Sébastien Salva
LIMOS – Université d'Auvergne

Projet VASOC

- Vers l'Audit de Sécurité des Objets Connectés

- UCA- LIMOS
  - ->software
- Université Saint-Etienne, Lab Hubert Curien
  - ->hardware

- Object-As-A-Service, Agaetis, Openium, Braincube, Elkya, Les Mowdoo

OdJ

- Site Web ! https://vasoc.limos.fr/
- Etude de cas Openium, état des lieux (matériel, traces ?)
- Partie Bluetooth (Monitoring, analyse de trace) (Voir présentation Elliott)
- Partie top down: analyse des besoins, présentation ADT, CVE ? CAPEC ?

Analyse des besoins

- De nombreuses façons
- Vues
  - Attaquant (possibilité, environnement), etc.
  - Organisation (structure, failles, etc.)
  - Système, software

  - Description attaque ou vulnérabilité ou weakness
  - Modèles formels, semi-formels ou pas (logiques, automates, arbres d'attaques, data flows, etc.)

Analyse des besoins

- Des méthodologies, Stride, dread, etc.

=> BUT: produire des modèles

- Stride (Microsoft)
- Catégories étudiées:
- **S**poofing of user identity
- **T**ampering
- **R**epudiation
- **I**nformation disclosure (privacy breach or data leak)
- **D**enial of service (D.o.S)
- **E**levation of privilege

Analyse des besoins

- Des methodologies, Stride, dread, etc.

- Dread (owasp)
  - **D**amage - how bad would an attack be?
  - **R**eproducibility - how easy is it to reproduce the attack?
  - **E**xploitability - how much work is it to launch the attack?
  - **A**ffected users - how many people will be impacted?
  - **D**iscoverability - how easy is it to discover the threat?

Analyse des besoins

⇒BUT: produire des modèles

Plusieurs modèles (faites par un ing. Spécialisé en sécurité)

- 1 modèle général (actions très générales), extrait de discutions
- Modèles de + en + détaillé, spécifiques
- Puis mise en place de solutions, Audit (tests)
- Utilisation de bases (CWE, CVE, CAPEC, OWASP, etc.)

Analyse des besoins

Common Weakness Enumeration (*CWE*)

https://cwe.mitre.org/

Organisation hierarchique

https://cwe.mitre.org/data/definitions/440.htmle

Analyse des besoins

Common Attack Pattern Enumeration and Classification (*CAPEC*)

