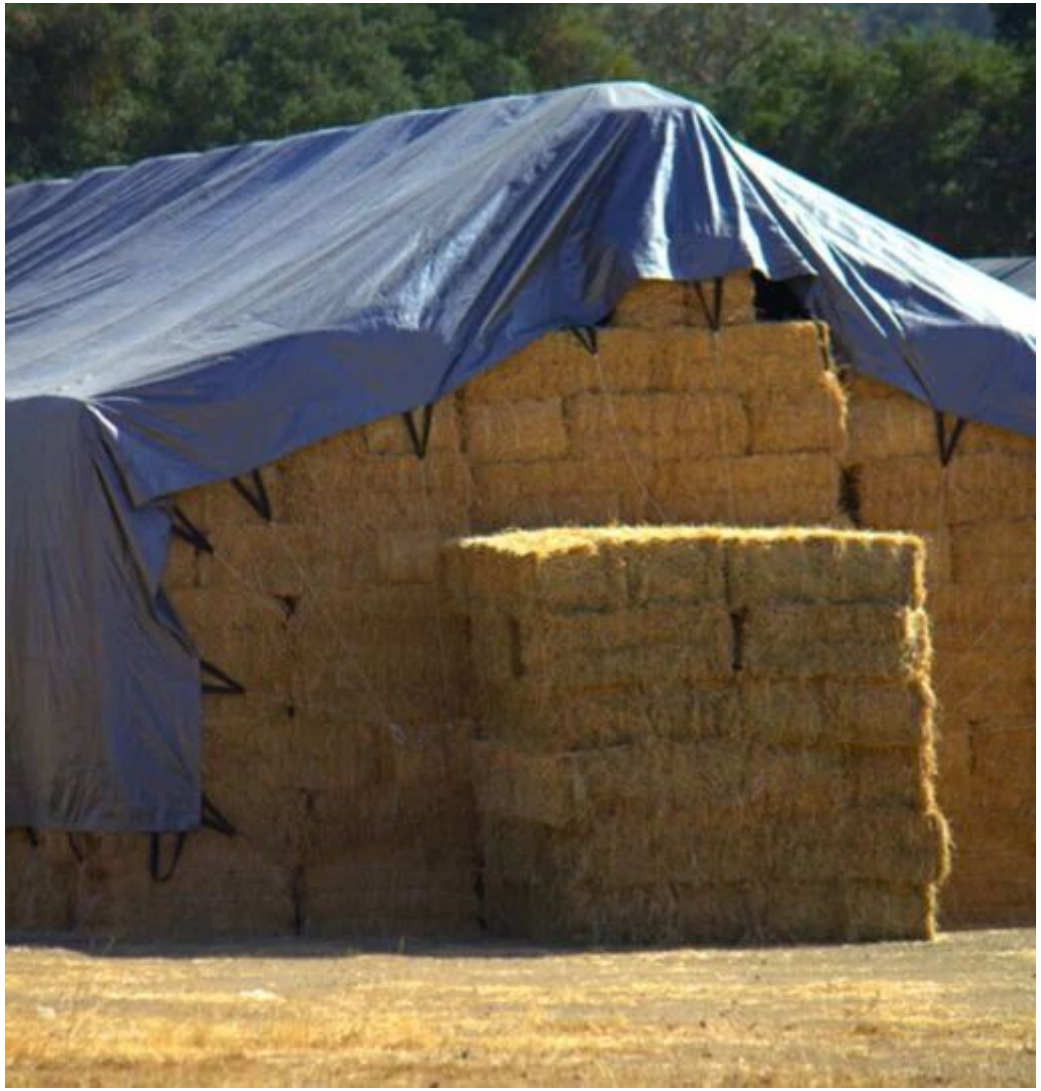
# Securing a Haystack

## How to approach Threat Modeling on the Design Level

Joern Freydank

https://www.linkedin.com/in/joernfre/



# Intro/Bio

**Bio**

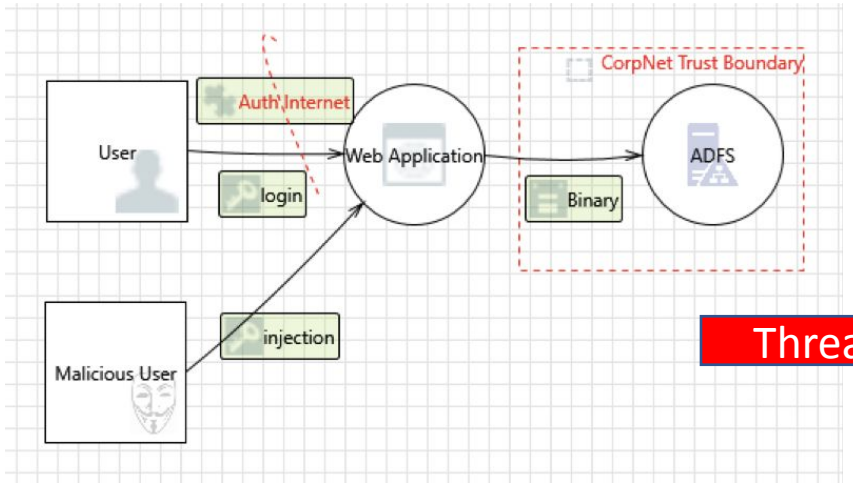
- TU-Berlin, Germany (MSc. Computer Engineering)
- Started in security by dongle cracking
- Software Developer embedded and Enterprise Systems
- 20+ years Experience, CISSP
- Bank/ATM/Payment Security
- Splunk (CISCO) Principal Product Security Engineer

Needs to secure stack of 800 bales of hay (20 tons) for 4 horses.

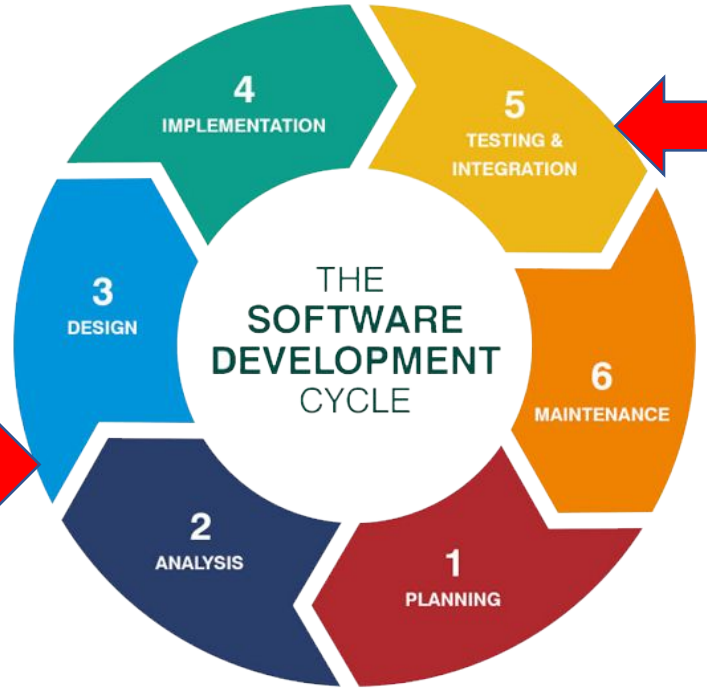
Email: joernfre@yahoo.com
LinkedIn: https://www.linkedin.com/in/joernfre/

# Design Threat Model in the SDLC

- Identified Threats to Custom Systems
- Created early during Software Design Phase
- Identifies (missing) controls
- Provides Security Requirements




Threat Model




Pen-testing

- Consumed by Pen-testers
- Provides Reference

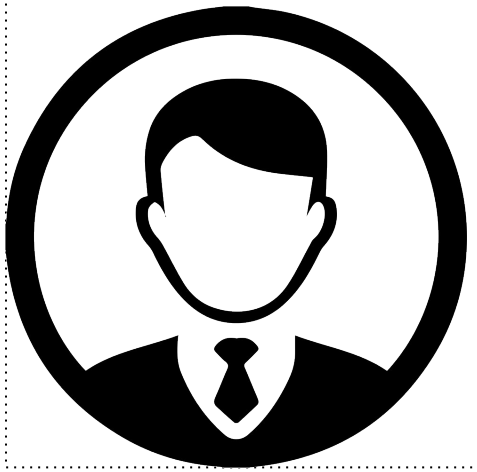
# Design Analysis – Behaviour Driven

- Follow the business behavior via the use cases (data routes)
- Systems execute actions on behalf of a business user

## Decompose: Who does what to Whom (and how)?

**Who**

- Users



Does What

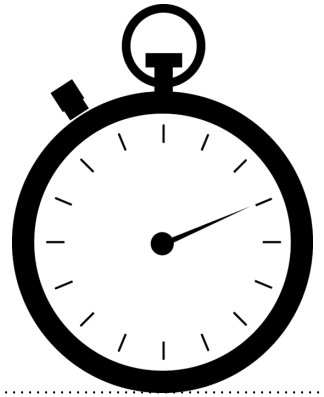
- Actions (Processes)

To Whom

- Assets

How much/when?

