Vous avez un problème avec vos URLs mais vous ne le savez pas

Didier BRINGER didier.bringer@orange.com
Lionel TAILHARDAT lionel.tailhardat@orange.com
Antoine CAWET antoine.cawet@orange.com

2 Décembre 2025

orange™

AZUR TECH Winter

Teaser

Nous allons démontrer l'application des ontologies pour protéger Orange (ou votre société) contre l'usurpation de domaine, ainsi que le vol de cookie et conduire un audit DNS de qualité

Teaser

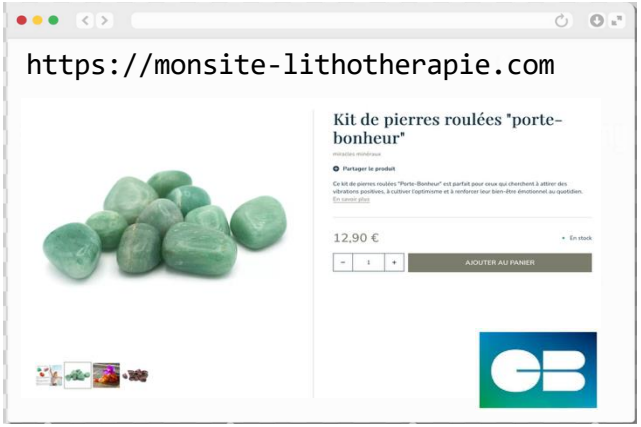
Defenders think in lists,

attackers think in graphs;

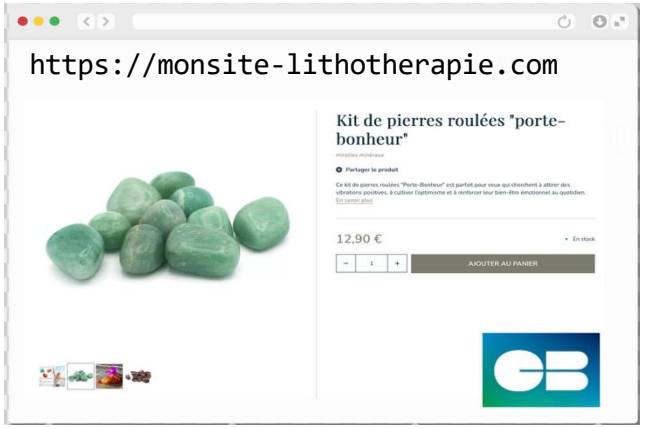
as long as this is true, attackers win.

John Lambert,a recognized professional in the field of cybersecurity and a member of Microsoft.

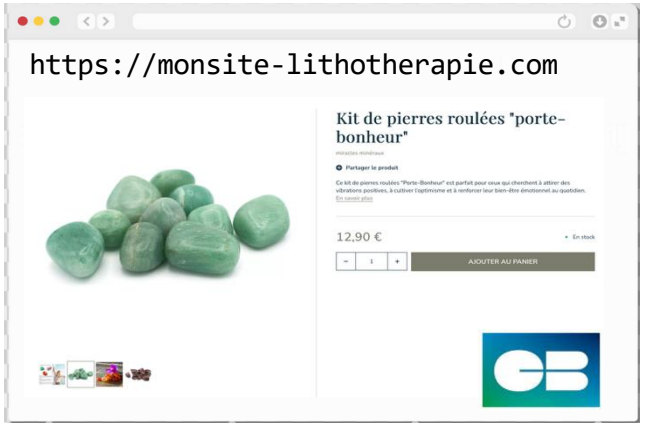
Didier,
tu sais quoi je me suis fait pirater !



IP: 195.101.12.12



IP: 31.16.1.12



?

… mes clients se sont fait pirater

Antoine,
je t'explique le DNS ...

```
monsite-lithotherapie.com IN CNAME monsite-lithotherapie.webagency.com.
monsite-lithotherapie.webagency.com IN A 195.101.12.12

monsite-lithotherapie.com IN CNAME monsite-lithotherapie.webagency.com.
monsite-lithotherapie.webagency.com IN A 31.16.1.12
```

Gérer un DNS … tout un savoir-faire !

