
defants
COLLABORATIVE & AUTOMATED SOLUTIONS
REDOCS 23

REDOCS

defants
COLLABORATIVE & AUTOMATED SOLUTIONS

Identification des erreurs de synchronisation temporelle dans les logs, et correction.

B.Beltzer, E.Klein, M.Racouchot, J.Soulé, O.Zari
en collaboration avec J.Duchêne

24/11/2023

UDS

Inria

GRENOBLE INP UGA

EURECOM
Sophia Antipolis

Introduction

Defants vSIRT

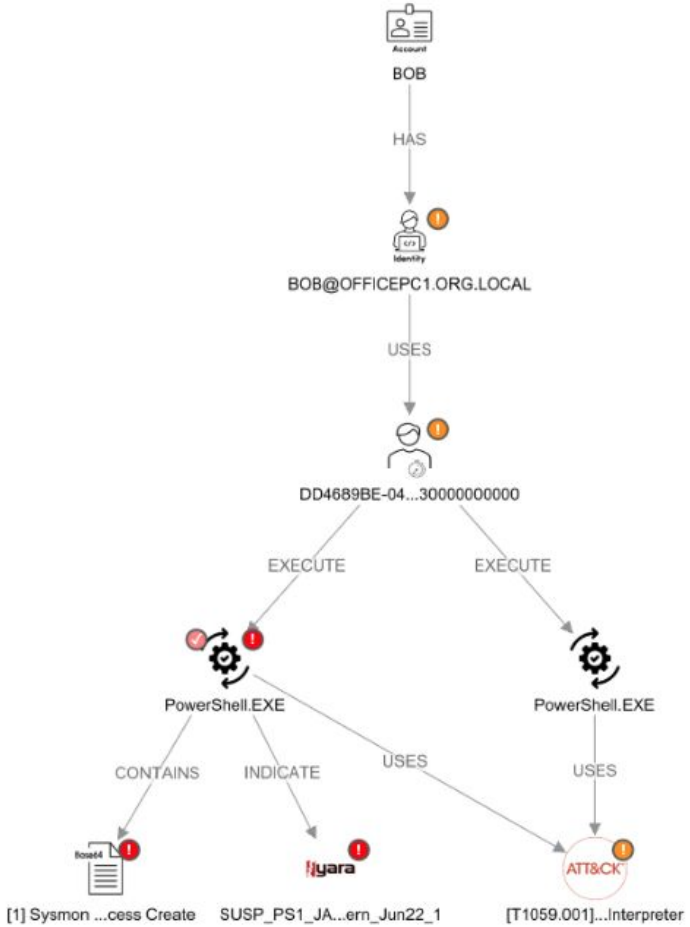
Principes

- Collecte d’informations par différents outils *(microservices)*
- Tri des informations selon des indicateurs de compromission
- Création d’un graphe sémantique d’investigation
- Identification des tactiques d’attaque *(MITRE ATT&CK framework)*

Defants vSIRT

Objectifs

- Automatisation de l’analyse
- Agrégation de données hétérogènes
- Visualisation de flots d'événements
- Optimisation de la recherche d’attaque



Problèmes généraux

- Chronologie des logs
  - Désynchronisations temporelles
  - Microservices sur différentes timezone
  - Absence de timestamps
- Séquence d’action
  - Enchaînement d’actions incohérentes
- Altération
  - Modifications par l’attaquant
  - Suppressions par l’attaquant

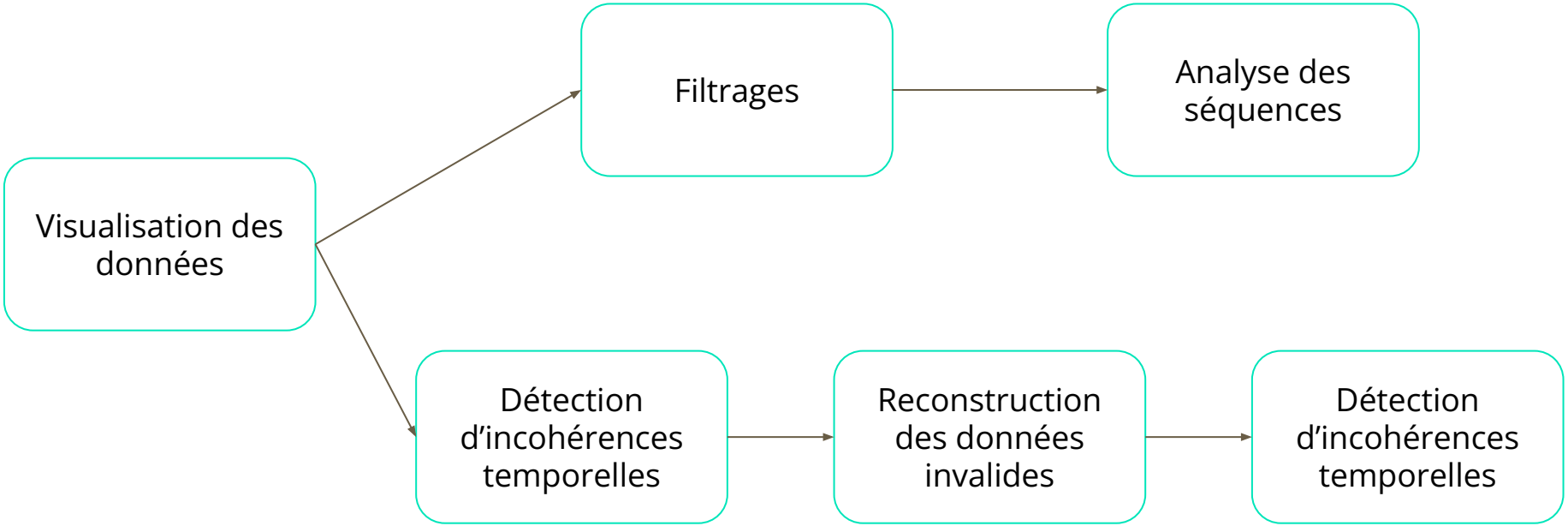
Demandes pour la semaine

- Estimer les temporalités des timestamps manquants
- Identifier si un attaquant a modifié des logs
- Identifier des activités suspectes/incohérentes
  - Incohérences temporelles
  - Incohérences séquentielles

Nos résultats

- Visualisation graphique des données
- Analyses de séquences
    - Filtrage linéaire
    - Analyse d’occurrence
    - Pattern matching
- Analyses temporelles
    - Filtrage linéaire
    - Détection d’incohérences temporelles
- Reconstruction et estimation des données temporelles invalides

Scénario d'utilisation



Analyse des données

Données fournies

- 2 fichiers json
  - nœuds (469,5 MB)
  - arêtes (413,3 MB)
- Environ 1 million de nœuds et 1 million d'arêtes
- Données anonymisées

Nœuds : informations importantes

- uuid : identifiant unique
- Timestamp :
  - First seen : première occurrence du nœud dans les logs
  - Last seen : dernière occurrence du nœud dans les logs
  - Seen : nombre d'apparition du nœud dans les logs
- Type : nature de l'acteur/effet (système, fichier, script…)

uuid
b505066d-3723-4737-a807-26d533df8634
First seen
27 March 2021, 14:06:28
Last seen
29 March 2021, 14:06:28
Seen
3
Type
PowerShell.exe

Remplissage du timestamp

Seen = 4

First seen

Last seen

Déroulement du log

Cas particuliers

First seen
=
Last seen

First seen, last seen
=
Date actuelle

Arêtes : informations importantes

- uuid : identifiant unique
- From : identifiant du nœud qui génère l'arête
- To : identifiant du nœud sur lequel pointe l'arête
- Timestamp : moment d'apparition de l'arête dans les logs
- Reason : nature de l'action représentée par l'arête

Visualisation

-**O.Z**. Put here the graphs statistics

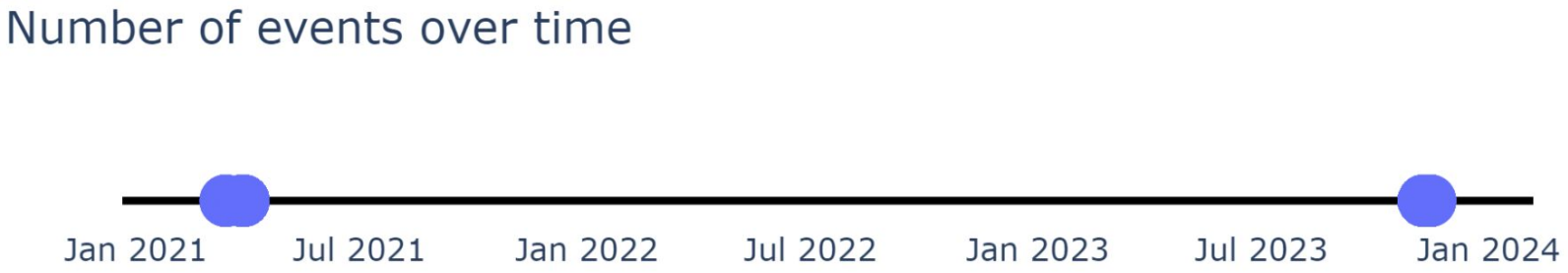
- Figure to show the degrees.

- Figure to show the stats of semantics.

- Figure to show the event rate activity —> introduce the concept of dummy timestamps and how it’s done(after 2023)

- Show stats on dummy timestamps(edges and nodes)