- Quantifiers



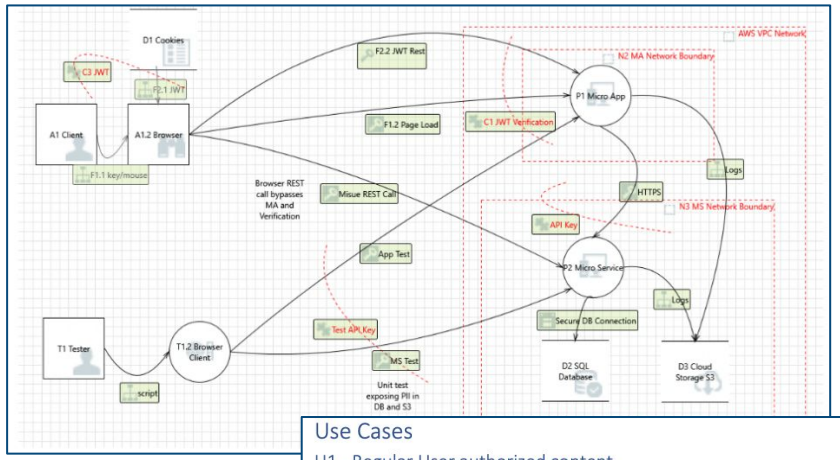
# Keep it Simple

+ High Level starting with Access
+ Categorical Level
  - ✔ web-app
  - ✔ API-service
+ Only behavioral use cases
  - ✔ authenticated flow
  - ✔ unauthenticated flow


**Goal: Identify what is NOT there but needs to be there!**

# WHAT CONSTITUTES A THREAT MODEL ?

## 1. Dataflow Diagram




## 2. Inventory: Users (Actors) , Assets, Boundaries

### Actors

#### Regular Actors

A1 NM User
User is registered with NM, has NM data record and went through stepped up MFA process.

A2 Yodlee User
Yodlee User who needs to be fully setup in NM system and additionally in Yodlee.

A1 FR User
User Financial representative who is registered with NM and represents multiple NM User clients. An FR user who can partially assume the role of a client.

A4 Okta
External Identity provider, providing Registration, MFA and user management services

A5 Yodlee
External Yodlee or TX system that is used to provide bank account accumulation by providing access to a set of client's bank accounts records.

### Security Assets

D1 Browser Cookie Storage
Browser Cookie Storage, persistent storage of the web browser's Cookies which is controlled (disclosure/read) by the browser's cookie policy and the physical machine's authentication mechanism.

D2 SQL Database
Cloud based SQL Database
D3 Cloud Storage
Cloud based File storage

P1 Website (Micro Application)
Micro Application, Website hosted in the cloud.

P2 Micro Service
Micro Service serving up data and business logic processes.

## 3. Use and Misuse Cases

### Use Cases

U1 - Regular User authorized content
A regular user uses his web browser to access the authorized content of a W

Since the request contains his JWT token obtained from a previous login process he is allowed to access the site. The cookie is stored in the browser's cookie storage.

| U1 - Regular User Cx-login | |
|---|---|
| Flows | F2.1, F1.2 |
| Assets/Actors: | T1, T1.2 |
| Processes | P1 |
| Boundaries | AWS VPC Network, N2 |
| Controls | J1 |
| Outcome | Control operates as intended |

U2 – Misuse: Test user accesses MicoService directly and retrieves bulk PII

| U2 – Test User accesses Micro Service directly | |
|---|---|
| Flows | F4, F9, F9.2 |
| Assets/Actors: | A1, A1.2 |
| Processes | P2 |
| Boundaries | N1, N3, N3, N4 |
| Controls | Test API Key, Missing quantity and timing control |
| Outcome | Intentional misuse case |

Documentation

## 4. Threats + Controls

### Threats

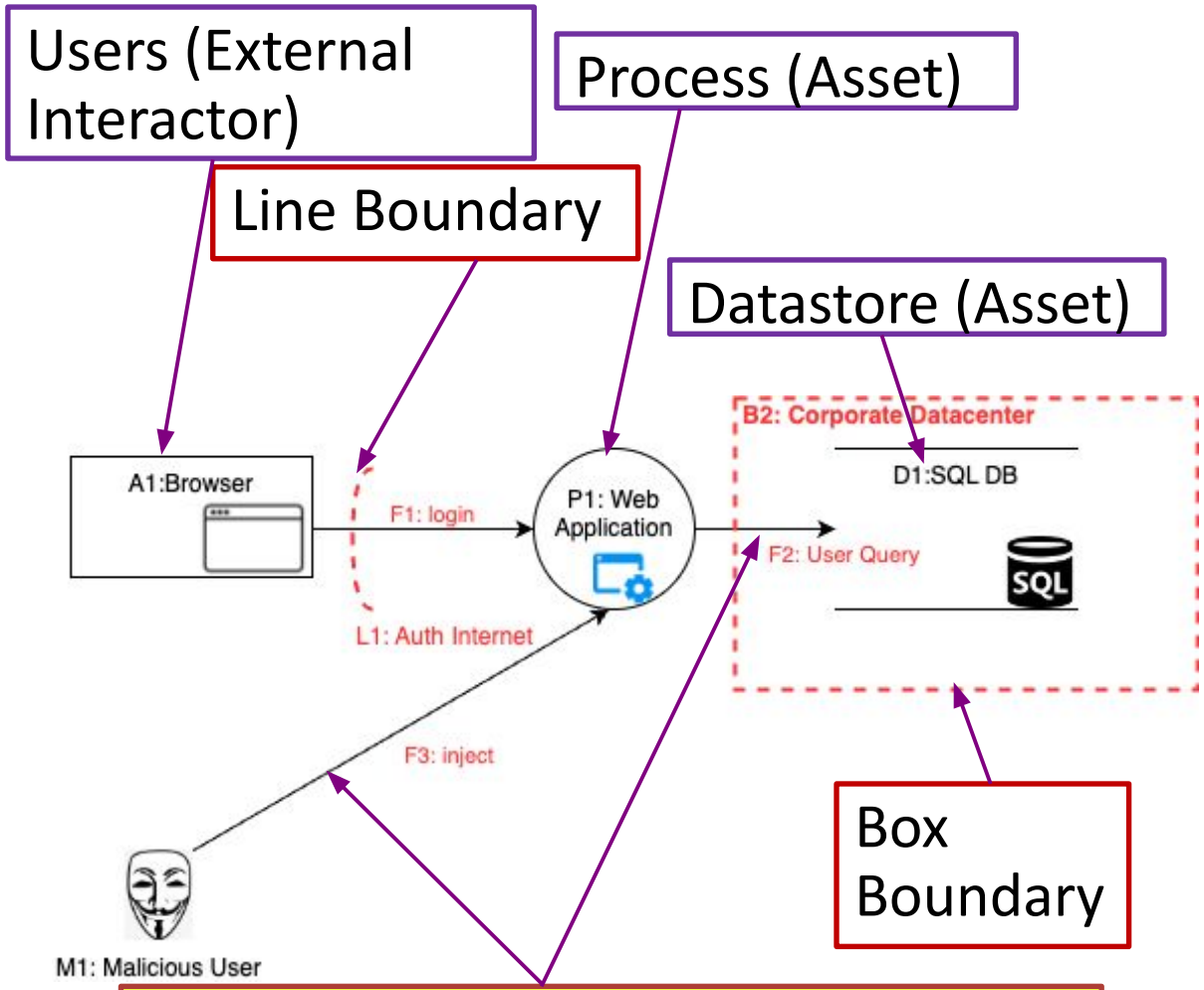
T1 - Datadump file access
Any system admin of the BIIPODS system – which might be different from the data admin role will still be able to access or recover the datadump (created during batch load U2, interaction F10) after the batch import completes on the file system.
Countermeasures (controls) include file level encryption of the datadump file and secure deletion.