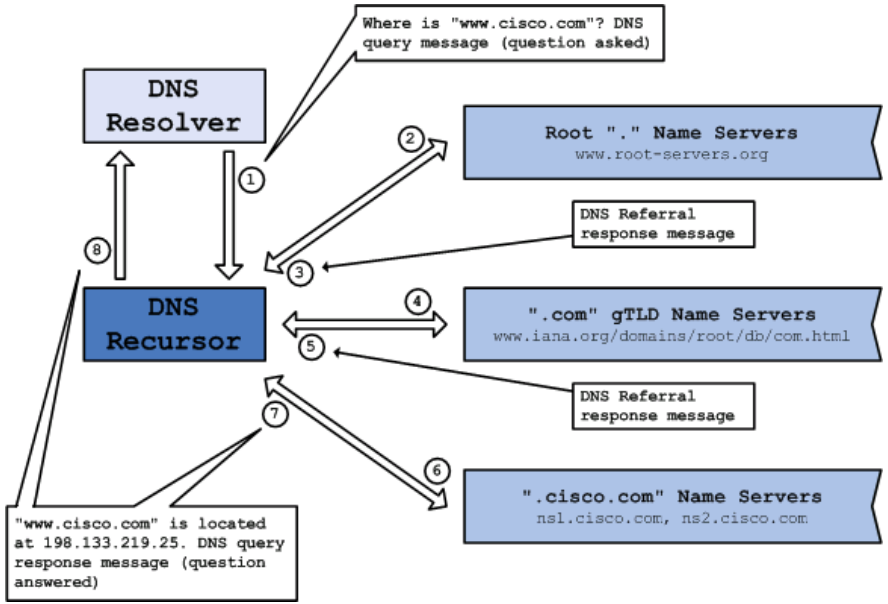
Une URL, c’est quoi ?
Comprendre la base du Web

- Une URL (Uniform Resource Locator) permet d’accéder à une ressource sur Internet (p.ex. https://www.orange.fr)

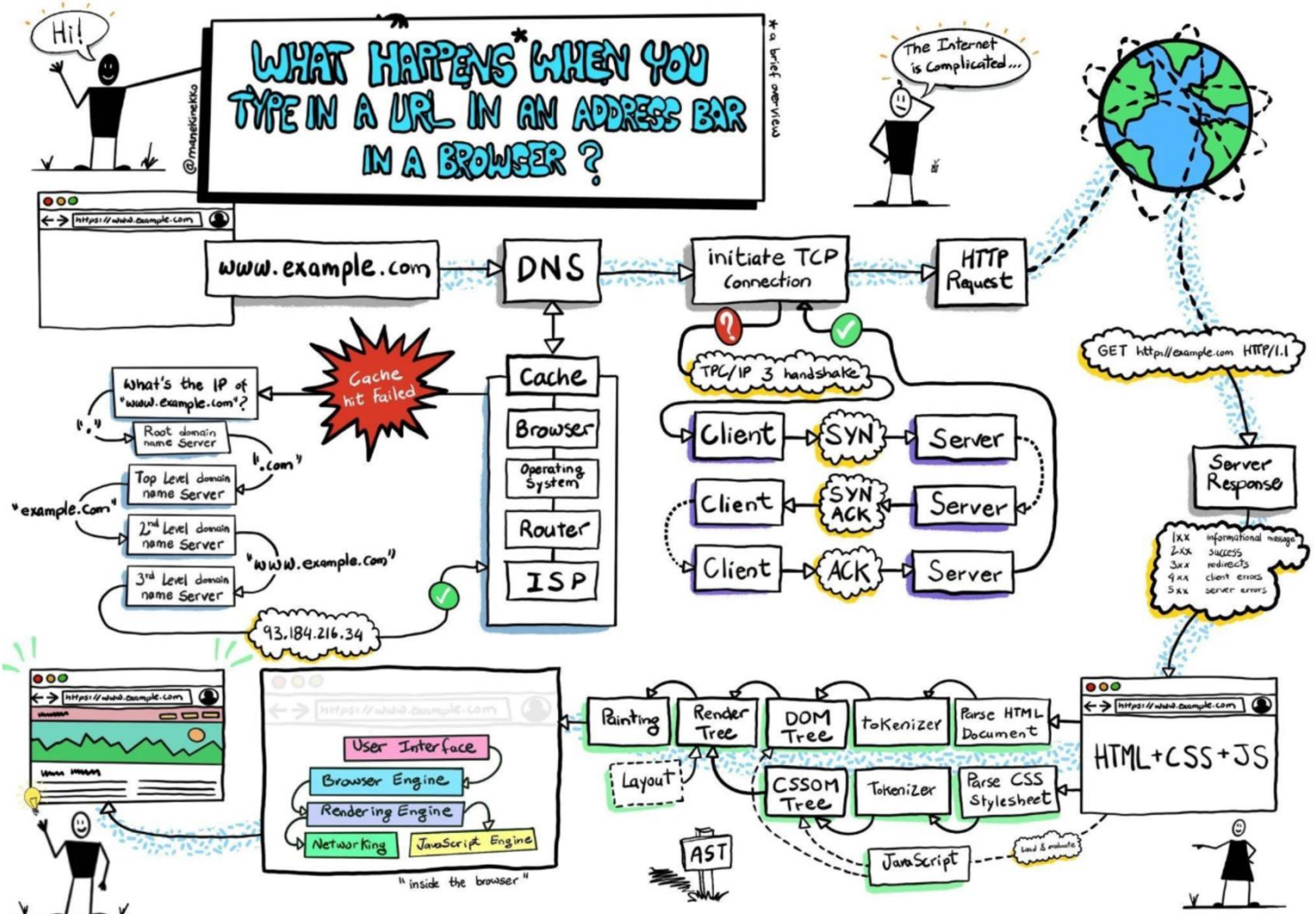
- Pour fonctionner, elle doit être traduite en adresse IP grâce au DNS (p.ex. 195.101.12.12)