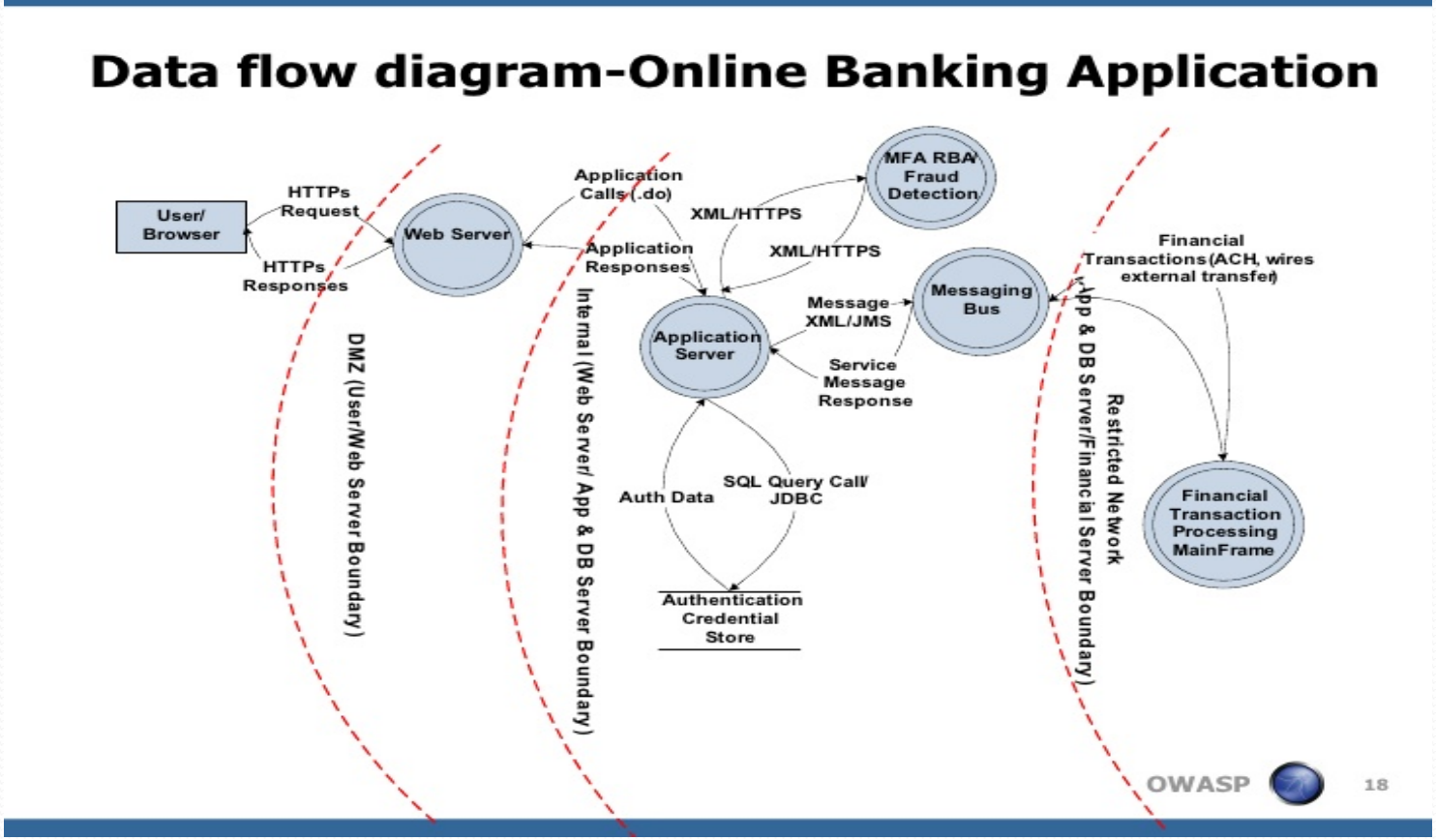
https://capec.mitre.org/

Organisation hiérarchique attaques

https://capec.mitre.org/data/definitions/66.html