T2 -Secrets stored in AWS Systems Manager
API and data encryption secrets stored in AWS Systems Manager could be accessed by rogue lambda or be disclosed in lambda logs. (TBD Verify STND-00577)

T3 – Tagging querying bypass (Future Considerations)
Tagging controls could be bypassed by modifying lambda functions directly or indirectly by only modifying the parameters, especially the lambda functions dealing with metadata designed to convert data across planes.

T4 – Reverse De-identification Measures (Future Considerations)
Zone specific De-identification controls could be bypassed (reverted) by accessing and querying the index data or cache data directly from metadata database which combines data across zones.

# Example: Elements – Flows, Threats and Controls




**Identified Threats (Web Application)**

(S) Spoofing of Identity of User

✔ Login Control

(I) Information Disclosure of PII

✔ TLS 1.3 (Control)

(E) Elevation of Privilege of Admin role

✔ Authorization Token (Control)

**Identified Threats (DB)**

(S) Spoofing of Identity of Web App

✔ Client Certificate (Control)

(I) Information Disclosure of PII

✔ VPN Protocol

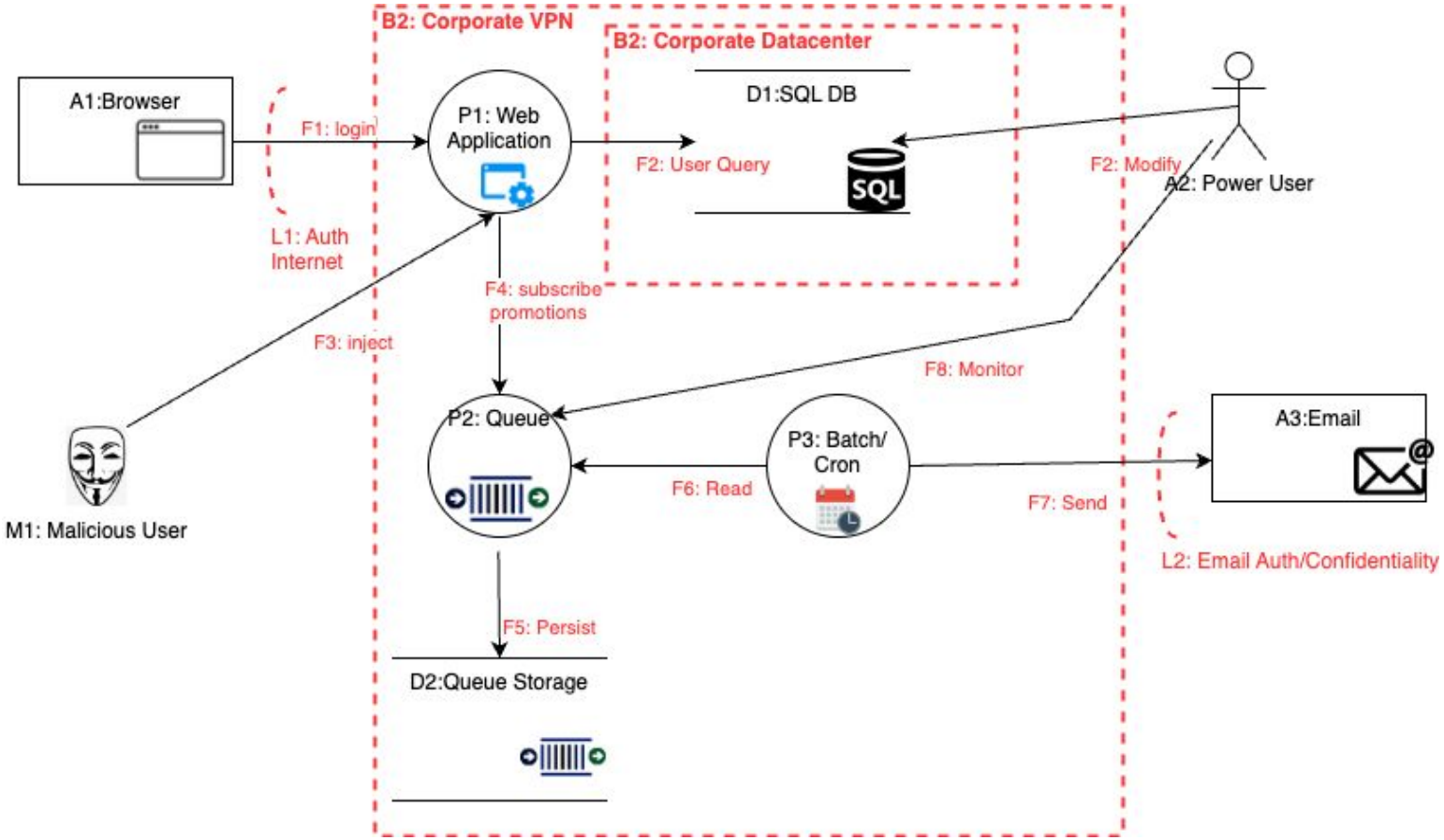
(T) Tampering SQL Injection

✗ No Sanitization Control

# Complex Use Case : Coupon Subscription



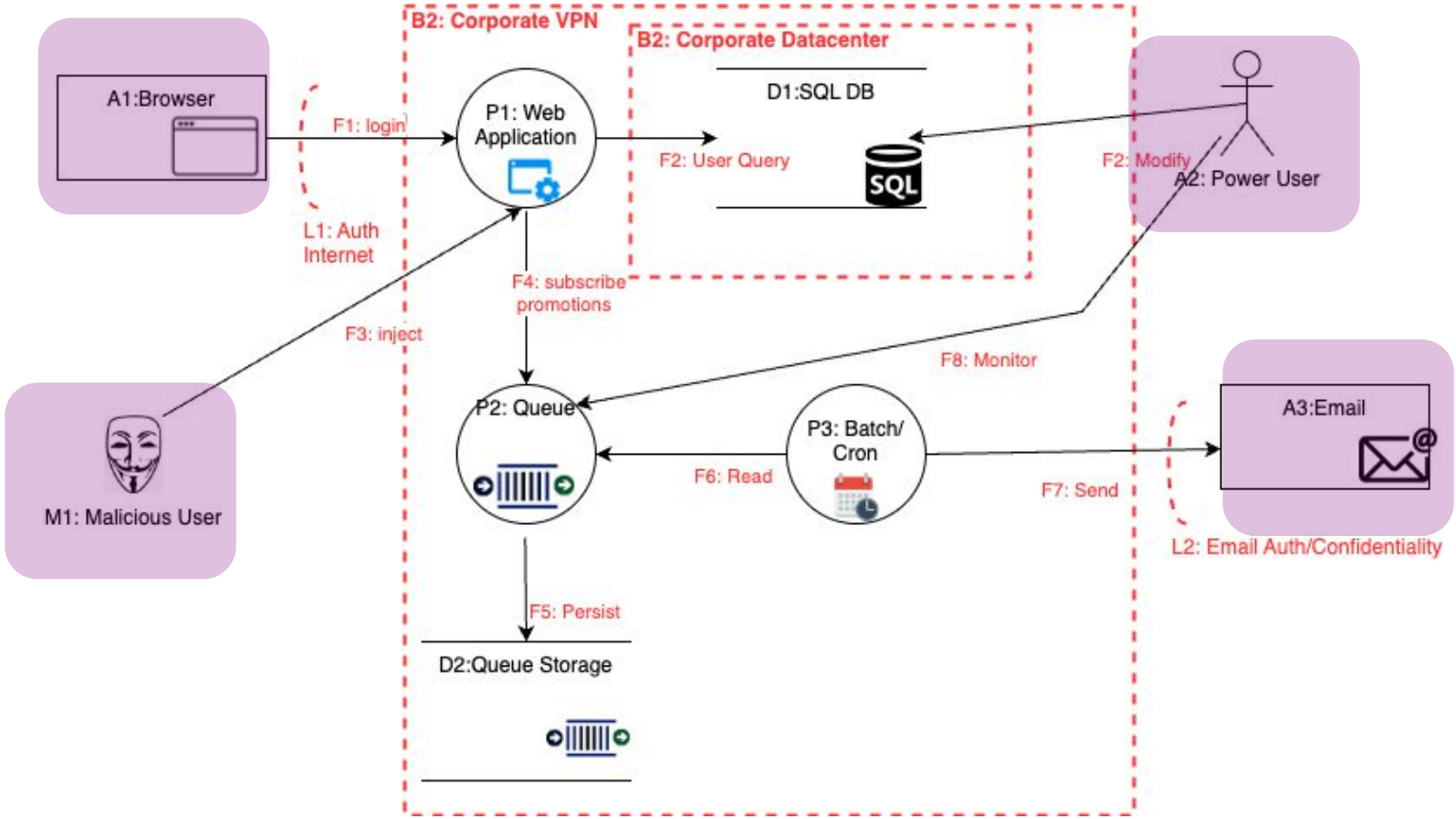
**Customers enter their email addresses into webpage and system sends out emails with coupons.**