Le DNS, le traducteur d'Internet
Le rôle clé du DNS

- Le DNS (Domain Name System) fait le lien entre le nom de domaine et l'adresse IP du serveur.
  - https://monsite-lithotherapie.com
  - https://www.orange.fr
  - https://www.cisco.com
  - ...
- Chaque fois qu'on saisit une URL, *n* requêtes DNS sont envoyées pour trouver l'adresse IP correspondante.
- Le DNS est donc un composant critique de la navigation Web.



… une histoire de récursion !



https://www.jpsymfony.com/culture-web/que-se-passe-t-il-quand-on-tape-une-url-dans-un-navigateur

Le DNS, le traducteur d'Internet
Des configurations diverses ...

Exemple : résolution d'une entrée DNS

2 étapes

```
www.orange.fr is an alias for www.orange.fr.multis.x-echo.com.
www.orange.fr.multis.x-echo.com is an alias for hpo-main.prod.hporange.gslb.fti.net.
hpo-main.prod.hporange.gslb.fti.net has address 193.252.117.207
hpo-main.prod.hporange.gslb.fti.net has IPv6 address 2a01:c9c0:b3:3000::71
```

3 étapes

```
www.yahoo.fr is an alias for rc.yahoo.com.
rc.yahoo.com is an alias for global-accelerator.dns-rc.aws.oath.cloud.
global-accelerator.dns-rc.aws.oath.cloud is an alias for a7de0457831fd11f7.awsglobalaccelerator.com.
a7de0457831fd11f7.awsglobalaccelerator.com has address 76.223.84.192
a7de0457831fd11f7.awsglobalaccelerator.com has address 13.248.158.7
```

0 étapes

```
www.google.fr has address 172.217.20.163
www.google.fr has IPv6 address 2a00:1450:4007:80c::2003
```

… des choix respectables, mais pouvant affecter la sécurité.
Il faut être capable de comprendre et gérer cette complexité ...

Pourquoi sécuriser le DNS ?

Une cible privilégiée pour les attaquants et des failles à fort impact

Domain hijacking

FQDN hors du domaine de gestion.

IP hijacking

A/AAAA hors du domaine de gestion.

Autres cas

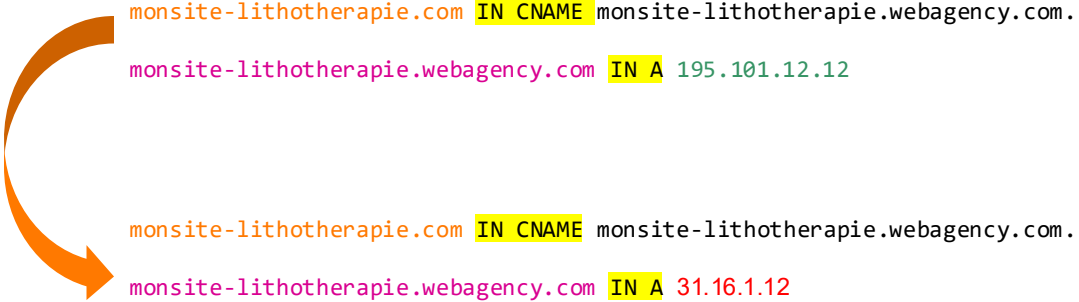
Mauvaises configurations,
problèmes de conformité.