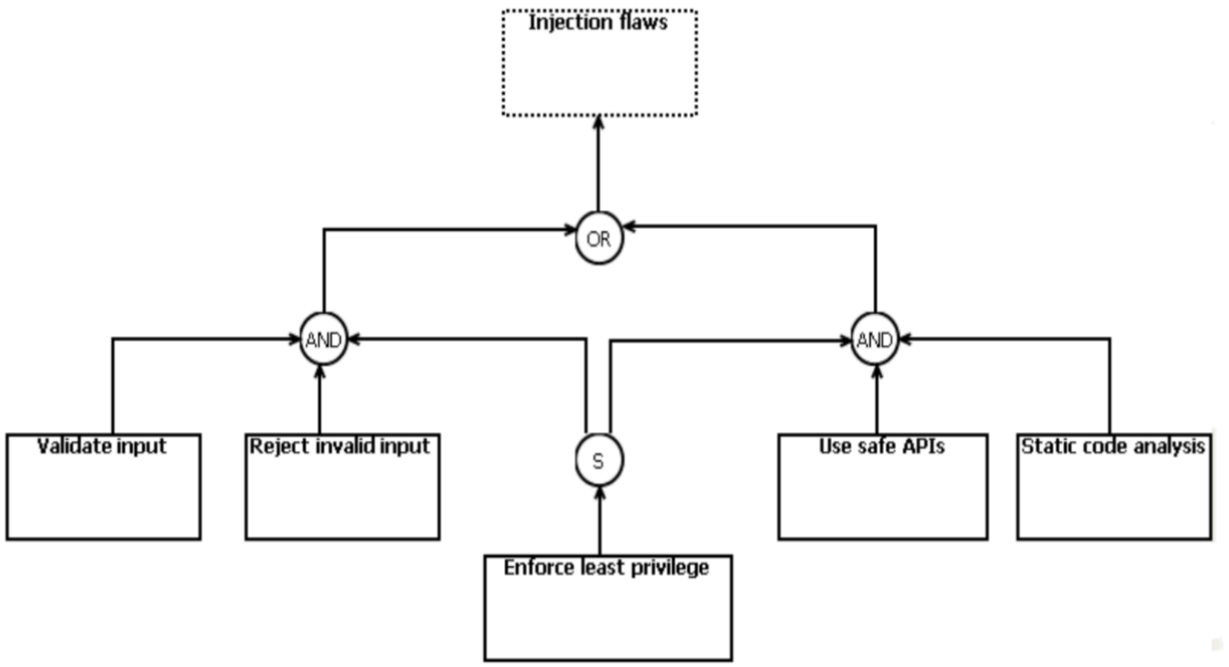
Analyse des besoins

- Exemple



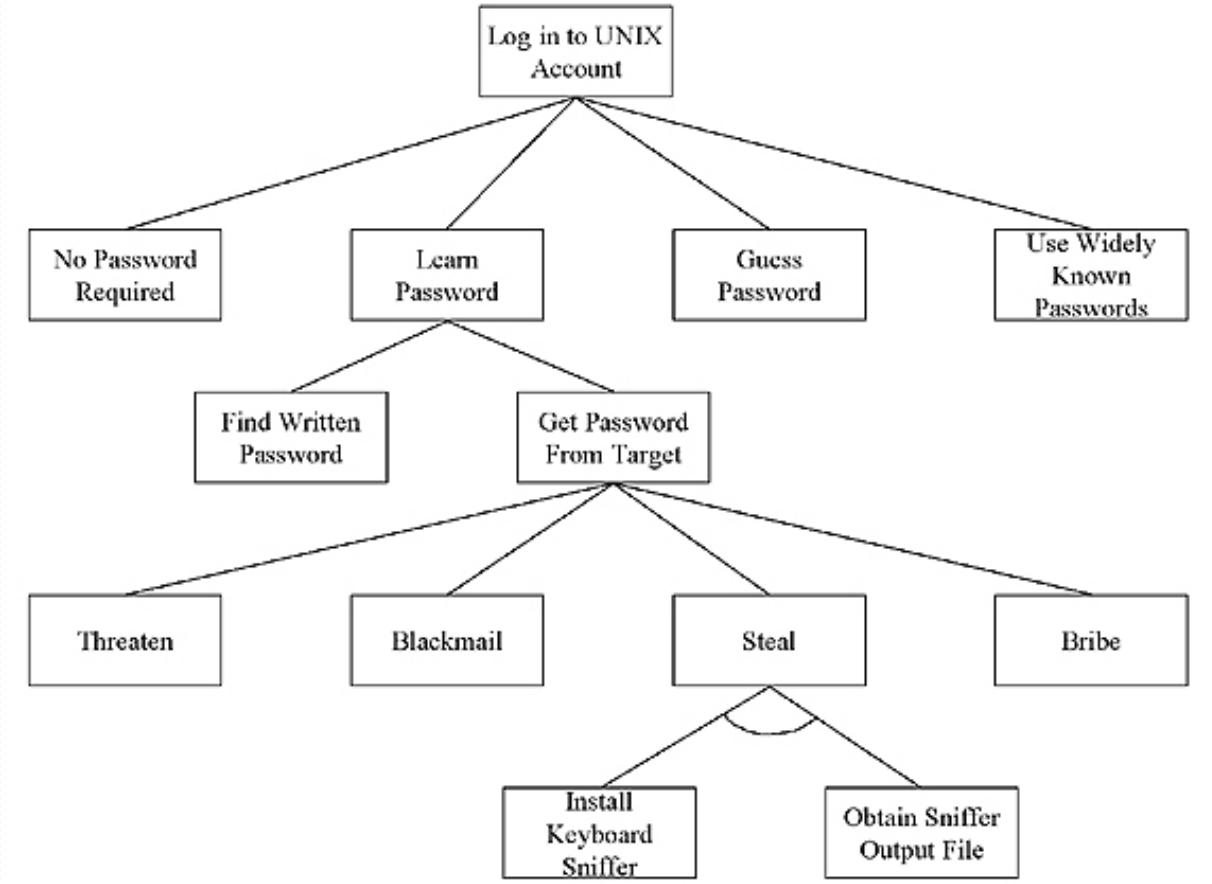
Analyse des besoins

- Exemple
- SAG
- (vulnérabilité)