# Complex Use Case: Users



**"Who does what to whom"**

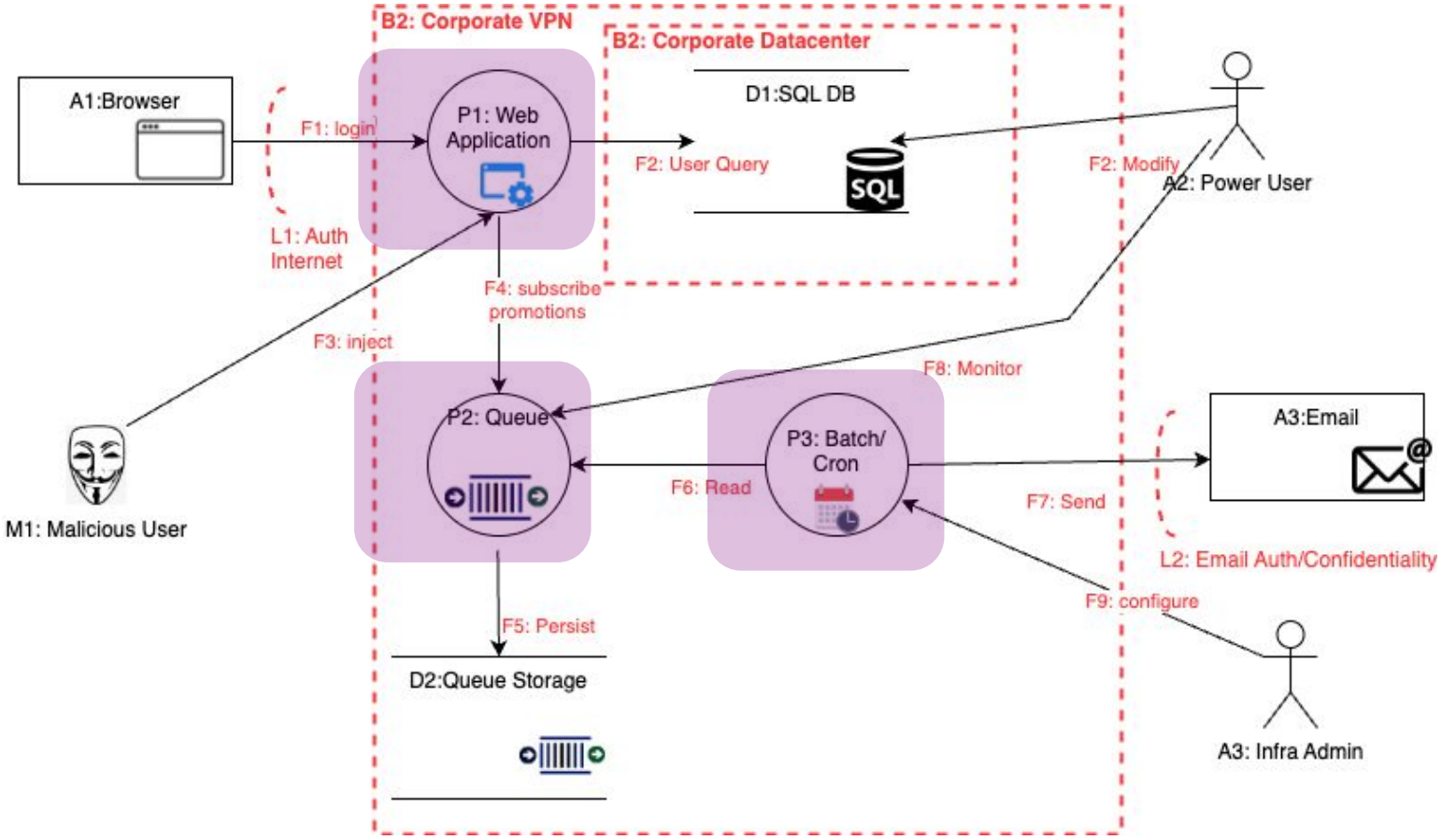



- Human Users or System Users initiate (start) use cases
- They provide data or input to the system
- External end-points (receive data)
- Malicious Users
- **Modeled system does not trust anything they do**

# Complex Use Case: Processes (Computational Assets)



**“Who does what to whom”**

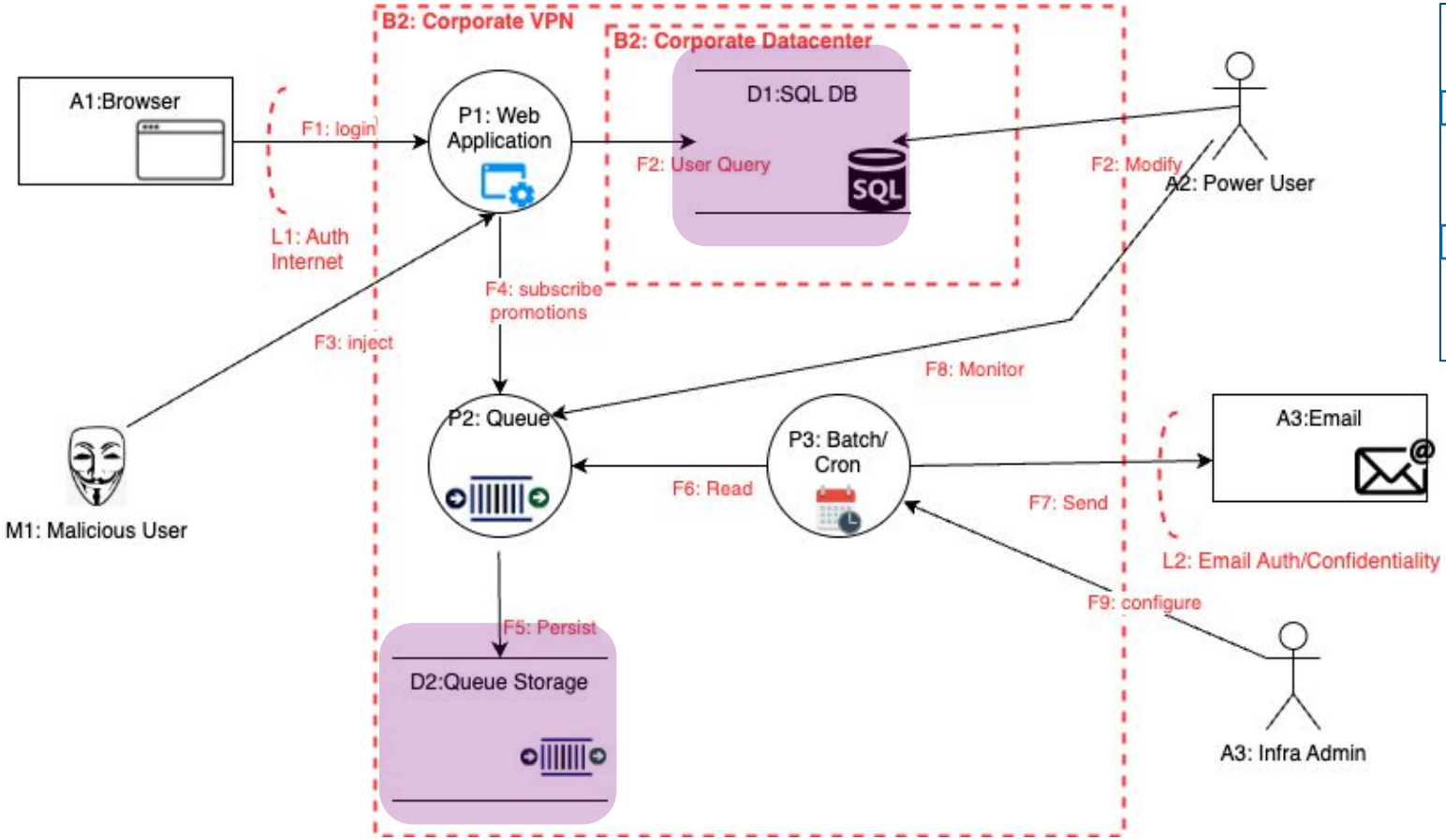



- **Modeled system trusts processes**
- Processes react to (not start) interactions from Users
- Controlled by dataflow and configuration
- Processes need to be protected