Des failles courantes !

... vol de données,
détournement de trafic,
indisponibilité de services,
atteinte à l'image de marque,
lenteurs.

Cas n°1
Domain hijacking

- Vous avez votre site hébergé par la société webagency ; votre DNS est donc *pour exemple* ⇒
- La société webagency fait faillite
- Le hacker rachète son nom de domaine webagency.com
- Il redirige tout votre trafic vers son hébergement
- Il dépose un site à vos couleurs avec un formulaire de login/passwd
- Il vole les identifiants / CB de vos clients



Cas n°2
IP hijacking AWS

- Vous avez un service chez Amazon AWS
- Vous basculez vers un autre fournisseur
  - Mais vous lâchez le service Amazon de suite après la bascule *(ignorant la propagation DNS)*
- Le hacker achète en boucle des services Amazon jusqu'à tomber sur l'IP que vous venez de lâcher
  - Il reçoit votre trafic résiduel
  - Il met une page au nom de votre société avec un formulaire de login
  - Il reçoit des identifiants / passwd de vos clients durant un « certain temps »

Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates

Kevin Borgolte, Tobias Fiebig, Shuang Hao, Christopher Kruegel, Giovanni Vigna

Abstract—Infrastructure-as-a-Service (IaaS), and more generally the "cloud," like Amazon Web Services (AWS) or Microsoft Azure, have changed the landscape of system operations on the Internet. Their elasticity allows operators to rapidly allocate and use resources as needed, from virtual machines, to storage, to bandwidth, and even to IP addresses, which is what made them popular and spurred innovation.

In this paper, we show that the dynamic component paired with recent developments in trust-based ecosystems (e.g., SSL certificates) creates so far unknown attack vectors. Specifically, we discover a substantial number of stale DNS records that point to available IP addresses in clouds, yet, are still actively attempted to be accessed. Often, these records belong to discontinued services that were previously hosted in the cloud. We demonstrate that it is practical, and time and cost efficient for attackers to allocate IP addresses to which stale DNS records point. Considering the ubiquity of domain validation in trust ecosystems, like SSL certificates, an attacker can impersonate the service using a valid certificate trusted by all major operating systems and browsers. The attacker can then also exploit residual trust in the domain name for phishing, receiving and sending emails, or possibly distribute code to clients that load remote code from the domain (e.g., loading of native code by mobile apps, or JavaScript libraries by websites).

Even worse, an aggressive attacker could execute the attack in less than 70 seconds, well below common time-to-live (TTL) for DNS records. In turn, it means an attacker could exploit normal service migrations in the cloud to obtain a valid SSL certificate for domains owned and managed by others, and, worse, that she might not actually be bound by DNS records being (temporarily) stale, but that she can exploit caching instead.

We introduce a new authentication method for trust-based domain validation that mitigates stateless issues without incurring additional certificate requester effort by incorporating existing trust of a name into the validation process. Furthermore, we provide recommendations for domain name owners and cloud operators to reduce their and their clients' exposure to DNS statelessness issues and the resulting domain takeover attacks.

I. Introduction

Over the past ten years, cloud services have grown tremendously. Generally, clouds are comprised of hundreds to thousands of commodity servers, which make up pools of computing resources that are shared by different users. One of the main drivers behind the clouds' rise in popularity is their elasticity: users can acquire and use resources as needed, on demand, and at scale, all while requiring almost no upfront investment. In fact, Amazon Web Services (AWS), Amazon's public cloud, serves over one million active users worldwide [1]. Microsoft Azure is gaining 120,000 new customers each month [2], and the global cloud IP traffic has reached 3.9 zettabytes (3.9 billion terabytes) in 2015 already [3]. Unfortunately, as the recent years have shown, the resource pooling and increased popularity of cloud-based deployments also pose severe security issues to the clouds' tenants [4, 5].