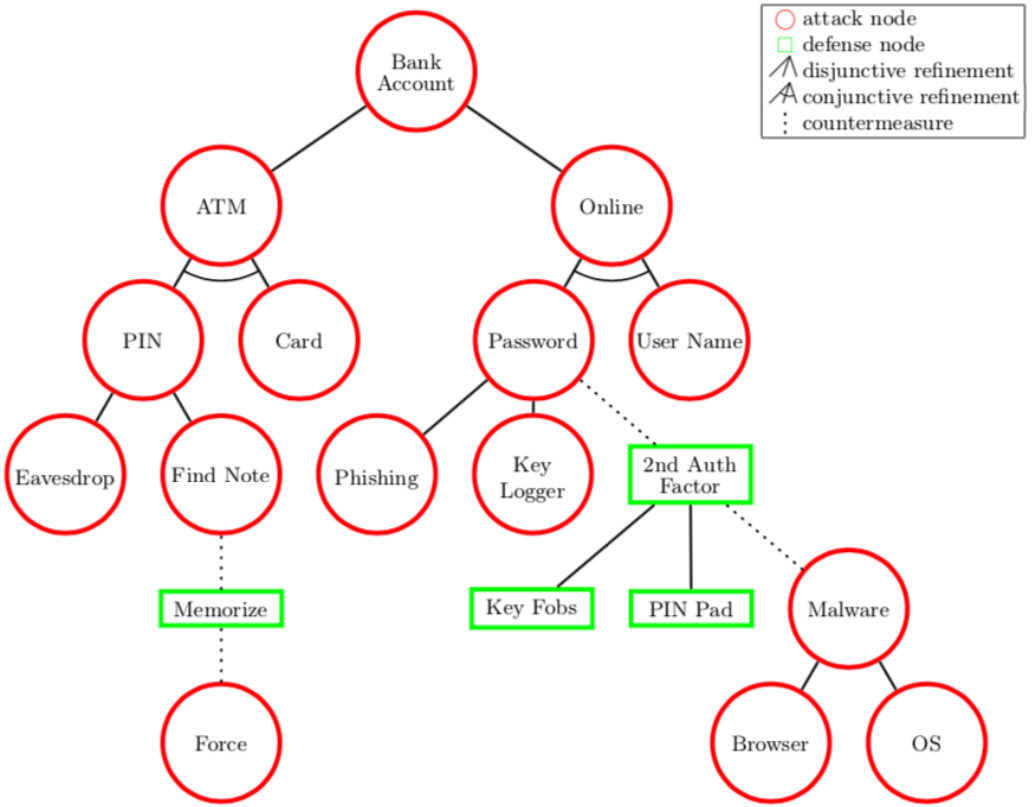
Analyse des besoins

- Exemple
- Attack Tree
- -> extraction scenarios



Analyse des besoins

- Exemple
- Attack Defense Tree
- (formel)
- -> extraction scenarios



Intégration de données pour sécuriser les applications

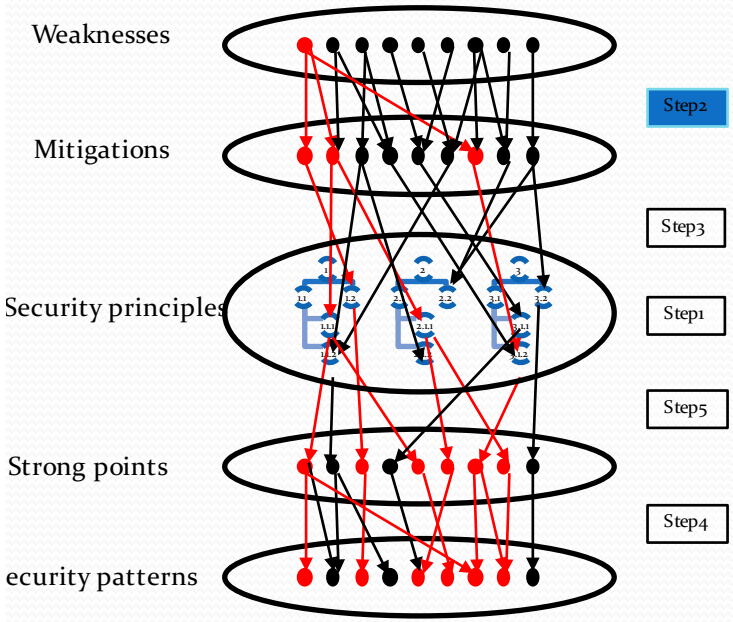
1. Exemple d'utilisation d'ADT et de bases pour générer base de connaissance + modèles + tests (voir travaux Salva-Regainia)