# Complex Use Case: Datastores ( Assets)



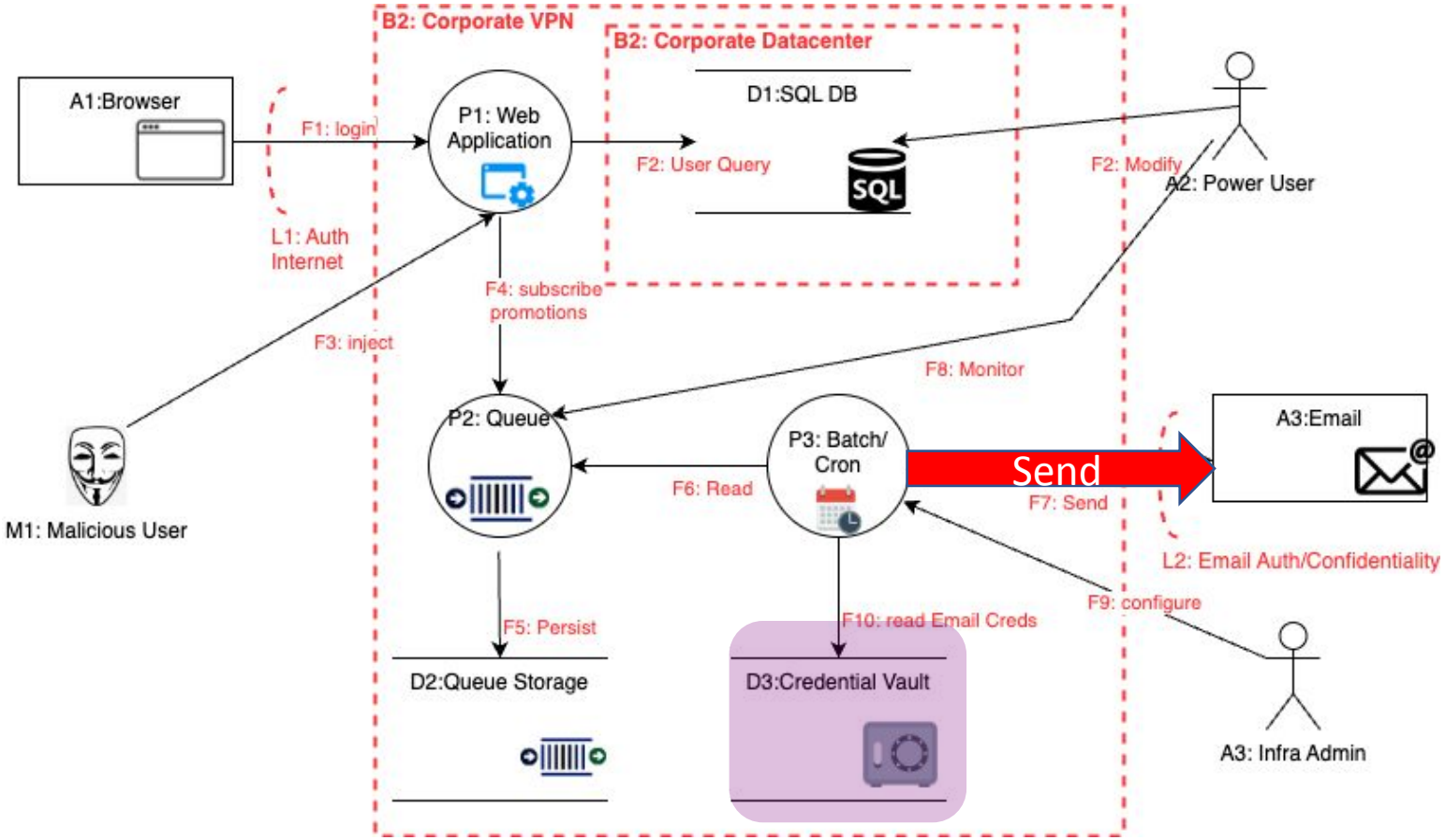
**“Who does what to whom”**




- Datastores are security assets that represent data
- Datastores need to be protected

# Complex Use Case: Boundaries, Flows and Threats






**Threats for Send (L2):**

- Information Disclosure
  - ✔ Control: SMTP TLS
- Spoofing of Identity
  - ✔ Control: Email Credentials

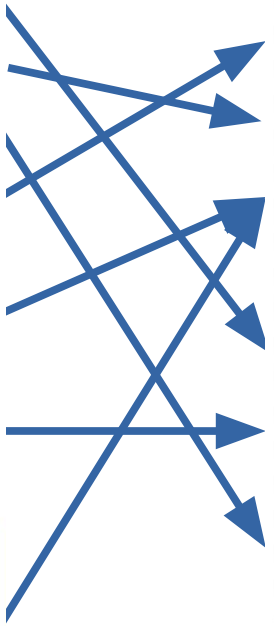
What is missing?

- Lookup Threats in Threat Library by Elements
- Use Tools: Microsoft Threat Modeler / Irius Risk
- Discuss What-If scenarios by adding Malicious Users and Use Cases
- Zooming in (more decomposition) on threats from lower-level components
- **Use categories of threats (methodologies)**

# Kicking Tires: Categories - STRIDE(A) or CIA/AAA Triade

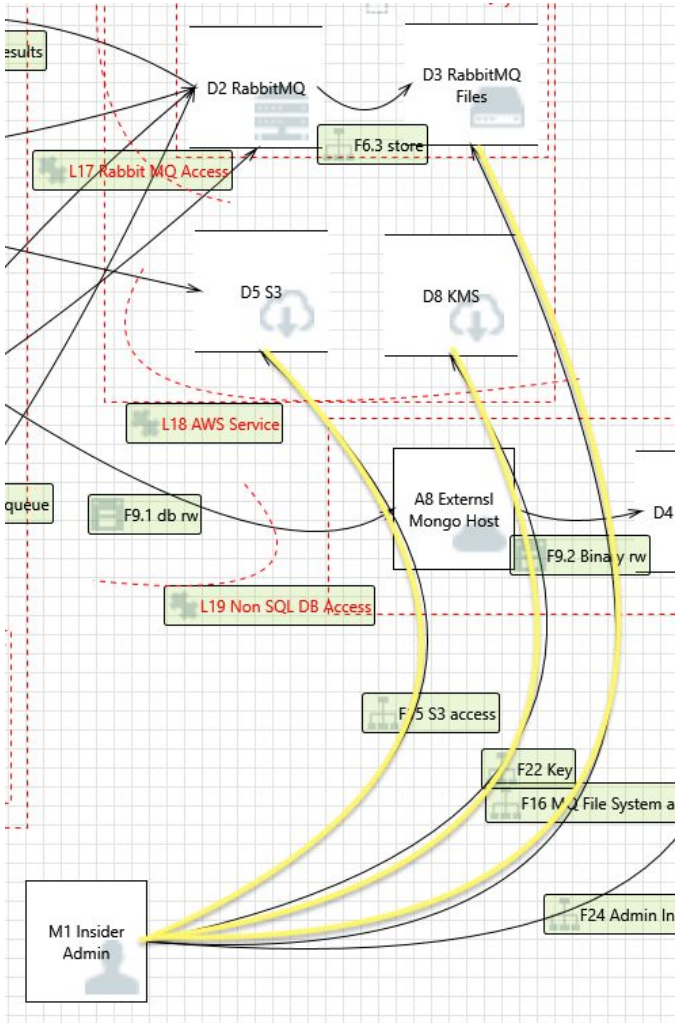


| Threat | Description |
|---|---|
| **S**poofing | Impersonating something or someone else |
| **T**ampering | Modifying data |
| **R**epudiation | Claiming to have not performed an action. |
| **I**nformation Disclosure | Exposing information to someone no authorized to see. |
| **D**enial of Service | Deny or degrade a service to users. |
| **E**levation of Privilege | Gain capabilities without proper authorization |
| **A**buse | Abuse a regular system by over extending its capabilities. |