With the clouds' increase in popularity and their commoditization, website operators have been empowered to deploy their website themselves instead of relying on more traditional web hosting. At the same time, HTTPS has become basically a requirement for any website operator, not only for dynamic websites trying to protect login credentials, but also for static websites. Unprotected websites are being ranked lower by search engines [6], they are limited in browser features that they can use [7], and they risk having content and advertisements injected, e.g., by wireless access point operators or Internet Service Providers [8, 9]. For HTTP/2, it has become practically mandatory because all major browsers support HTTP/2 over TLS only [10]. Website operators now typically deploy SSL certificates for their domains and use HTTPS to ensure integrity and confidentiality for any communication with their website. For certificates to be trusted by the websites' visitors' browsers, however, they need to be issued by trusted certificate authorities (CAs). Traditional verification approaches involve identity documents, like verifying passports, which incurred high processing overhead. To cope with the high-volume demand for digital certificates, CAs adopted automated approaches to verify and issue certificates, and now heavily rely on domain validation. Having launched only in April 2016, Let's Encrypt has since been dominating the domain-validation part of the certificate authority ecosystem through openly available and well-designed tooling that uses the Automatic Certificate Management Environment protocol (ACME) [11] to validate domain ownership and issue certificates almost transparently for users. Today, Let's Encrypt has issued over 100 million certificates in less than 15 months and their certificates account for 80% of all publicly trusted certificates [12, 13].

Unfortunately, combining the elasticity of cloud infrastructure and the automation of certificate issuance introduces new security vulnerabilities. In this paper, we discover that stale and abandoned DNS entries pointing to cloud IP addresses can be exploited by attackers to deceive domain-based certificate validation and obtain certificates for the victim domains. The problem stems from the ephemeral nature of the cloud resources. More specifically, if a user releases a cloud IP address, but does not remove the corresponding DNS entry before

Network and Distributed System Security (NDSS) Symposium 2018
18-21 February 2018, San Diego, CA, USA
ISBN 1-1891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23327
www.ndss-symposium.org

Kevin Borgolte, et al. "Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates." 25th Network and Distributed System Security Symposium (NDSS). Internet Society (ISOC), 2018. 10.14722/ndss.2018.23327

Cas n°3
NS entries

- Le géant Mastercard a laissé une entrée DNS à haut risque durant 5 ans

- Sauriez-vous la retrouver ?

```
└─# dig +tcp @dns1.mastercard.com az.mastercard.com

; <<>> DiG 9.19.17-2~kali1-Kali <<>> +tcp @dns1.mastercard.com az.mastercard.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45077
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 5, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
; COOKIE: 6d51066062f6102a13bff6c8678149d366a3aabb89779394 (good)
;; QUESTION SECTION:
;az.mastercard.com.             IN      A

;; AUTHORITY SECTION:
az.mastercard.com.      3600    IN      NS      a1-29.akam.net.
az.mastercard.com.      3600    IN      NS      a7-67.akam.net.
az.mastercard.com.      3600    IN      NS      a22-65.akam.ne.
az.mastercard.com.      3600    IN      NS      a26-66.akam.net.
az.mastercard.com.      3600    IN      NS      a9-64.akam.net.

;; Query time: 92 msec
;; SERVER: 216.119.218.53#53(dns1.mastercard.com) (TCP)
;; WHEN: Fri Jan 10 11:24:51 EST 2025
;; MSG SIZE  rcvd: 191
```

https://krebsonsecurity.com/2025/01/mastercard-dns-error-went-unnoticed-for-years/

Cas n°3
NS entries

- Le géant Mastercard a laissé une entrée DNS à haut risque durant 5 ans

- .ne => niger
- 20% du trafic était concerné

```
└─# dig +tcp @dns1.mastercard.com az.mastercard.com

; <<>> DiG 9.19.17-2~kali1-Kali <<>> +tcp @dns1.mastercard.com az.mastercard.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45077
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 5, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
; COOKIE: 6d51066062f6102a13bff6c8678149d366a3aabb89779394 (good)
;; QUESTION SECTION:
;az.mastercard.com.             IN      A

;; AUTHORITY SECTION:
az.mastercard.com.      3600    IN      NS      a1-29.akam.net.
az.mastercard.com.      3600    IN      NS      a7-67.akam.net.
az.mastercard.com.      3600    IN      NS      a22-65.akam.ne.
az.mastercard.com.      3600    IN      NS      a26-66.akam.net.
az.mastercard.com.      3600    IN      NS      a9-64.akam.net.

;; Query time: 92 msec
;; SERVER: 216.119.218.53#53(dns1.mastercard.com) (TCP)
;; WHEN: Fri Jan 10 11:24:51 EST 2025
;; MSG SIZE  rcvd: 191
```

https://krebsonsecurity.com/2025/01/mastercard-dns-error-went-unnoticed-for-years/

Sécuriser le DNS

Vers une analyse intelligente et automatisée

Comment identifier rapidement et efficacement les failles DNS dans un parc hétérogène et volumineux ?

Quelques verrous scientifiques et techniques à lever ...