2. Acquisition et intégration de données de sécurité (semi-auto) -> base de connaissance contenant:
   - Attaques, contre-mesures, principes, étapes de tests (Cucumber), etc.

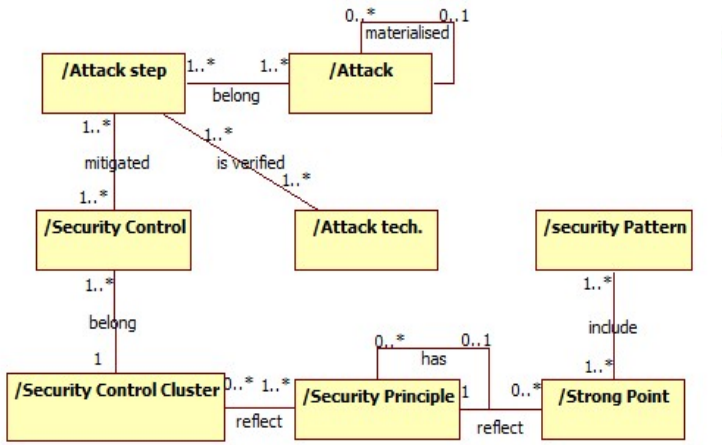
3. Extraction de SAG et d'ADT

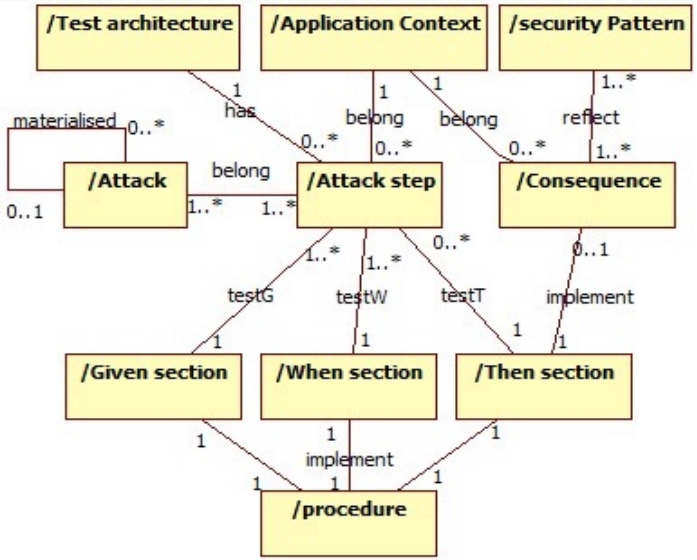
4. Gen. De squelettes de cas de test (cucumber) qui font appel à des outils de test de pénétration (Zap, sqlmap, etc.)