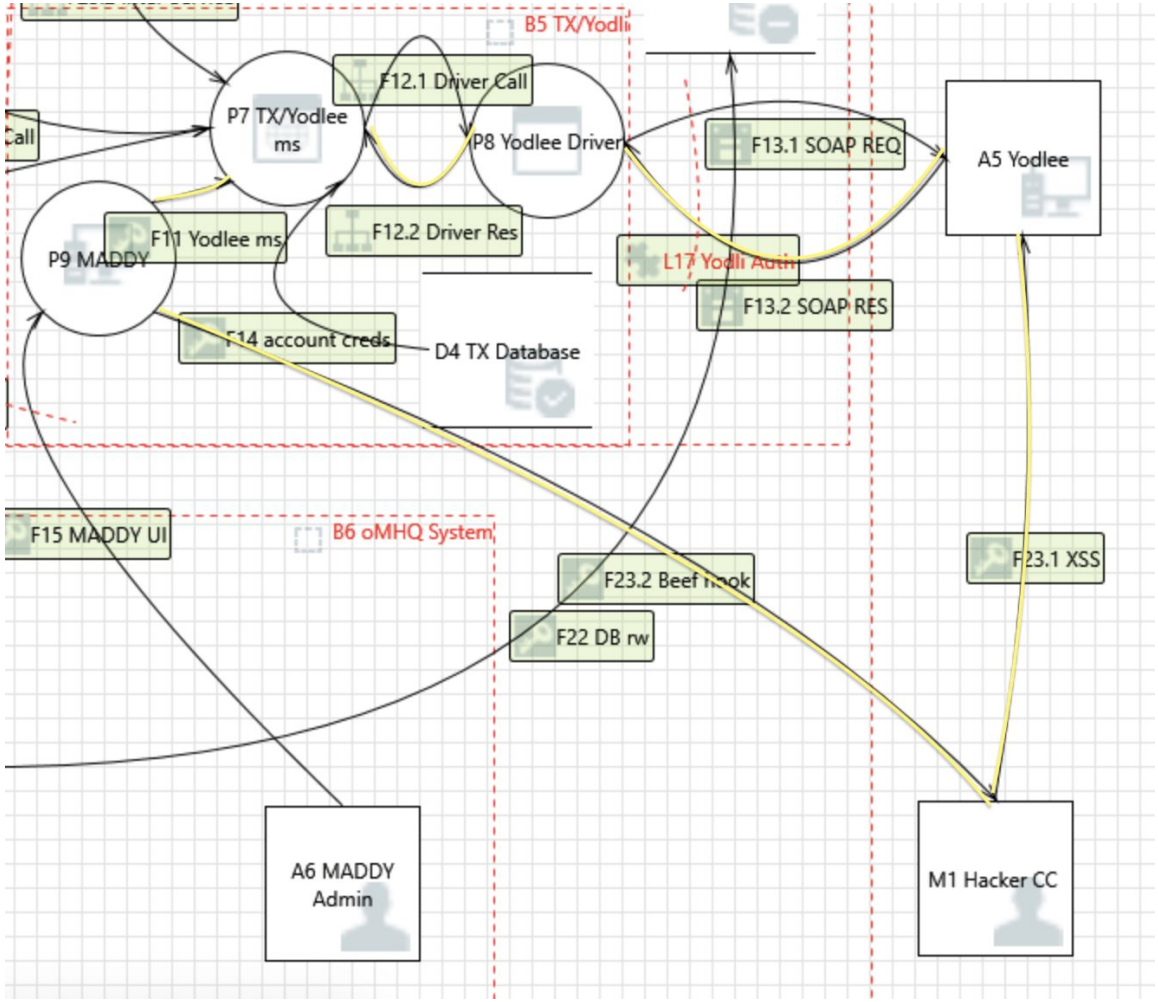


| Security Property |
|---|
| **C**onfidentiality |
| **I**ntegrity |
| **A**vailability |
| |
| **A**uthentication |
| **A**uthorization |
| **A**ccounting |

# SIMULTANEOUS ACCESS




# PIVOTING

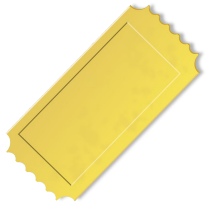


# We found Threats, now what?

**Unmitigated Threats:**

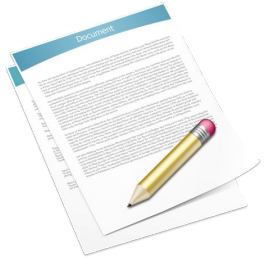
- Missing Controls –> document requirement or create ticket