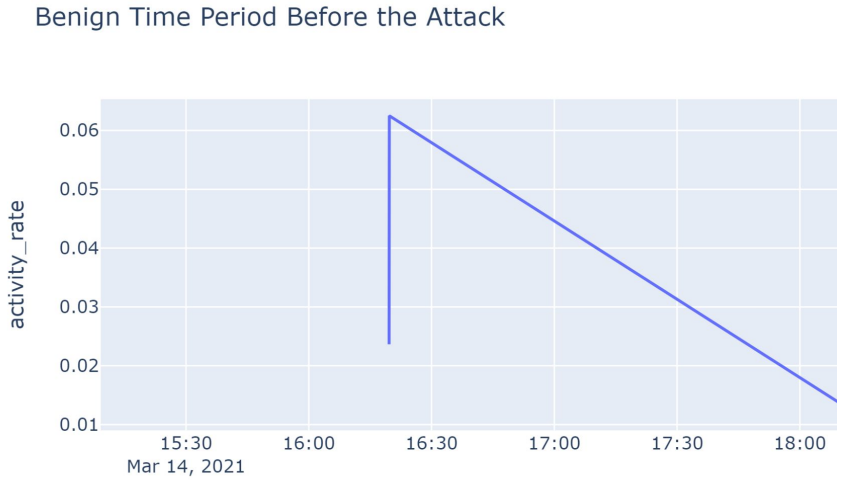
Concentration des timestamps d’actions

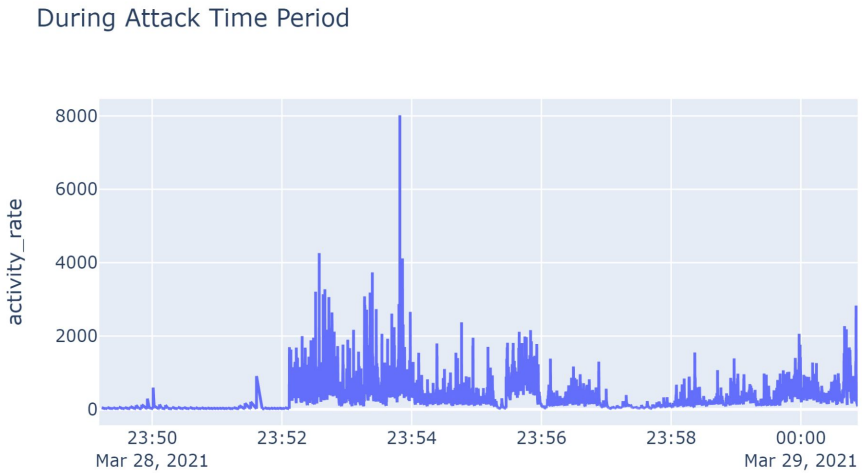


Arêtes datées
199966 / 993607
20.1%

Arêtes non-datées
793641 / 993607
79.9 %

Taux d’activité

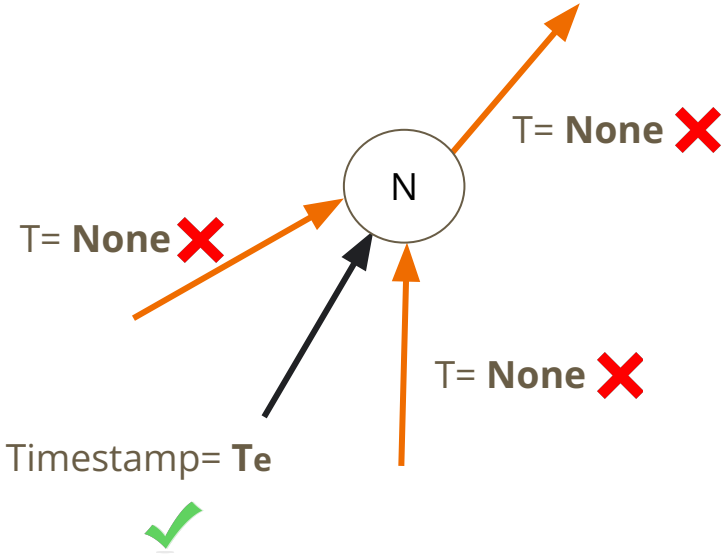




fenêtre = 100 activités consécutifs

Taux d’activité = #activités dans la fenêtre / Temps total pour la fenêtre

Noeuds non-datés



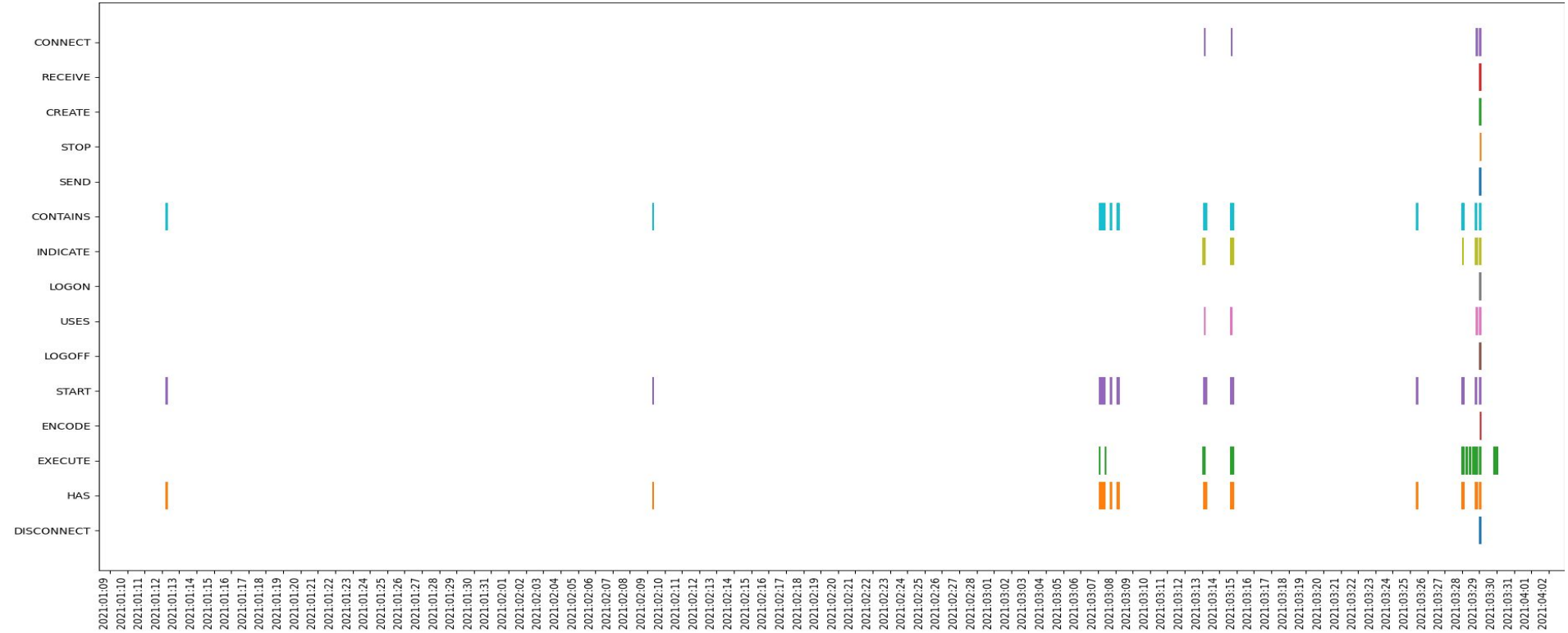
Put Stats Here

Nœuds non-datés

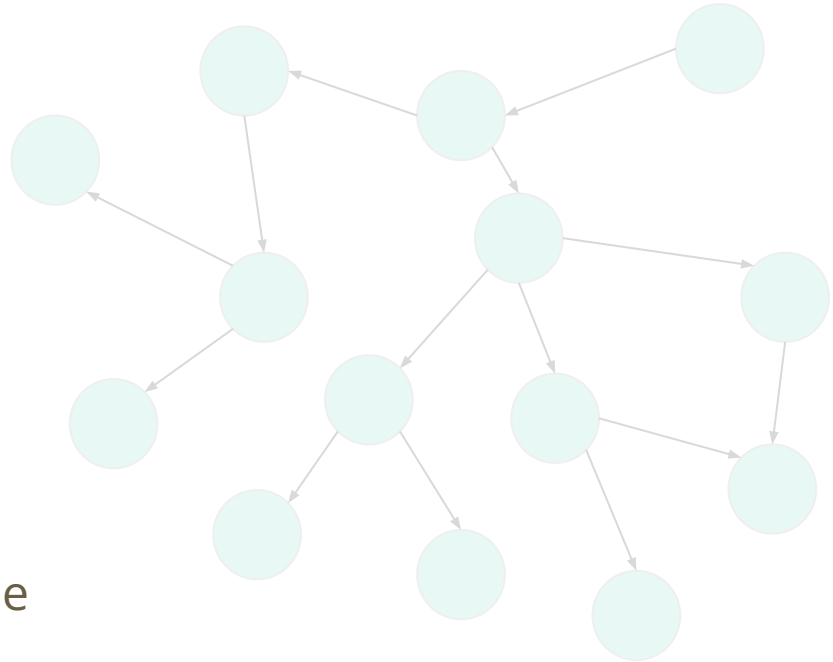
#Dummy_edges = 793641

= 97895 (12.3%)

Visualisation des actions dans le temps



Structure de données



3 objectifs :

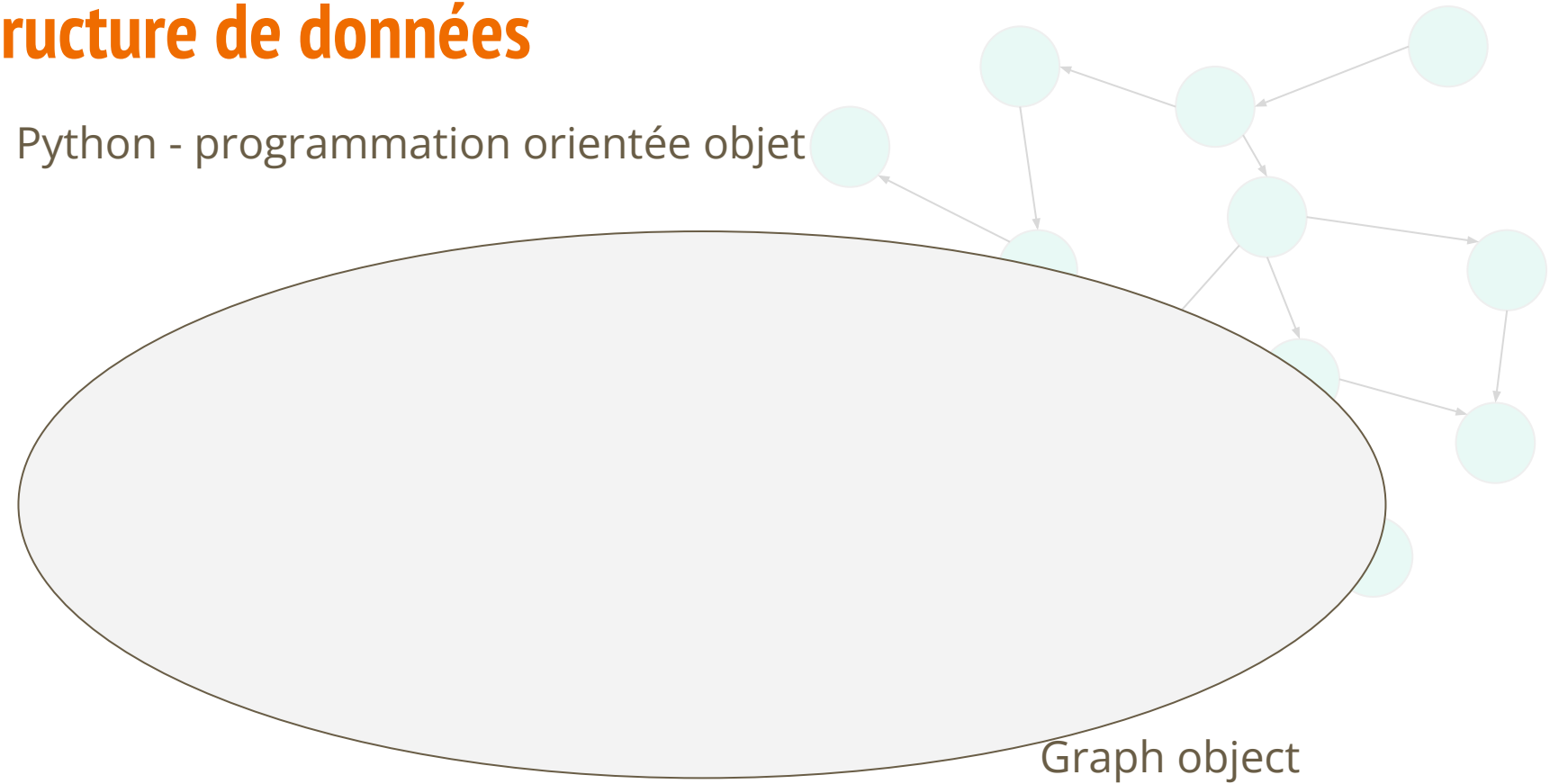
- Proposer un cadre de travail
- Organiser le code
  - Maîtriser les complexités
- Fournir une abstraction sémantique