Intégration de données pour sécuriser les applications



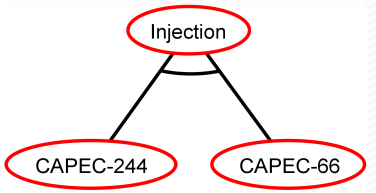
Entités et quelques relations

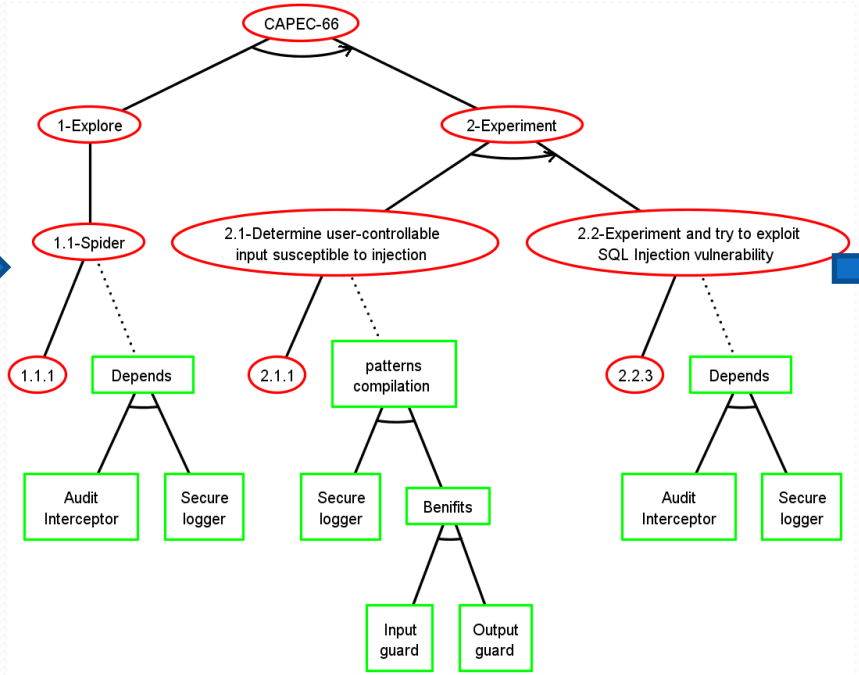




Meta-modèle de la base de connaissance

Intégration de données pour sécuriser les applications





ADT initial

Gen. ADT détaillé

```
Feature: CAPEC-66: SQL Injection

#1. Explore
Scenario: Step1.1 Survey application
#The attacker first takes an inventory of the functionality
     exposed by the application.

Given  a new scanning session
When   spider the application
Then   the application is spidered

#assertions for security pattern testing (checker whether
     the pattern consequences are observed)
Then  Output Guard security pattern is present
Then  Input Guard security pattern is present
```

Figure. 6. An example of Feature file

```
@When("^spider the application$")
public void theApplicationIsSpidered() {
// Try one of the following techniques :
//   1. Spider web sites for all available links
//   2. Sniff network communications with application using
     a utility such as WireShark.

getSpider().setMaxDepth(10);
getSpider().setThreadCount(10);
url = "URL to be scanned";
try {
spider(url);
} catch (InterruptedException e) {
e.printStackTrace();}
waitForSpiderToComplete();}
```

Figure. 7. The procedure related to the When section of Figure 6

Gen de tests Cucumber

Résultats en cours

- 2 publication sur l'inférence de mdl,
  - Afadl(grenoble), Icsoft (portugal)
- 3eme en cours (ictss)
- Prototype d'outil + étude de cas (traces, exemple de mdls)
  - https://github.com/Elblot/COnfECt

Indutriels ?

- Qu'attendez vous du projet ?

- Transfert de savoir ?
  - Bases ? (CVE ? Etc.) méthodologie ?
- Etude / travail commun ? (stages ?)
- Outils ?