- Control Bypass -> Create vulnerability (ticket) and track

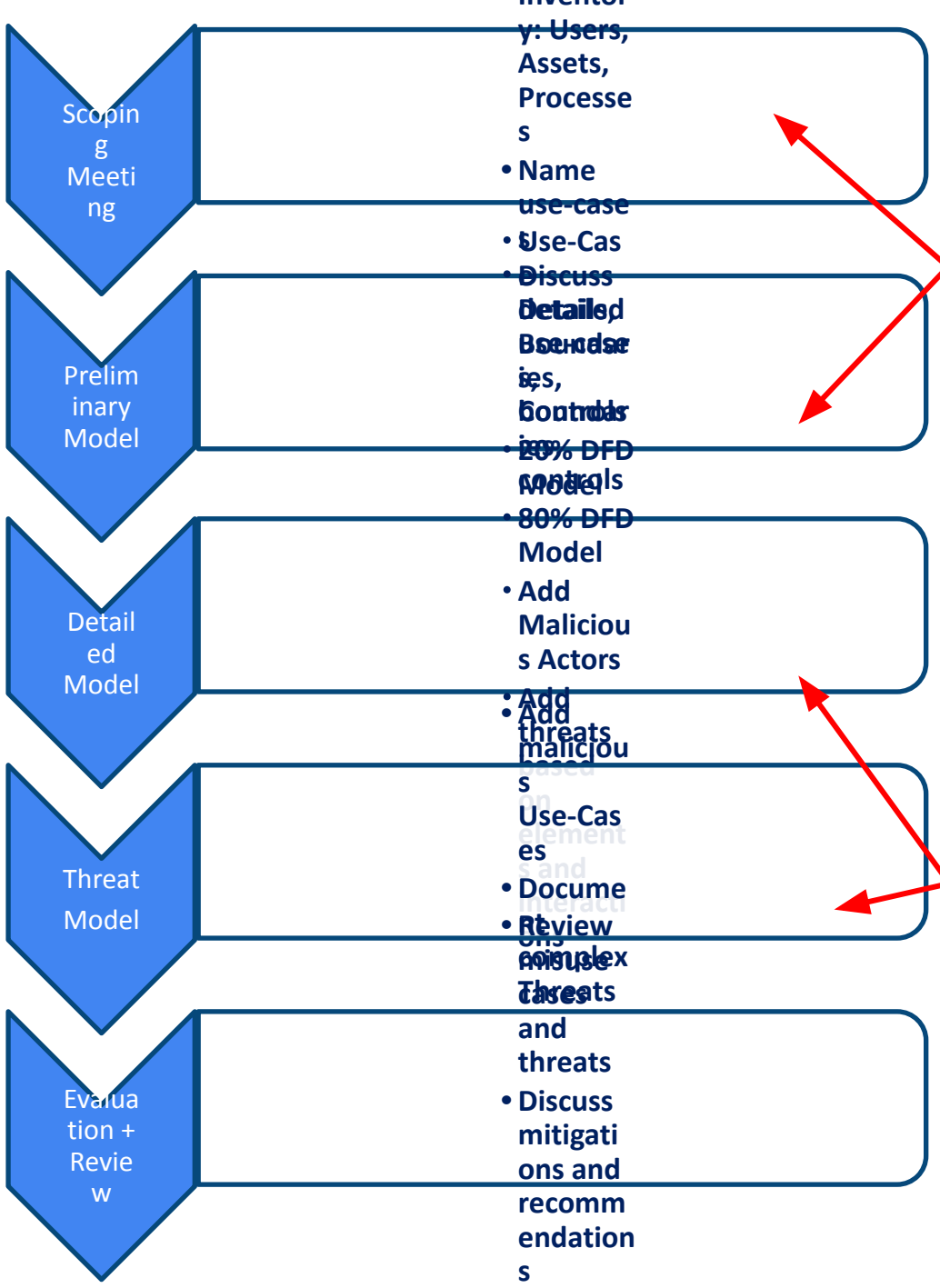


**Mitigated Threats:**

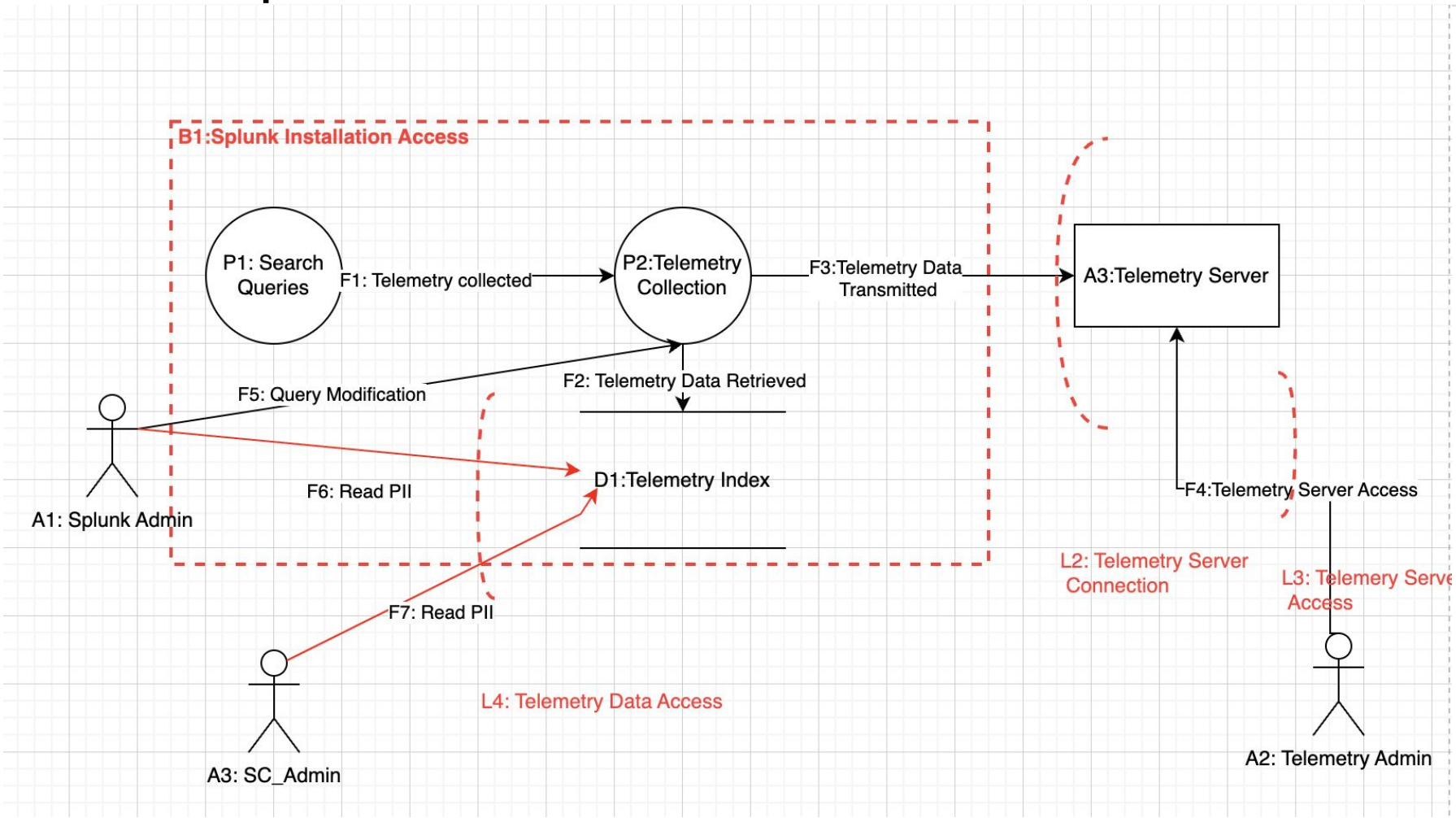
- Document together with Controls



- Schedule penetration test -> Verify Control implementation


Scoping Meeting
Preliminary Model
Detailed Model
Threat Model
Evaluation + Review
y: Users, Assets, Processes
• Name use-case
• Use-Cas
• Discuss
Detailed
Use-case
es,
Controls
20% DFD
Model
80% DFD
Model
• Add Malicious Actors
• Add
threats
malicious
Use-Cases
• Docume
• Review
misuse
cases
and
threats
• Discuss mitigations and recommendations
Insert Security Knowledge
Discover hidden elements:
Unassigned actors (name dropping), e.g. hidden roles, assets, Processes.
Discover behavioral threats:
Boundary bypasses, pivoting flows, missing controls

# Best practices: Point, define and label elements




**Naming and Numbering**

User (Actor) **A<no>**

Malicious User **M<no>**

Process **P<no>**

Datastore **D<no>**

Line Boundary **L<no>**

Box Boundary **B<no>**

# Pitfalls - Inversion of Power: Business User vs Admin



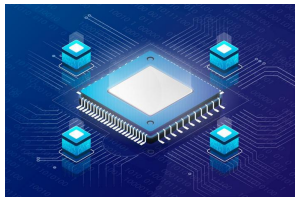
**Which role is the most valuable from an attacker point of view?**

**Business Fairy**

**Sysadmin (root)**

**User**

**Money**



**Compute**


**Data**

VALUE

Business User has higher value than Sysadmin!

# Pitfalls - Insufficient Specifications

Developer: We have no business user !

The code is checked-in into CI/CD (Gitlab), then the pipeline is triggered and the update is deployed ~~automagically~~ automatically to production.

**Users**

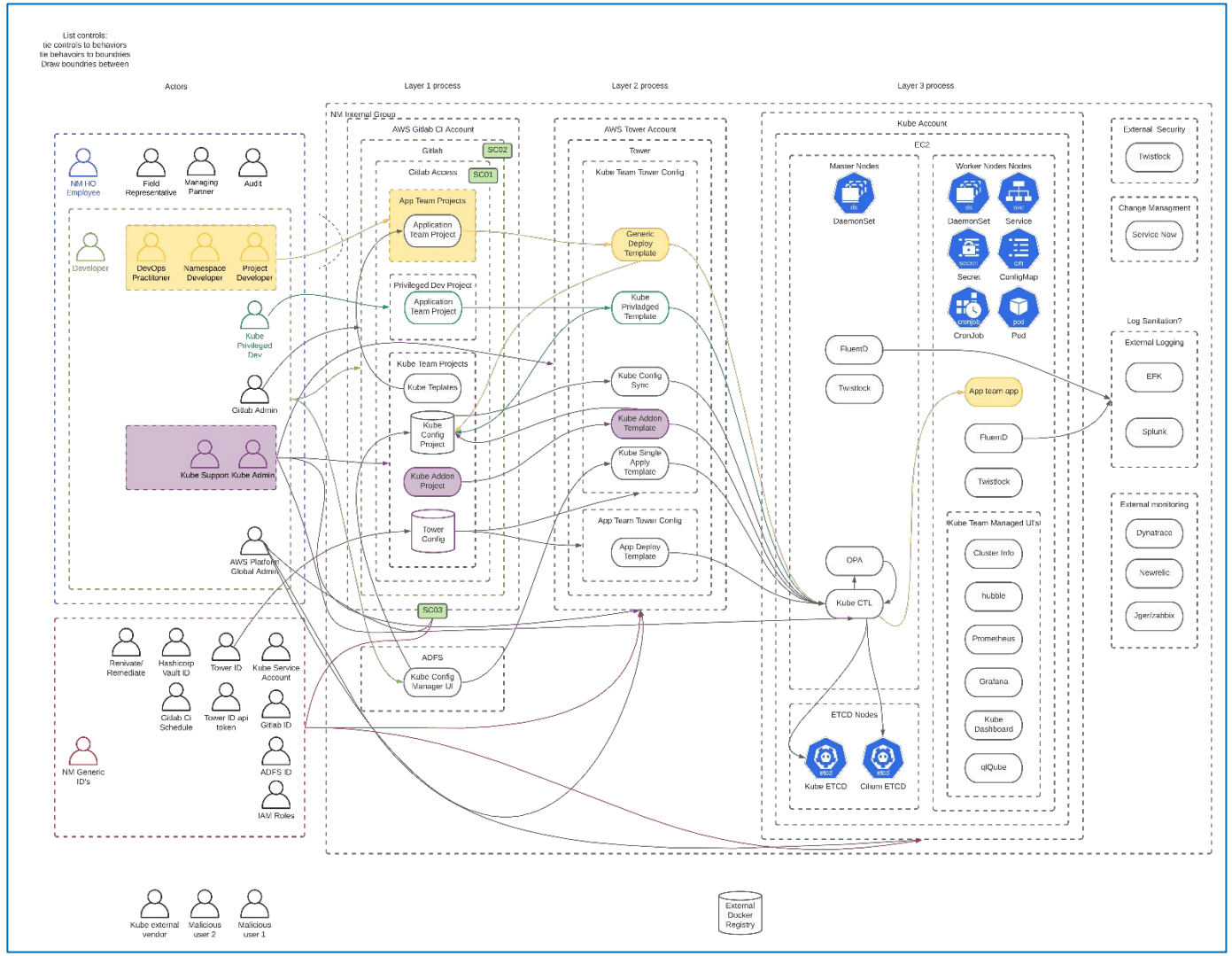
- Developers
- Deployer (Admins)
- CI/CD Admins