Structure de données

Python - programmation orientée objet

Structure de données

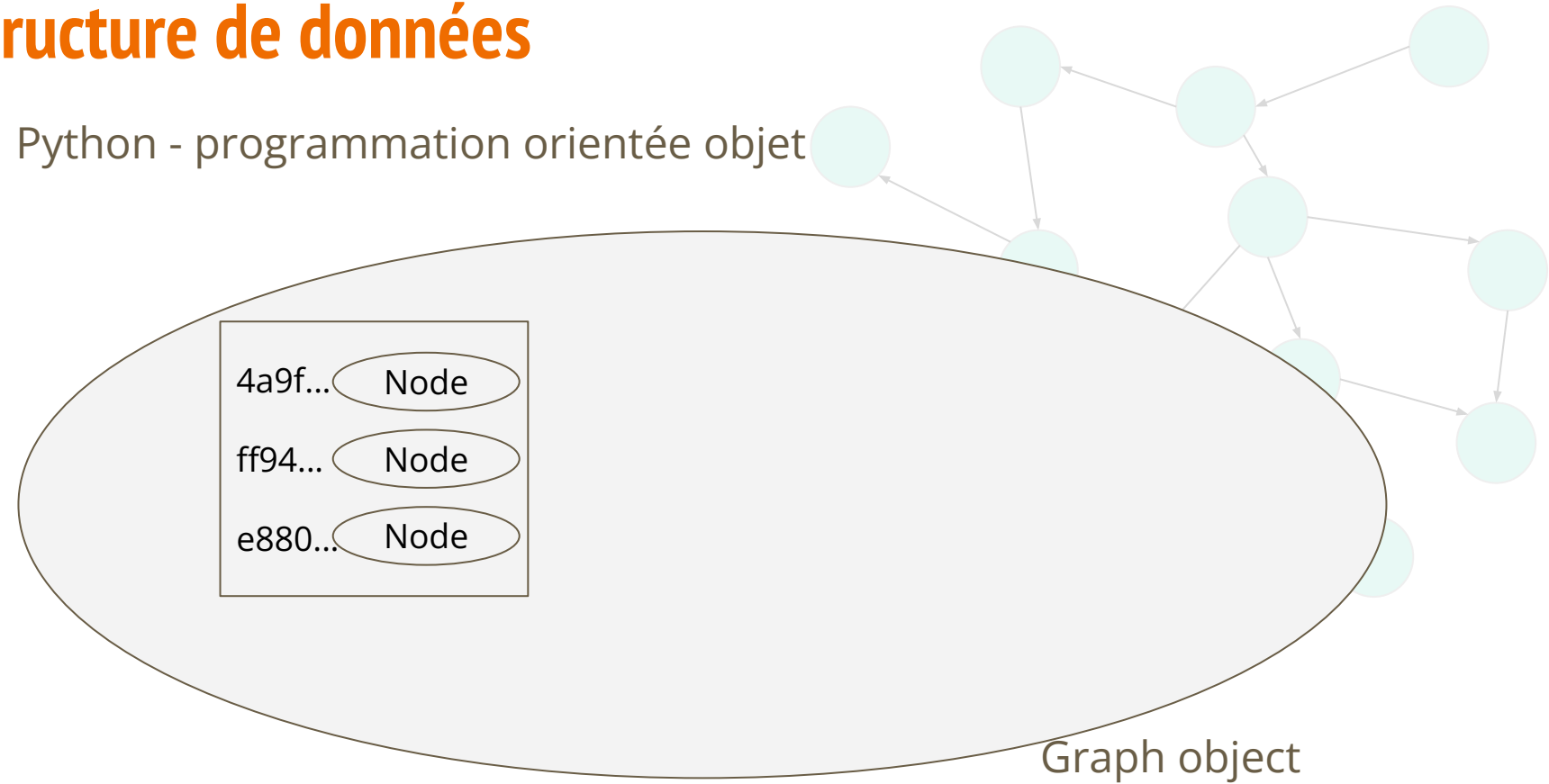
Python - programmation orientée objet



Graph object

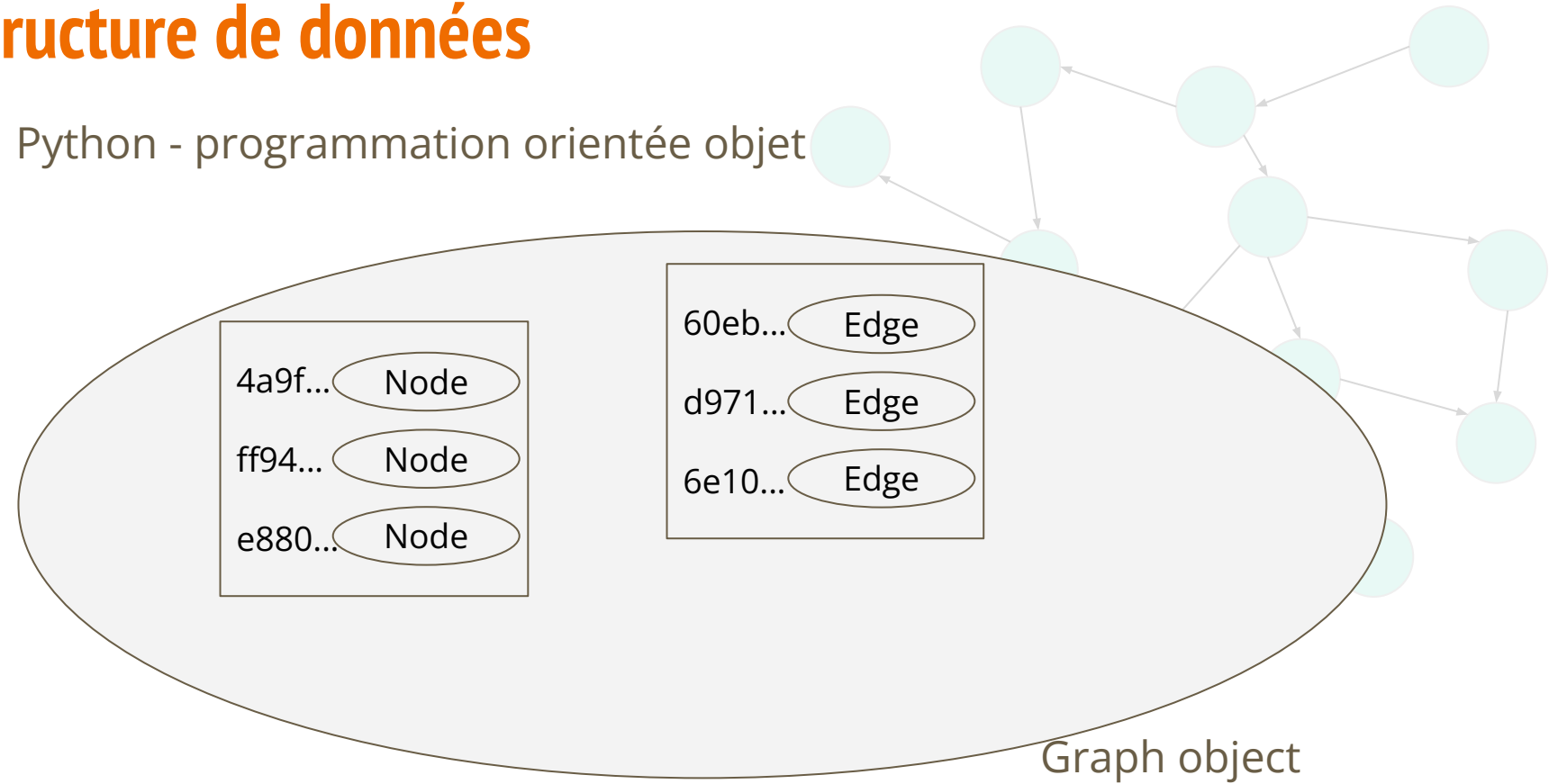
Structure de données

Python - programmation orientée objet



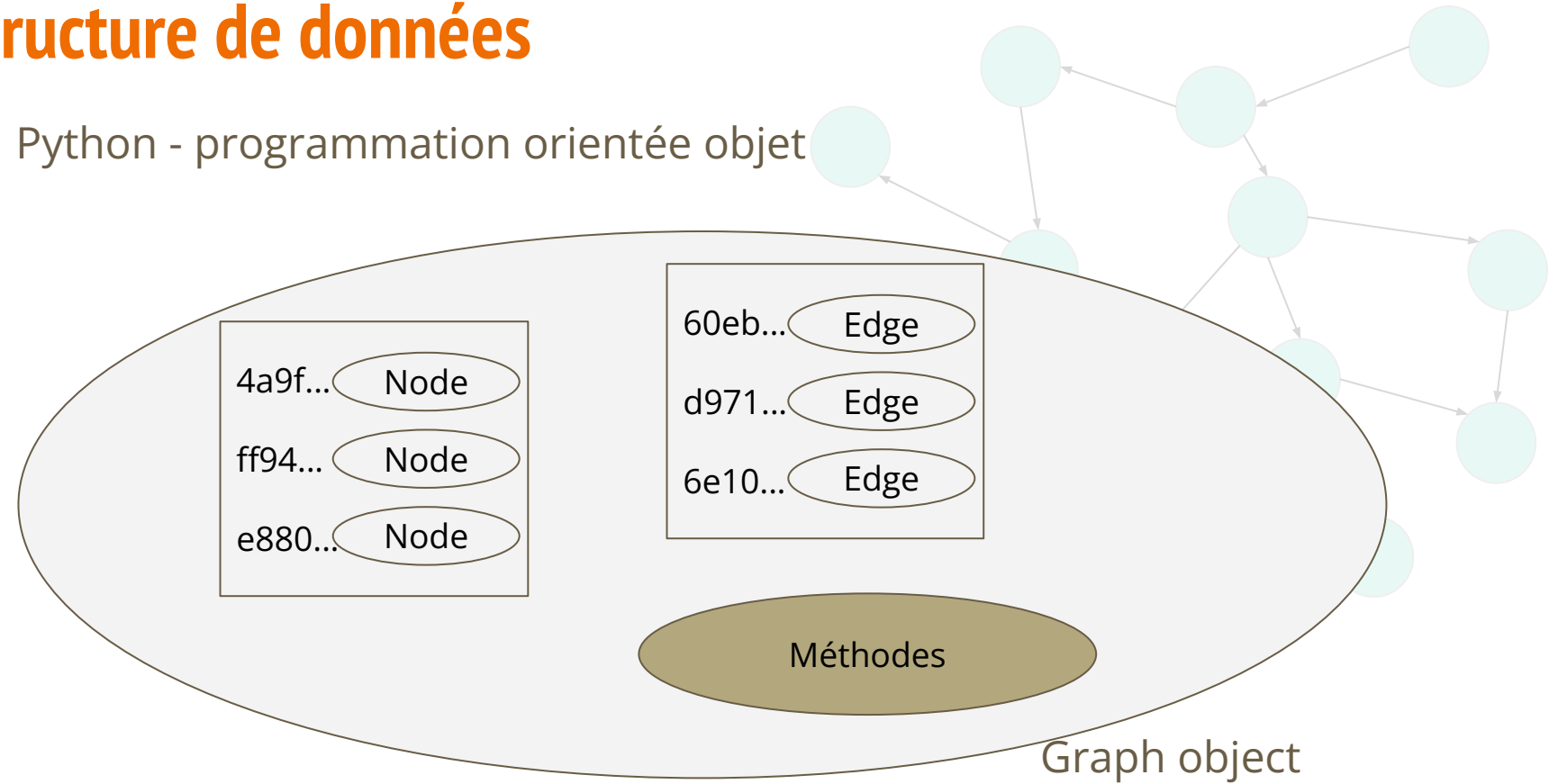
Structure de données

Python - programmation orientée objet



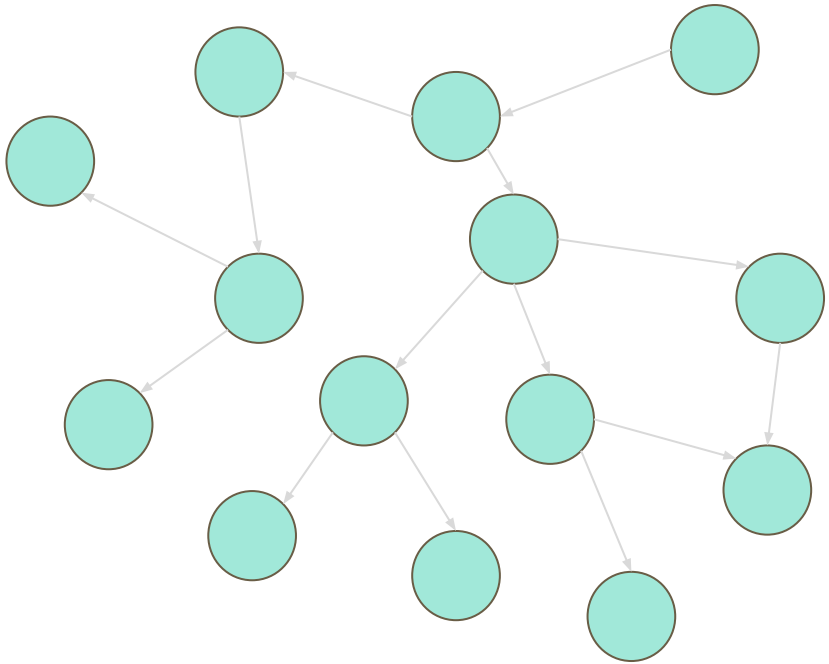
Structure de données

Python - programmation orientée objet



Structure de données

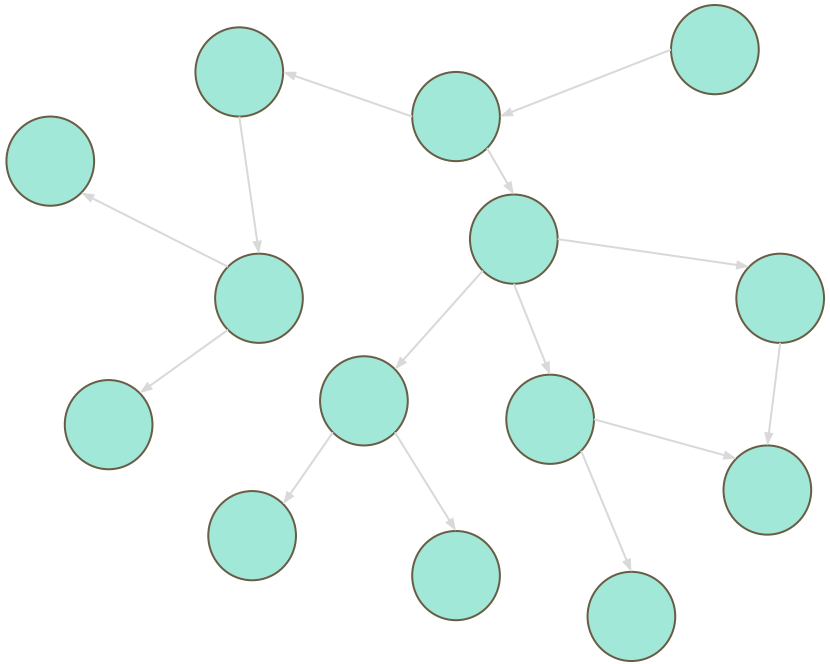
Structure de données



Structure de données

Node

- uuid
- type
- timestamp
  - First Seen
  - Last Seen

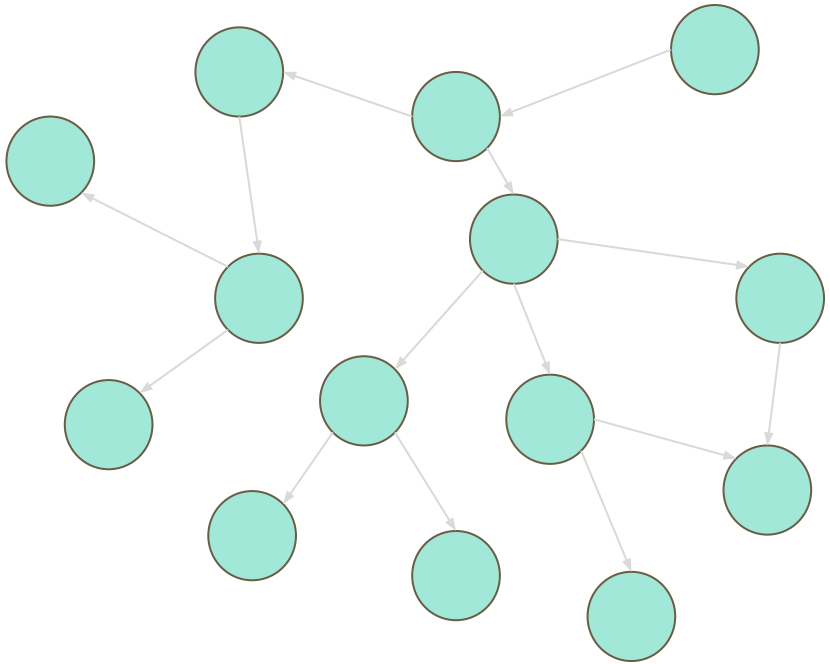


Structure de données

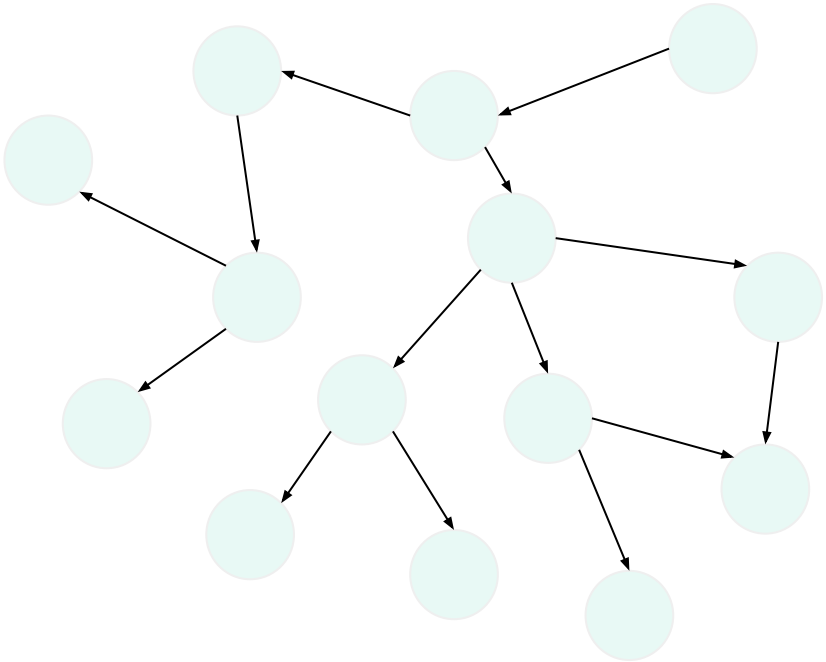
Node

- uuid
- type
- timestamp
  - First Seen
  - Last Seen

- Méthodes
  - Getter/Setter



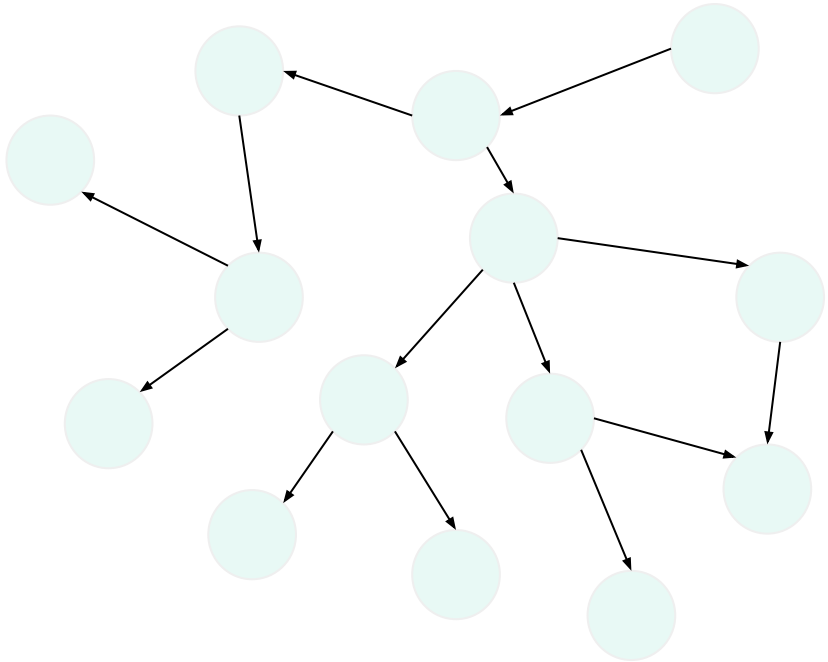
Structure de données



Structure de données

Edge

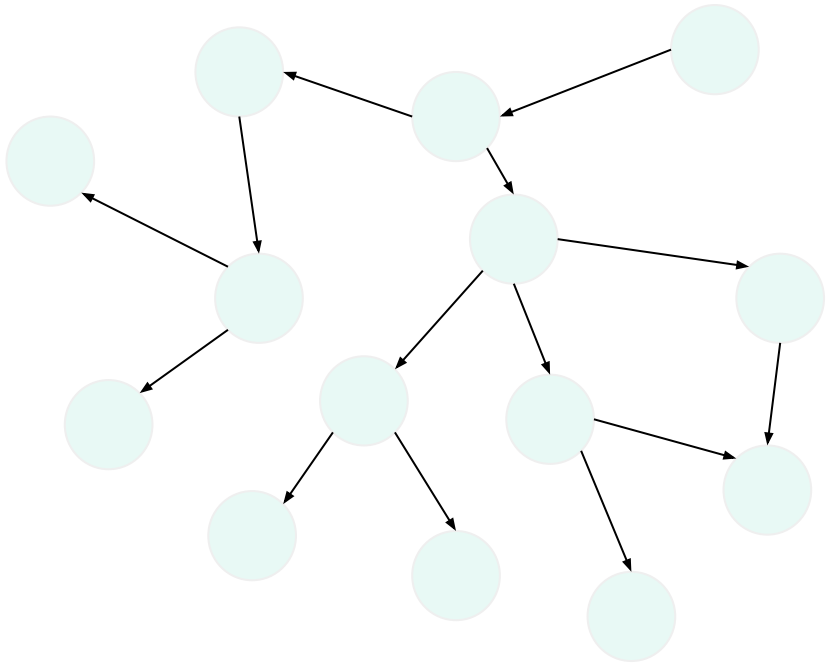
- uuid
- reason
- node_from_uuid
- node_to_uuid
- timestamp



Structure de données

Edge

- uuid
- reason
- node_from_uuid
- node_to_uuid
- timestamp



- Méthodes
  - Getter/Setter
  - Accès aux nœuds voisins
  - Opérateurs de comparaison de timestamp

Structure de données

Structure de données

Timestamp

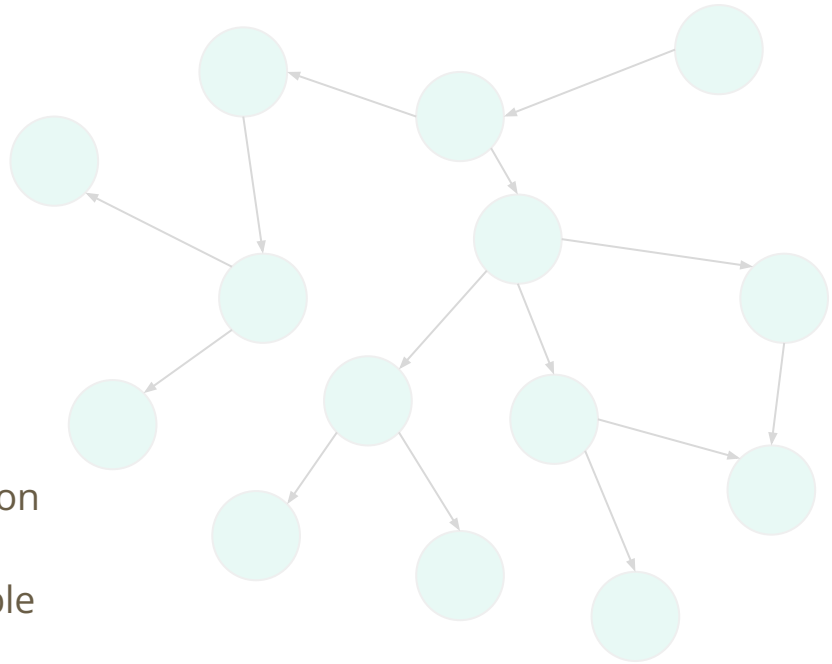
- nanosecondes

Structure de données

Timestamp

- nanosecondes

- Méthodes
  - Opérateurs de comparaison
  - Traduction
  - Formatage human readable
  - Gestion des timezones



Filtrage

Principe

Demande de l’entreprise : filtres linéaires sur le dataset

Structure de données :

- Abstraction sémantique
- Accès facile aux caractéristiques des objets

⇒ Facilité d’écriture des filtres !

Complexité : O(#E)

Exemples

- “Les horaires de la boîte, c’est 7h-21h”
  - pour toute arête, si timestamp de l’arête n’est pas entre 7h et 21h:
    - log : “l’arête uuid(...) apparaît en-dehors des heures autorisées”
  - 195 991 logs en-dehors de la plage horaire : 🤔

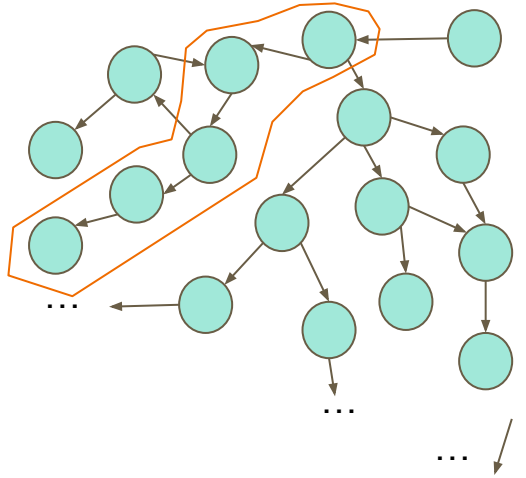
- Fileless attack : exécution de macro malveillantes dans un .xlsx ou .docm
  - pour toute arête, si reason_arête = CONNECT et type_nœud_parent = document:
    - log : “attention, le nœud uuid(...) fait l’action uuid(...)”
  - 0 log : ouf !
  - Possible avec autre type et autre reason

Autres filtres intéressants

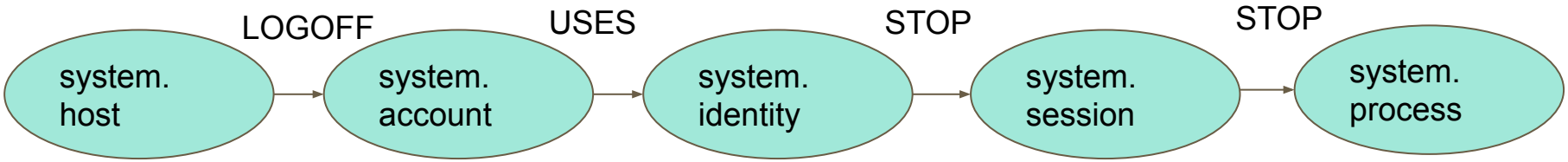
- Nœuds très actifs
  - pour tout nœud, si seen_nœud > 50:
    - log : “attention, le nœud uuid(...) est très actif (>50)
- Nœuds “data.entropy” (chiffrement)
  - pour tout nœud, si type_nœud = data.entropy:
    - log : “attention, le nœud uuid(...) a une entropie très élevée”
- Détection d’élévation de privilèges (system.group = “Administrator”)
- Repérage des nœuds “malwares” ou “indicators” (Windows Defender)

Étude des séquences d'actions

Occurrences des séquences d'action



- **Identifier des séquences d'actions**
  - Depuis un noeud et profondeur donnés



- Liste pondérée d'occurrences des séquences

  ⇒ Aide pour trouver séquences communes, suspectes, rares...