**Processes**

- Gitlab/CI/CD
- Pipeline

**Assets**

- Code
- Production system




# Diagram needs to fit on one page (Readable)!

# Pitfalls - Audit vs Threat Model

## Audit:

- Checkbox (Yes/No) Questions
- Single word answers
- Lots of “do you have” questions
- Most common controls

Do you have a needle in the haystack?

vs

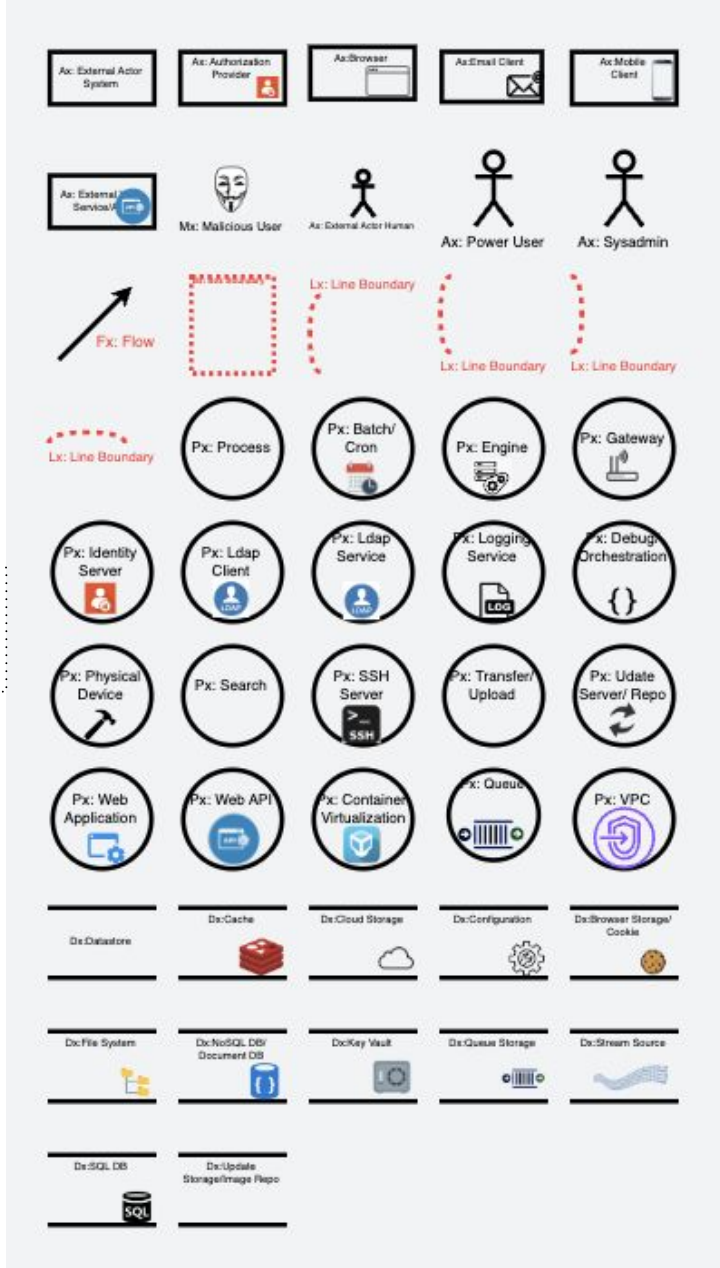
## Threat Model:

- Open Ended Questions
- Who initiated behavior?
- “Name dropping”
- Lots of “innocent questions” - “what happens if”’
- Looking for missing pieces

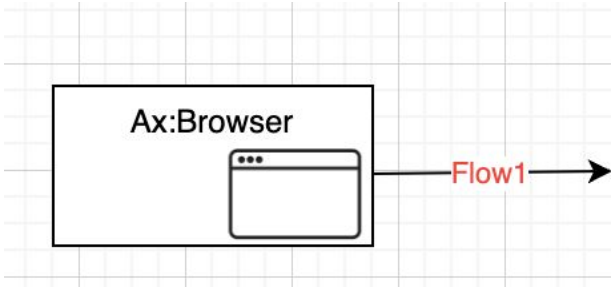
What happens if there are foreign objects in the haystack?

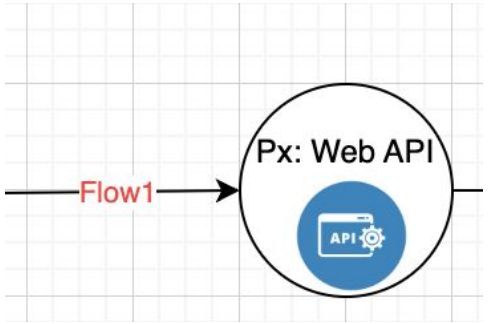
# NEXT STEPS – TEMPLATE

Example: draw.io stencil



# NEXT STEPS - THREAT LIBRARY




Browser (EI.Common.Browser)

| Threat Category > Potential Threats v | Spoofing | Tampering | Repudiation | Information Disclosure | D |
|---|---|---|---|---|---|
| **Browser sends data to another (target) element** | **Threat (TH13): Weak cross domain\| High Configuration**<br>An adversary can gain unauthorized access to API end points due to weak CORS configuration<br>***Possible Mitigations:***<br>*Ensure that only trusted origins are allowed if CORS is enabled on web server.*<br>*Refer (Azure):*<br>*Mitigate against Cross-Site Request Forgery*<br>*Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET Web APIs*<br>**Threat (TH22): Improper Logout \| High**<br>An adversary can get access to a user's session due to improper logout and timeout.<br>The session cookies is the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token he would be able to access all user data and perform all actions on behalf of user.<br>***Possible Mitigations:***<br>*Set up session for inactivity lifetime.*<br>*Implement proper logout from the application.* | **Threat (TH23): Session Cookie Readout**<br>An adversary can get access to a user's session due to insecure coding practices.<br>The session cookies is the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token he would be able to access all user data and perform all actions on behalf of user.<br>***Possible Mitigations:***<br>*Enable ValidateRequest attribute on ASP.NET Pages.*<br>*Encode untrusted web output prior to rendering.*<br>*Avoid using Html.Raw in Razor views.*<br>*Sanitization should be applied on form fields that accept all characters e.g, rich text editor*<br>*Do not assign DOM elements to sinks that do not have inbuilt encoding.*<br>**Threat (TH177): Cross domain clickjacking \| High**<br>An adversary may be able to perform action(s) on behalf of another user due to lack of controls against cross domain requests<br>***Possible Mitigations:*** | **Threat( TH173): Anonymous functions, lack of auditing \| Medium**<br>An adversary can deny performing actions against **web server** due to lack of auditing and anonymous actions leading to repudiation issues.<br>***Possible Mitigations:***<br>*Implement application level auditing and logging toed to a user ID or session especially for sensitive operations, like accessing secrets from secrets storage solutions. Other examples include user management events like successful and failed user logins, password resets, password changes, account lockouts and user registrations.* | **Threat (TH78): Cloud traffic Sniffing (I) \| High**<br>An adversary may conduct man in the middle attack and downgrade TLS connection to clear text protocol, or forcing browser communication to pass through a proxy server that he controls. This may happen because the application may use mixed content or HTTP Strict Transport Security policy is not ensured.<br>***Possible Mitigations:***<br>*Configure TLS certificate for custom domain in Cloud App Service.*<br>**Threat (TH79):Fingerprint Cloud web application \| Low**<br>An adversary can fingerprint web application by leveraging server header information<br>***Possible Mitigations:***<br>*Remove standard server headers on Cloud Web Applications to avoid fingerprinting.*<br>**Threat (TH8) Insecure Cookie Attribute \| High**<br>Attackers can steal user session cookies due to insecure cookie attributes<br>The session cookies is the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token he would be able to access all user data and perform all actions on behalf of user.<br>***Possible Mitigations:***<br>*Applications available over HTTPS must use secure cookies.*<br>*All http based application should specify http only for cookie definition.* | |

**Note: Section for incoming Data (arrow pointing to element) not displayed here.**

# Next steps – Organizing Models


- Searchable Model repository/tools, e.g. confluence


- Version / Check-in of diagrams

- Associate Models with code repos, e.g. YML file pointing to model

# Q&A & Discussion

Email: joernfre@yahoo.com
LinkedIn: https://www.linkedin.com/in/joernfre/