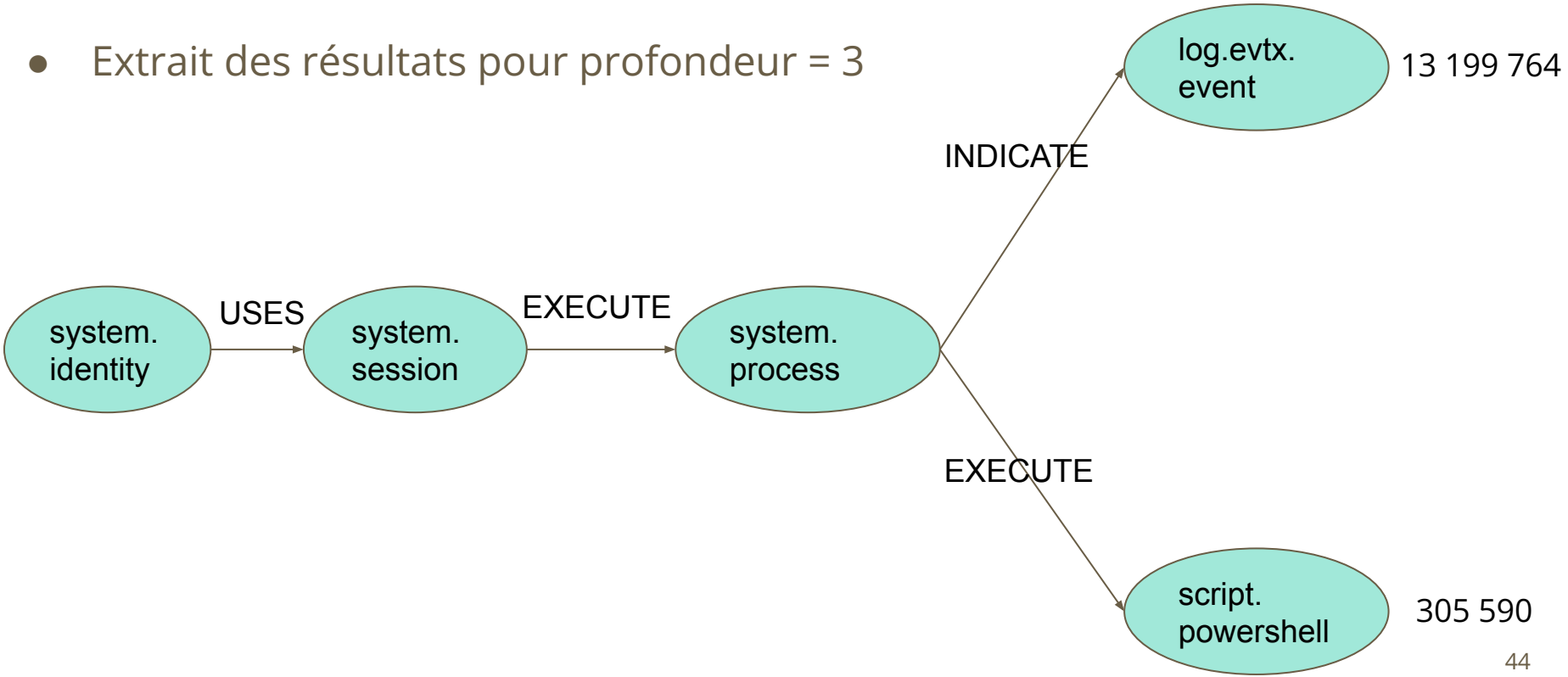
Occurrences des séquences d'action

- Approche récursive
  1. Choisir un nœud de départ
  2. Explorer et garder en mémoire les arêtes (actions) depuis ce nœud
  3. Appel récursif pour chaque nœud voisin
     3.1. Si profondeur atteinte, renvoyer séquence d'actions
  4. Compter le nombre de séquences identiques
     (optimisable en comptant à la volée)

- Complexité
  - $O(c^n)$
  - Estimation expérimentale : c ~ 4

Occurrences des séquences d'action

- Extrait des résultats pour profondeur = 3



Occurrences des séquences d'action

- Première approche pour exploitation des résultats
  1) Avoir une référence du nombre occurrences de séquences à partir de datasets bénins
  2) Analyse du nombre de séquence sur les logs à évaluer

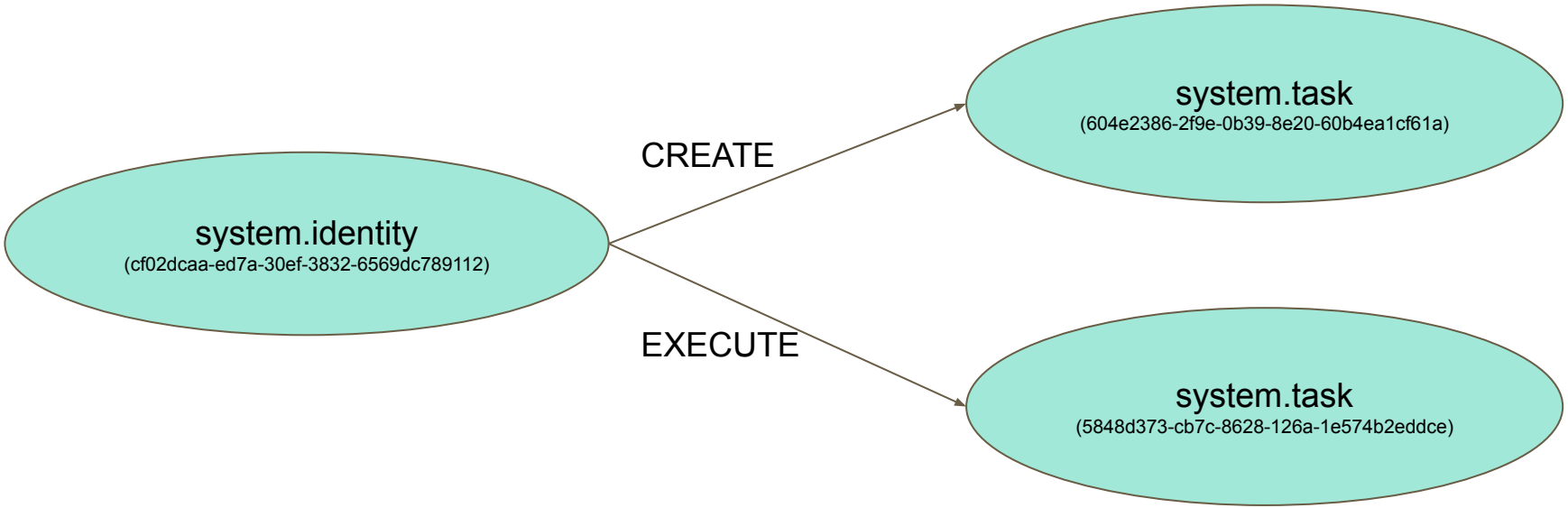
	Nombre occurrence important	Nombre occurrence faible
Séquence longue	*Activités régulières des utilisateurs ?*	*Attaque complexe sur une période longue ?*
Séquence courte	*Attaque répétant des mêmes actions sur un temps court ?*	*Activités journalières rares ?*

Matching des séquences d'action

- **Identifier des séquences d'actions à partir de pattern d'attaque**
- Choix format pattern : $[[A_1,A_2...][E_1,E_2...][B_1,B_2...]...]$
  - [] = n'importe quelle node/arête
- Même approche récursive
  - Avec utilisation des regex pour le matching de chaque séquence

Matching des séquences d'action

- Pattern: [["system.identity"],[],["system.task"]]



Détection des incohérences temporelles

Intégrité temporelle d'un graphe

Pourquoi ?

- Erreurs durant remontée des logs
- Données falsifiées par l'attaquant

Comment ?

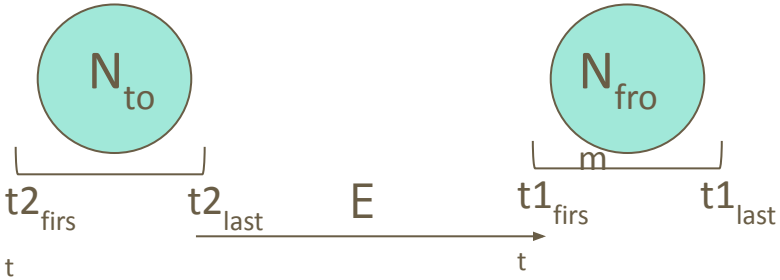
- Analyse de trio
- Séparation par code d'erreur

Erreur temporelle sur les nœuds

Err001 :

Err002 :

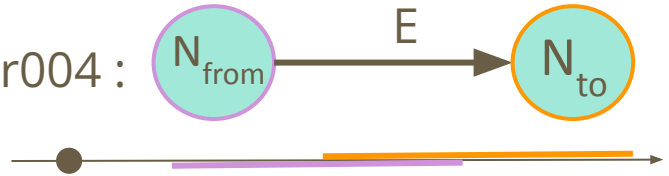
Err003 :



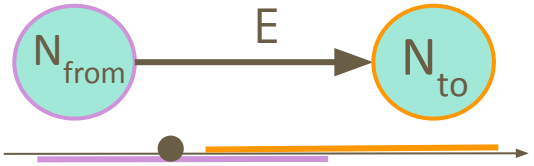
$t2_{last} < t1_{first}$

Erreur temporelle sur les arêtes

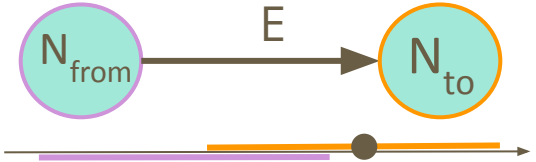
Err004 :



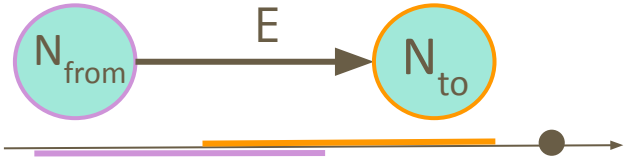
Err006 :



Err005 :



Err007 :



Résultats sur le dataset initial

Err001	509	< 1%
Err002	1 883	< 1%
Err003	28 849	2,9%
Err004	3 876	< 1%
Err005	2 418	< 1%
Err006	27 094	2,7%
Err007	6 542	< 1%

Résultats sur le dataset initial

		Daté + Non daté	Juste daté
Err001	509	< 1%	< 1%
Err002	1 883	< 1%	< 1%
Err003	28 849	2,9%	14,5%
Err004	3 876	< 1%	2%
Err005	2 418	< 1%	1,2%
Err006	27 094	2,7%	13,5%
Err007	6 542	< 1%	3%

Résultats sur le dataset initial

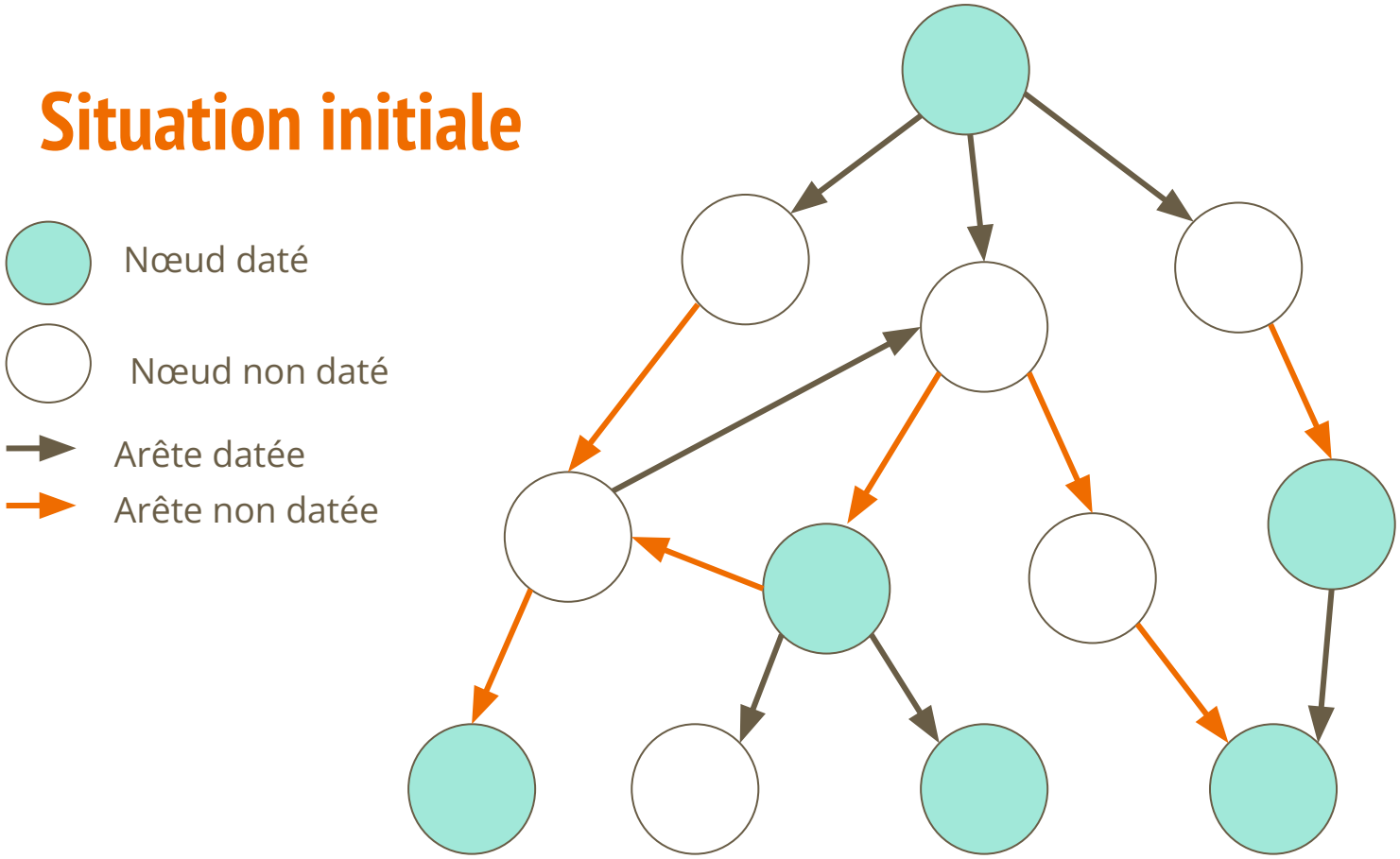
		Daté + Non daté	Juste daté
Err001	509	< 1%	< 1%
Err002	1 883	< 1%	< 1%
Err003	28 849	2,9%	14,5%
Err004	3 876	< 1%	2%
Err005	2 418	< 1%	1,2%
Err006	27 094	2,7%	13,5%
Err007	6 542	< 1%	3%

Fichiers de log explicites avec uuid pour analyse approfondie

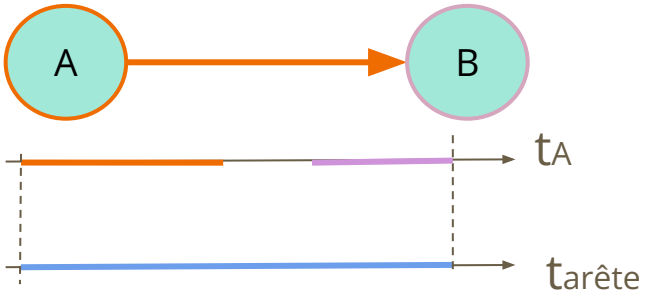
Complexité : O(#E)

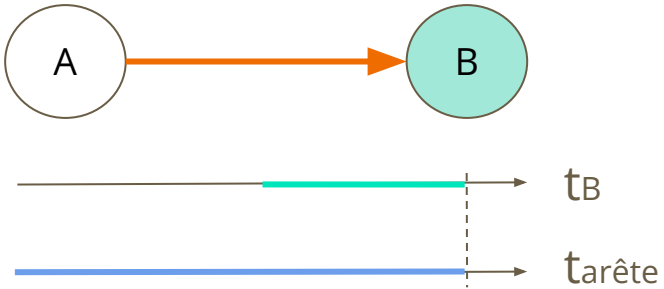
Remplissage temporel

Situation initiale

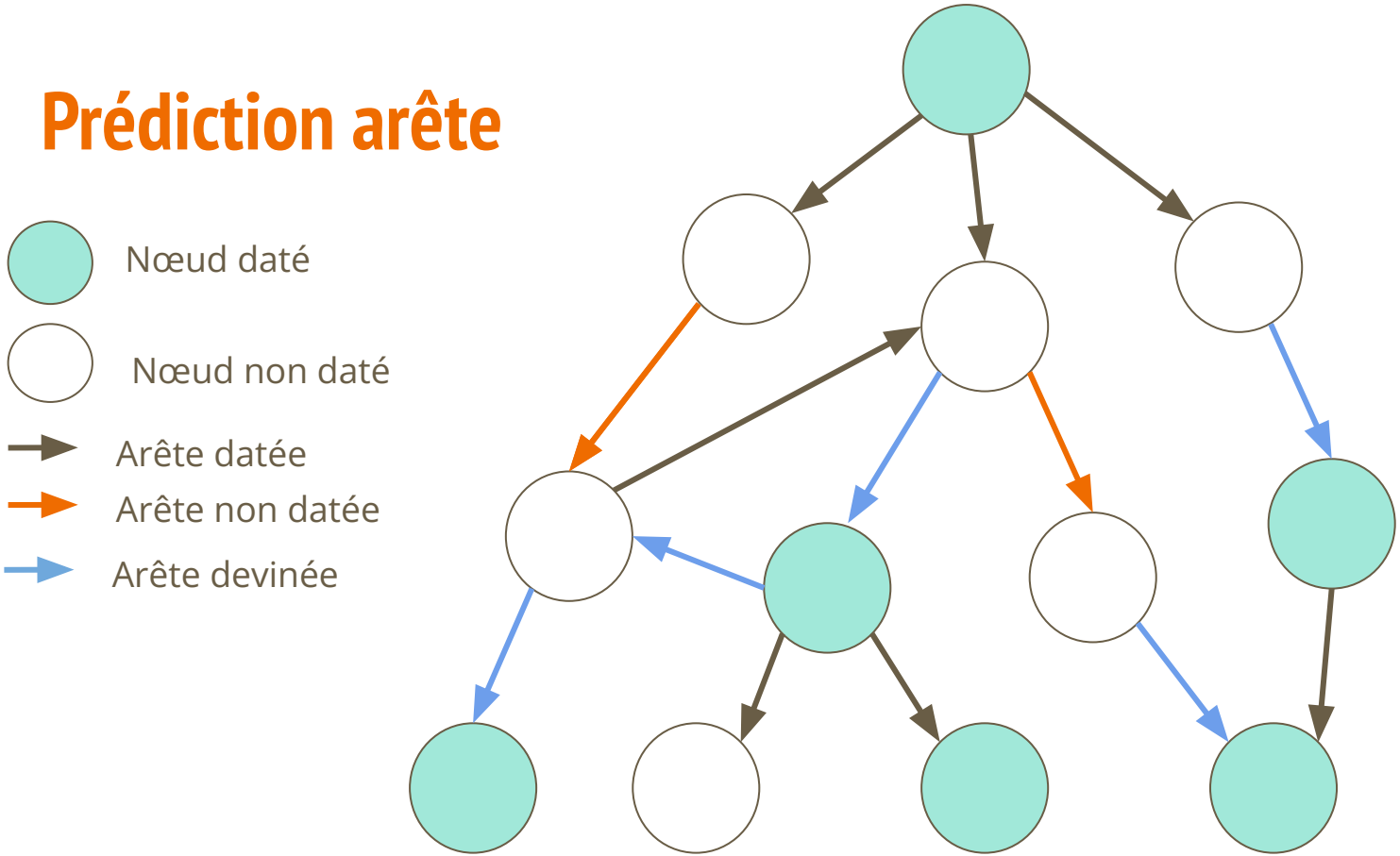


Remplissage: arête non datée

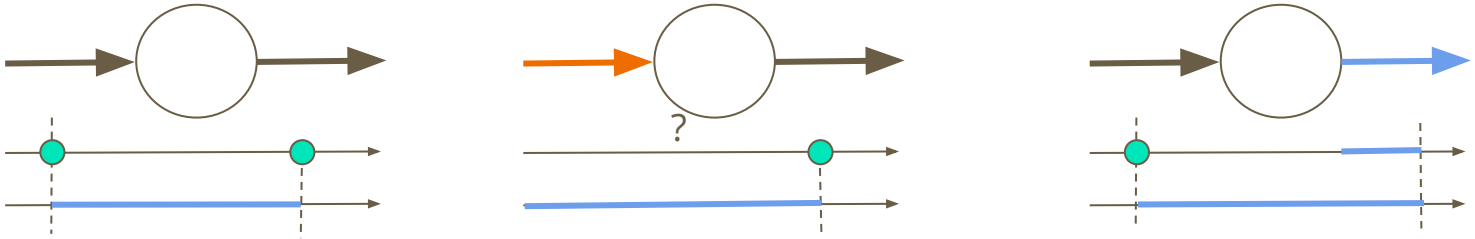


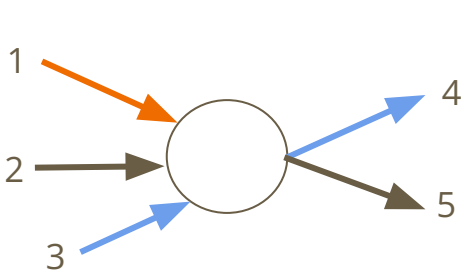


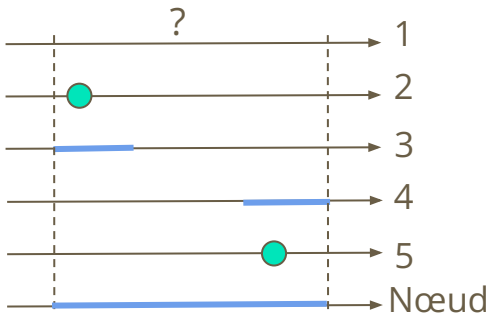
Prédiction arête



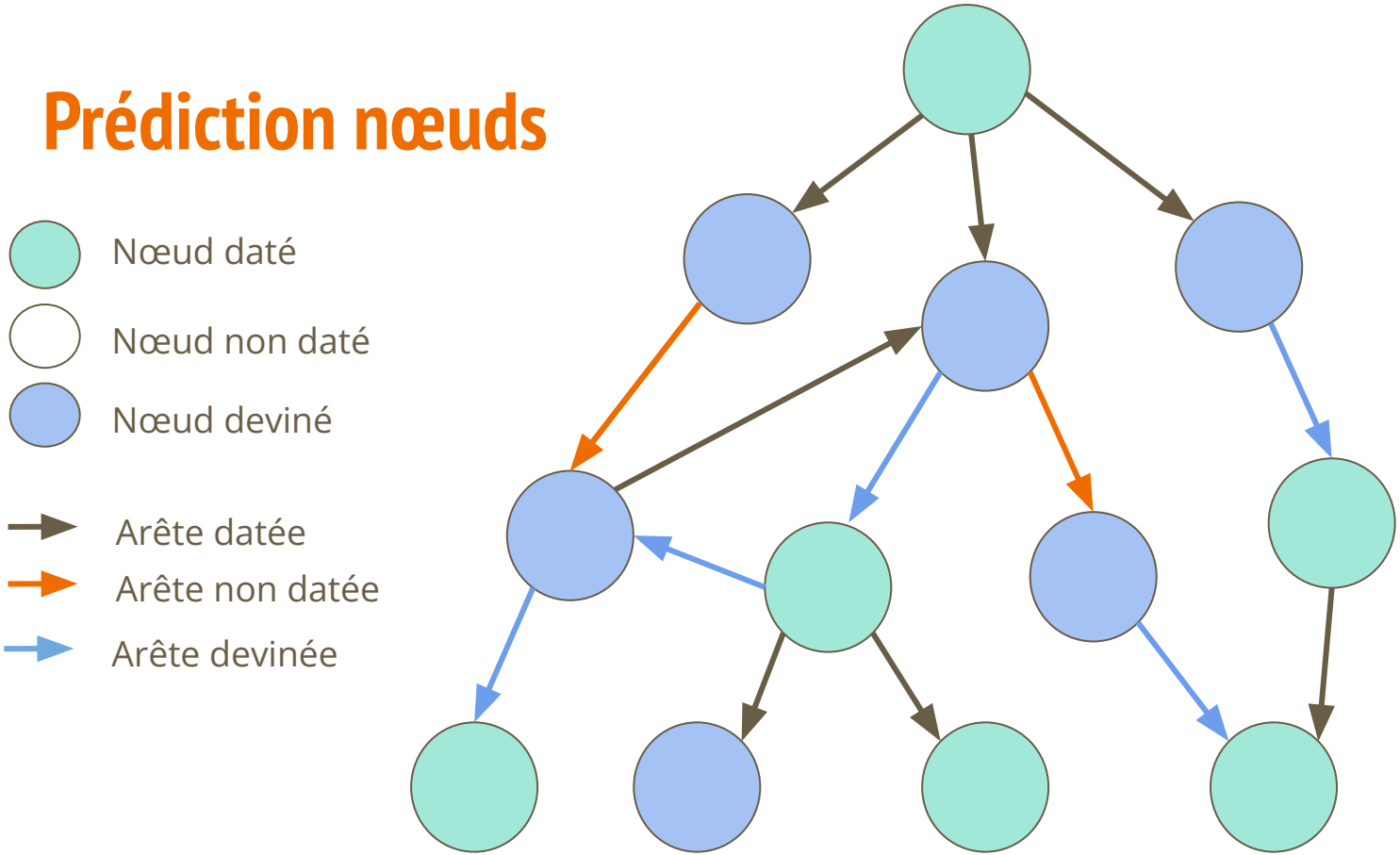
Remplissage: nœud non daté







Prédiction nœuds



Propagation

Nœud daté

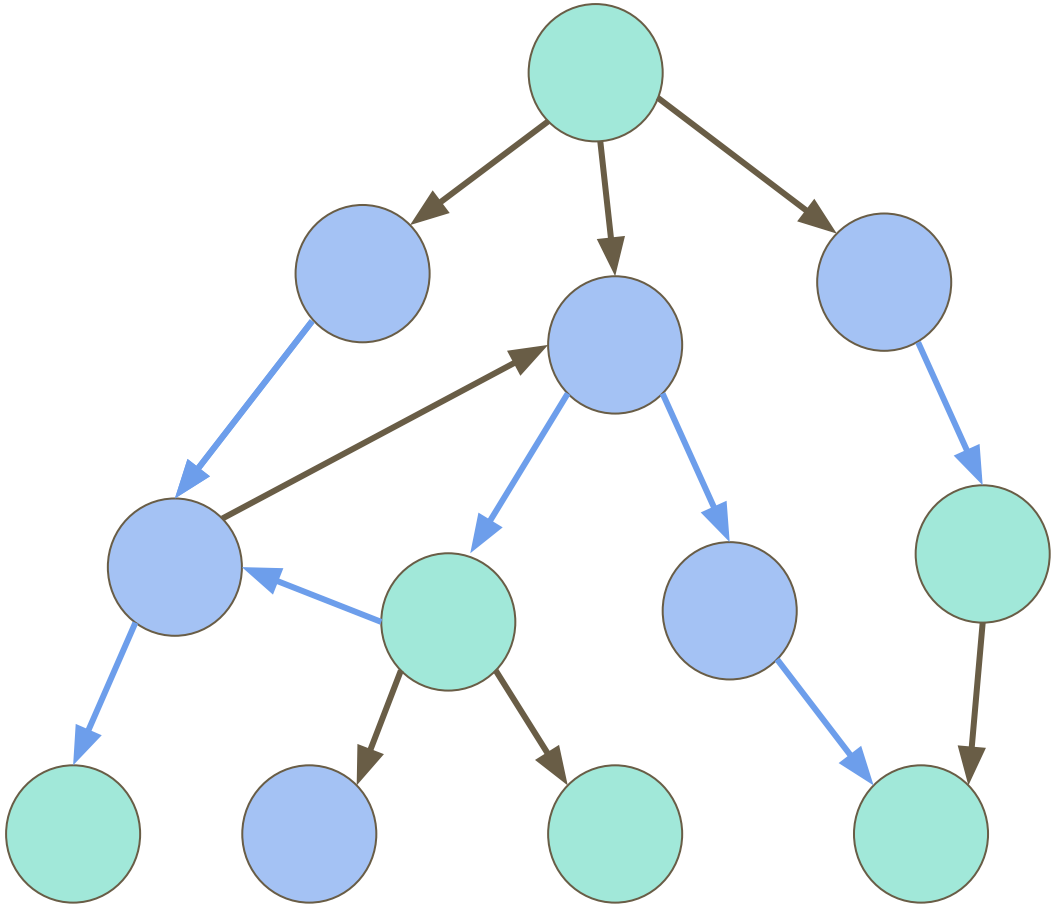
Nœud non daté

Nœud deviné

Arête datée

Arête non datée

Arête devinée



Détails sur l'algorithme

- Parcours sur les arêtes puis parcours sur les nœuds

- Mu = $\Sigma_{\text{nœuds}}$ (first_seen - last_seen) + $\Sigma_{\text{arêtes}}$ (first_seen - last_seen)
- Stop quand Mu ne bouge plus suffisamment (stabilisation)

- Complexité théorique: O(#E * #N)
- En pratique: beaucoup moins

Résultats

- 18 passages sur les arêtes et les nœuds
- Environ 5 minutes d'exécution
- Plus aucune information à propager
- Modifications sur 731 605 noeuds : 20,80% -> 79,19%
- Modifications sur 990 757 arêtes: 20,1% -> 99.71%

Nouvelle vérification de l'intégrité du graphe :

- Quelques swaps ajoutés (28 849 avant, 28 899 après)
- Sans doute propagés

Conclusion

Nos résultats

- Visualisation graphique des données
- Analyses de séquences
    - Filtrage linéaire
    - Analyse d’occurrence
    - Pattern matching
- Analyses temporelles
    - Filtrage linéaire
    - Détection d’incohérences temporelles
- Reconstruction et estimation des données temporelles invalides

Proposition de gestion des timestamps

- Laisser les champs manquants vides

  Ou

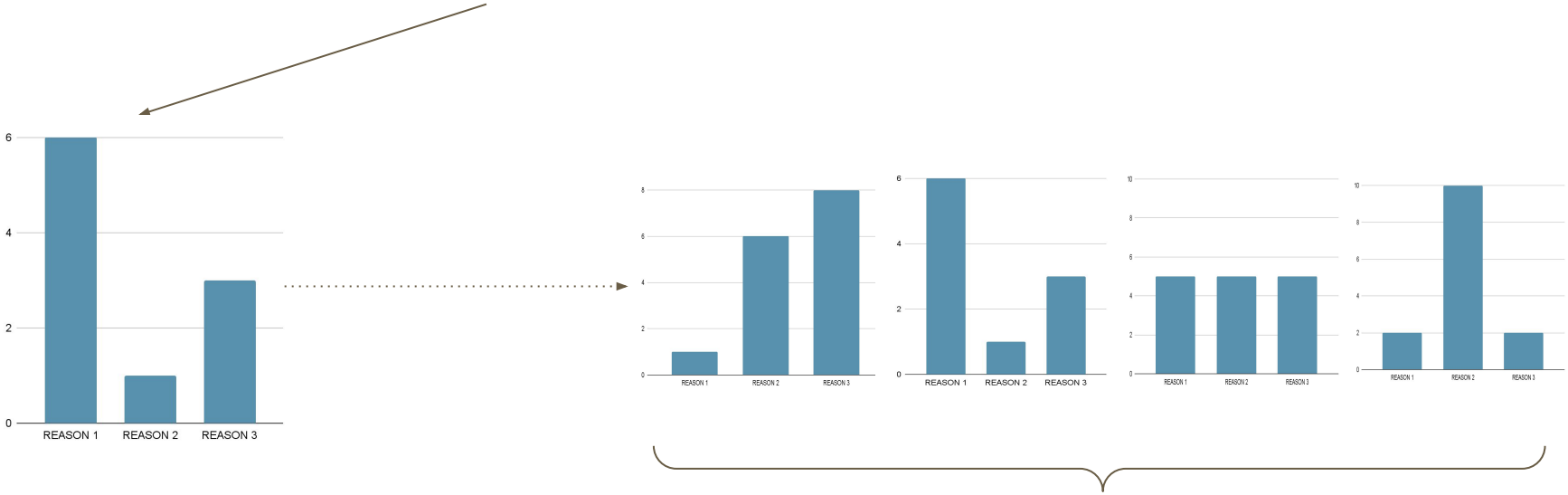
- Mettre une date unique pour l'analyse et la stocker quelque part

Gestion des timestamps manquants :

- Ajouter des champs "first estimated" et "last estimated" pour stocker l'intervalle deviné

Apprentissage non supervisé sur les séquences d'action

LOG ON -> … -> REASON 1 -> … -> REASON 2 -> … -> REASON 3 -> … -> LOG OFF



Séquences contexte bénin et malveillant
⇒ Apprentissage non supervisé

Travaux connexes dans la littérature

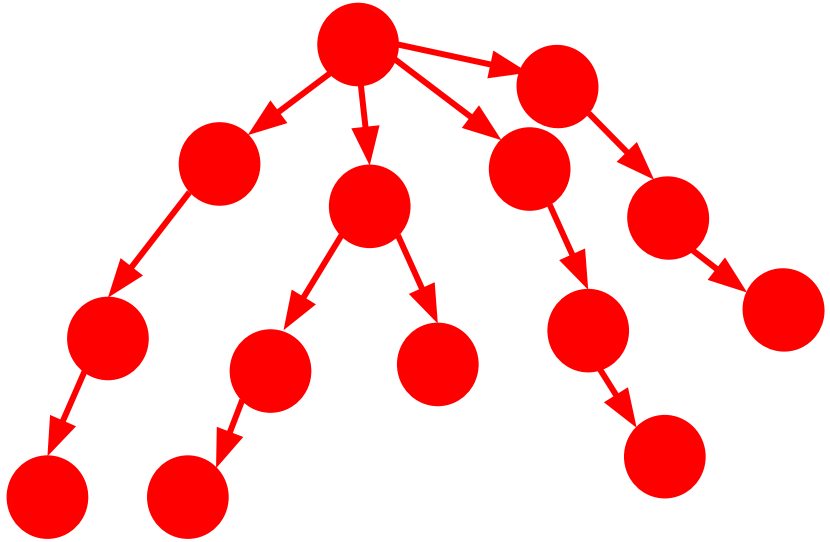
- **GraphTS**: Zarei, R., Huang, G., & Wu, J. (2023). GraphTS: Graph-represented time series for subsequence anomaly detection. Plos one, 18(8), e0290092.
    - Inclut une visualisation de série temporelle en 2D et construit un graphe basé sur la représentation spatiale 2D pour capturer l'ensemble des motifs récurrents et rares
    - Complexité en temps pas explicitement indiquée dans les résultats de la recherche

- **Series2Graph**: Boniol, P., & Palpanas, T. (2022). Series2graph: Graph-based subsequence anomaly detection for time series. arXiv preprint arXiv:2207.12208.
    - Basé sur une représentation en graphe de sous-séquences
    - Ne requiert pas de données étiquetées ou données bénignes et identifie anomalies de longueur variables
    - $O(3|T|(\ell - \lambda))$, où |T| est la longueur de la série temporelle, |component| est la taille d'un composant , l est la longueur de l'input et λ la longueur de la requête

- **STOMP**: Zhu, Y., Zimmerman, Z., Senobari, N. S., Yeh, C. C. M., Funning, G., Mueen, A., ... & Keogh, E. (2016, December). Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In 2016 IEEE 16th international conference on data mining (ICDM) (pp. 739-748). IEEE.
    - Méthode cherchant à identifier des motifs dans série temporelles
    - Exploite les performances du GPU et un nouvel algorithme pour adresser le problème du passage à l'échelle
    - Complexité pas explicitement mentionné dans les résultats de recherche

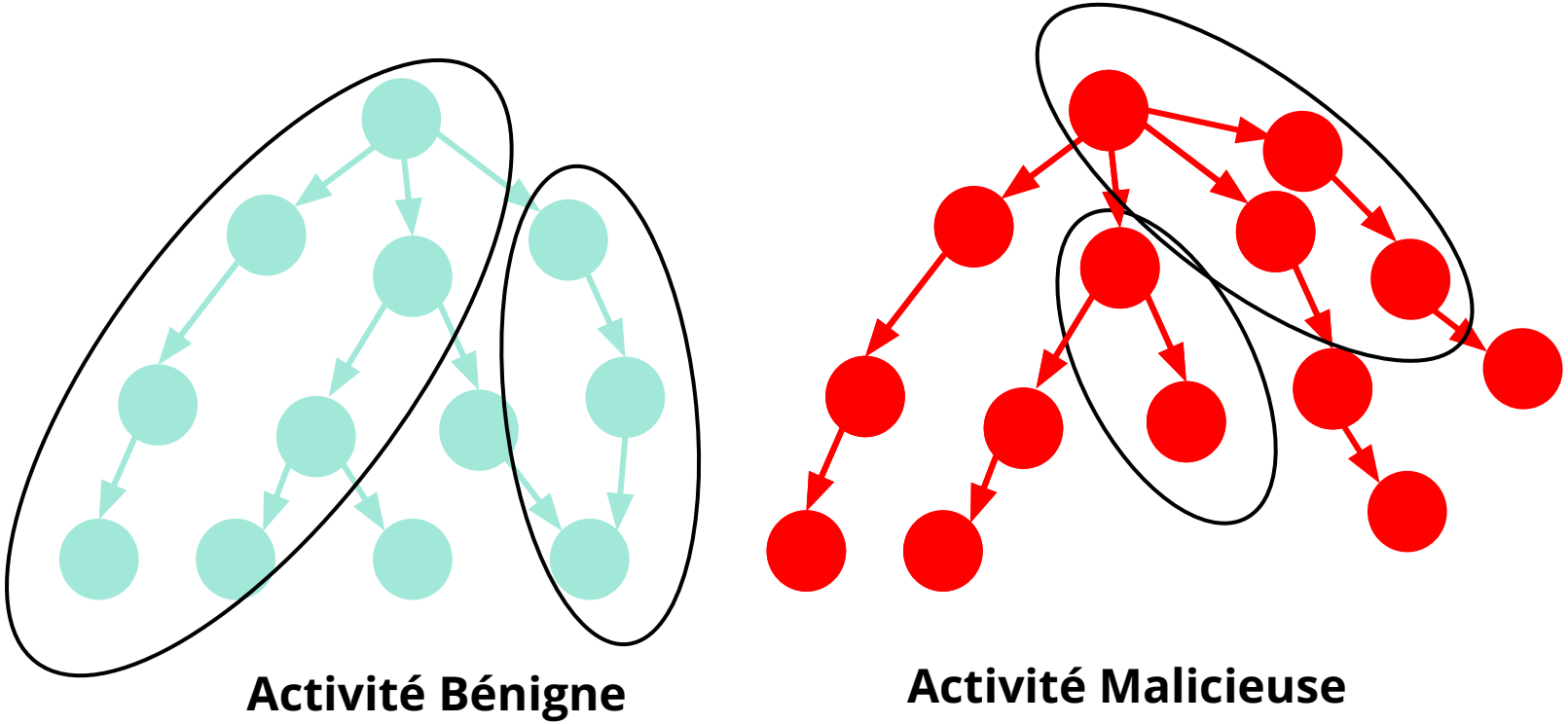
Approche Apprentissage Automatique

Activité Bénigne



Activité Malicieuse

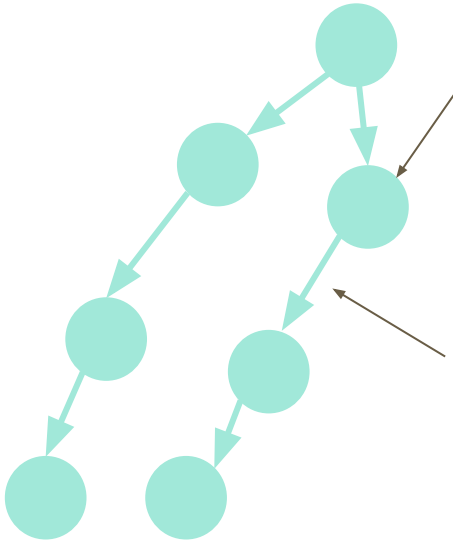
Approche Apprentissage automatique



Activité Bénigne

Activité Malicieuse

Approche Apprentissage automatique



X_N = [[0,0,0,0,0,1], ..., [0,0,0,1,0,0]]

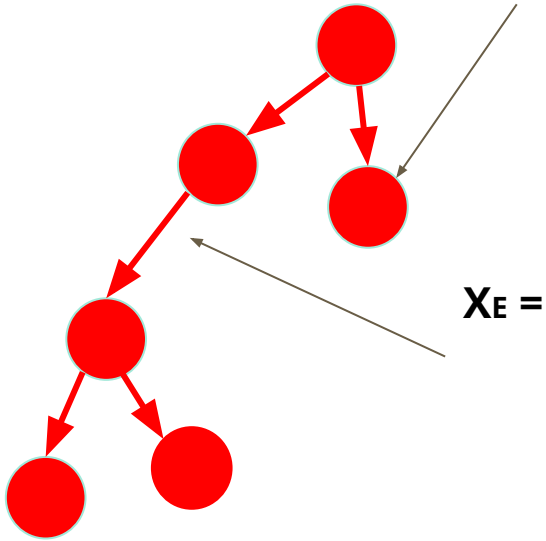
type **name**

X_E = [[0,0,1,0,0,0], ..., [0,1,0,0,0,0]]

reason **entity**

Activité Bénigne

Approche Apprentissage automatique



X_N = [[0,0,0,0,0,1], ..., [0,0,0,1,0,0]]
type **name**

X_E = [[0,0,1,0,0,0], ..., [0,1,0,0,0,0]]
reason **entity**

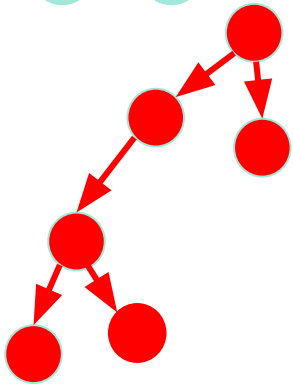
Activité Malicieuse

Approche Apprentissage automatique

X_N = [[0,0,0,0,0,1], ..., [0,0,0,1,0,0]]
type **name**

X_E = [[1,0,0,0,0,0], ..., [0,1,0,0,0,0]]
reason **entity**

Label = Activité Bénigne

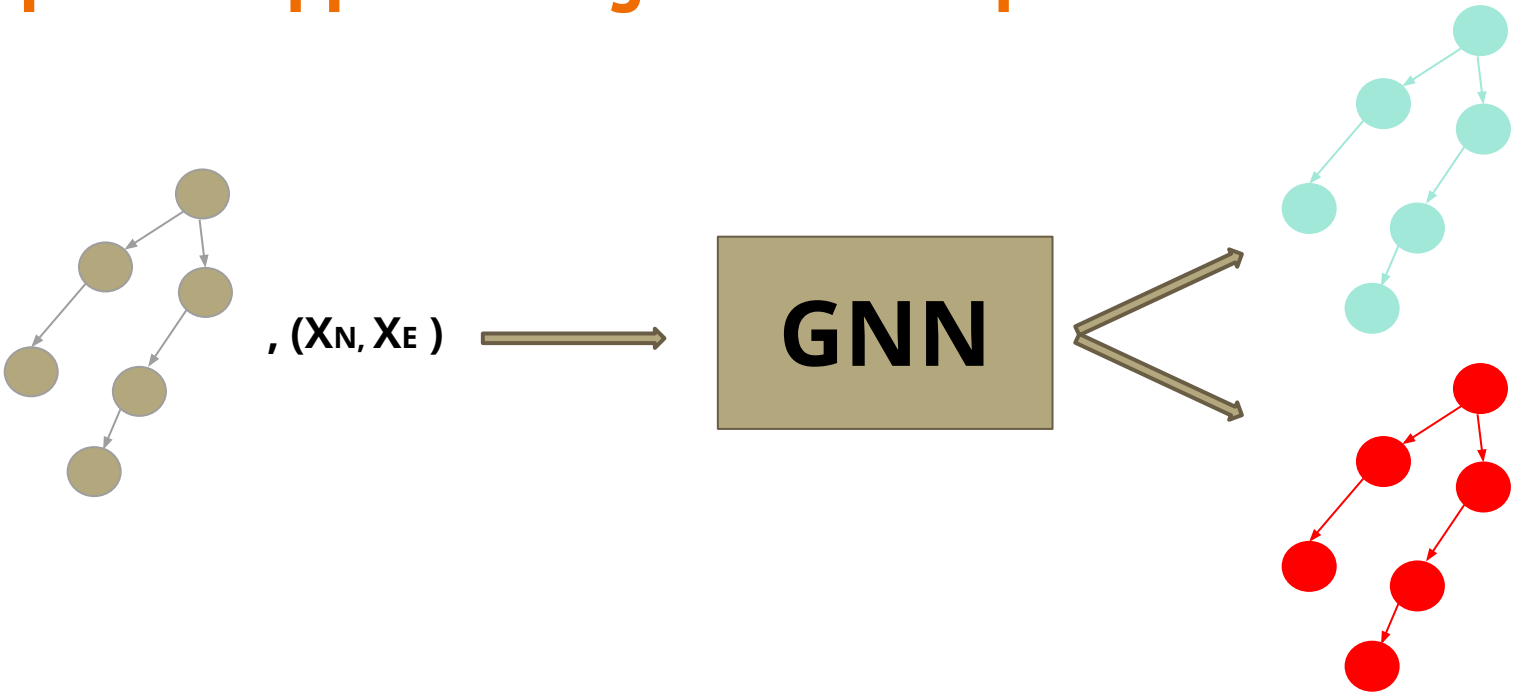


X_N = [[1,0,0,0,0,0], ..., [0,0,1,0,0,0]]
type **name**

X_E = [[0,0,0,0,0,1], ..., [0,0,0,0,1,0]]
reason **entity**

Label = Activité Malicieuse

Approche Apprentissage automatique



Li, Linjie, et al. "Relation-aware graph attention network for visual question answering." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
Schlichtkrull, M. et al. “Modeling Relational Data with Graph Convolutional Networks.” *Extended Semantic Web Conference* (2017).

Merci de votre attention !
Questions ?

Vérifier la cohérences des inférences sur les timestamps

1. Extraire les arêtes de la période d’attaque dans le dataset initial
2. Calculer la distribution des arêtes "actions" sur la période d’attaque
3. Inférer les timestamps pour les noeuds et les arêtes qui n’ont pas de données temporelles exacte
4. Extraire les arêtes sur la période d’attaque dans le dataset “inféré”
5. Calculer la distribution des arêtes "actions" sur la période d’attaque
6. Comparer les deux versions pour calculer la divergence
7. S’assurer que la divergence est plus petite qu’un threshold donné

Ébauche: Machine Learning pour la détection d'attaque

Pour un graphe initial obtenu sur une longue période de capture (qui contient des attaques et des périodes d'activité normales)

1. Créer un sous graphe "attaque" GA et un sous graphe "normal" GB
2. Échantillonner GA et GB de manière représentative (SA et SB)
3. Entraîner un GNN pour classer des éléments de SA comme True
4. Entraîner un GNN pour classer des éléments de SB comme False
5. Tester le GNN entraîné sur le reste de SA et SB