- Outils traditionnels (scripts, audits manuels) : effort de développement et de maintenance non négligeable, portabilité limitée, logiques métier variables et non partagées.
- Sources de données : diverses zones DNS (conforme spécs. RFC) à combiner et à enrichir avec des informations services et métier (p.ex. subnet, gestionnaire).
- Efficacité opérationnelle : gérer des interdépendances parfois longues et à fort impact entre éléments de configuration.

Sécuriser le DNS

Vers une analyse intelligente et automatisée, quelques intuitions ...

Le DNS fonctionne sous forme de chaîne de résolution

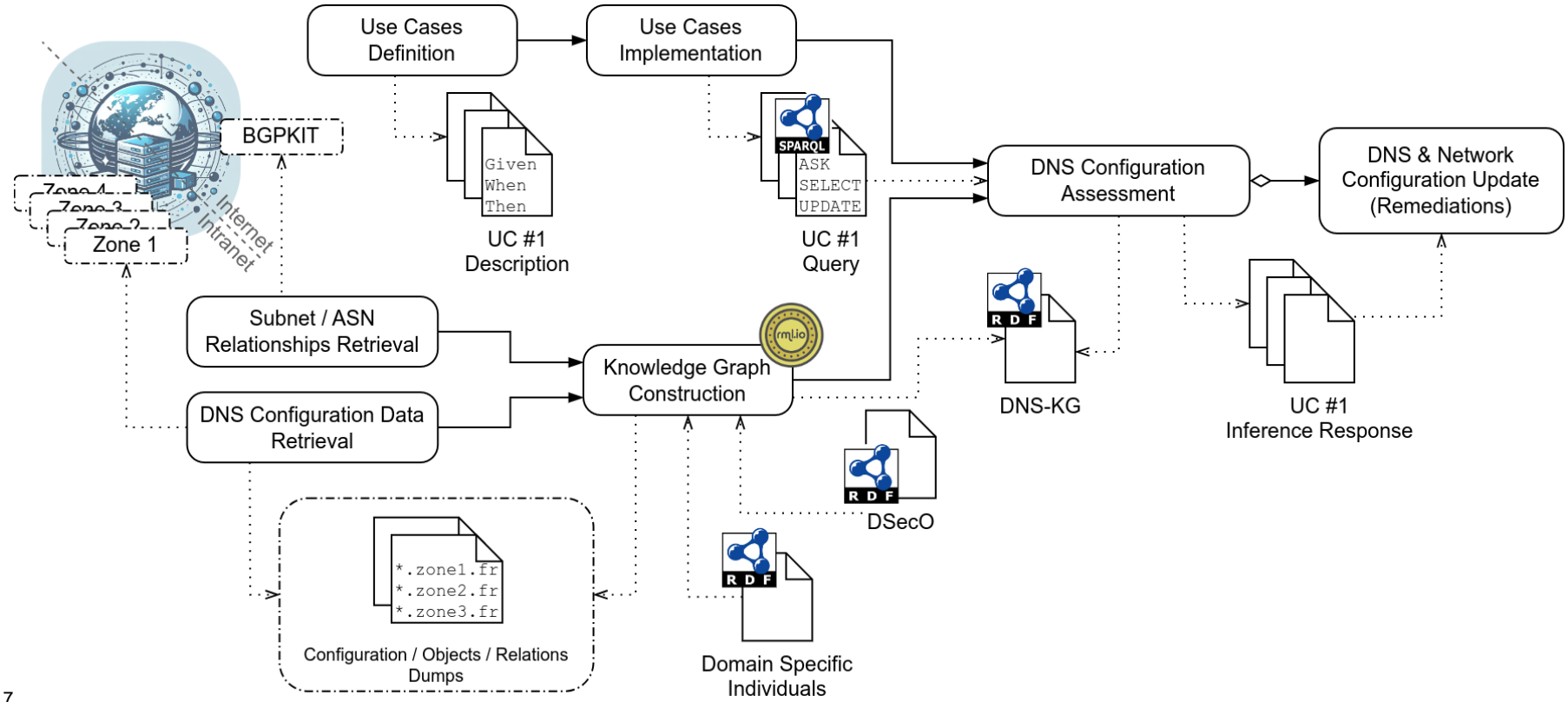
Les graphes sont une approche naturelle pour modéliser ce fonctionnement...

Un vocabulaire partagé pour structurer des données hétérogènes

Une représentation explicite permettrait d'analyser les configurations DNS selon un prisme commun et logique.

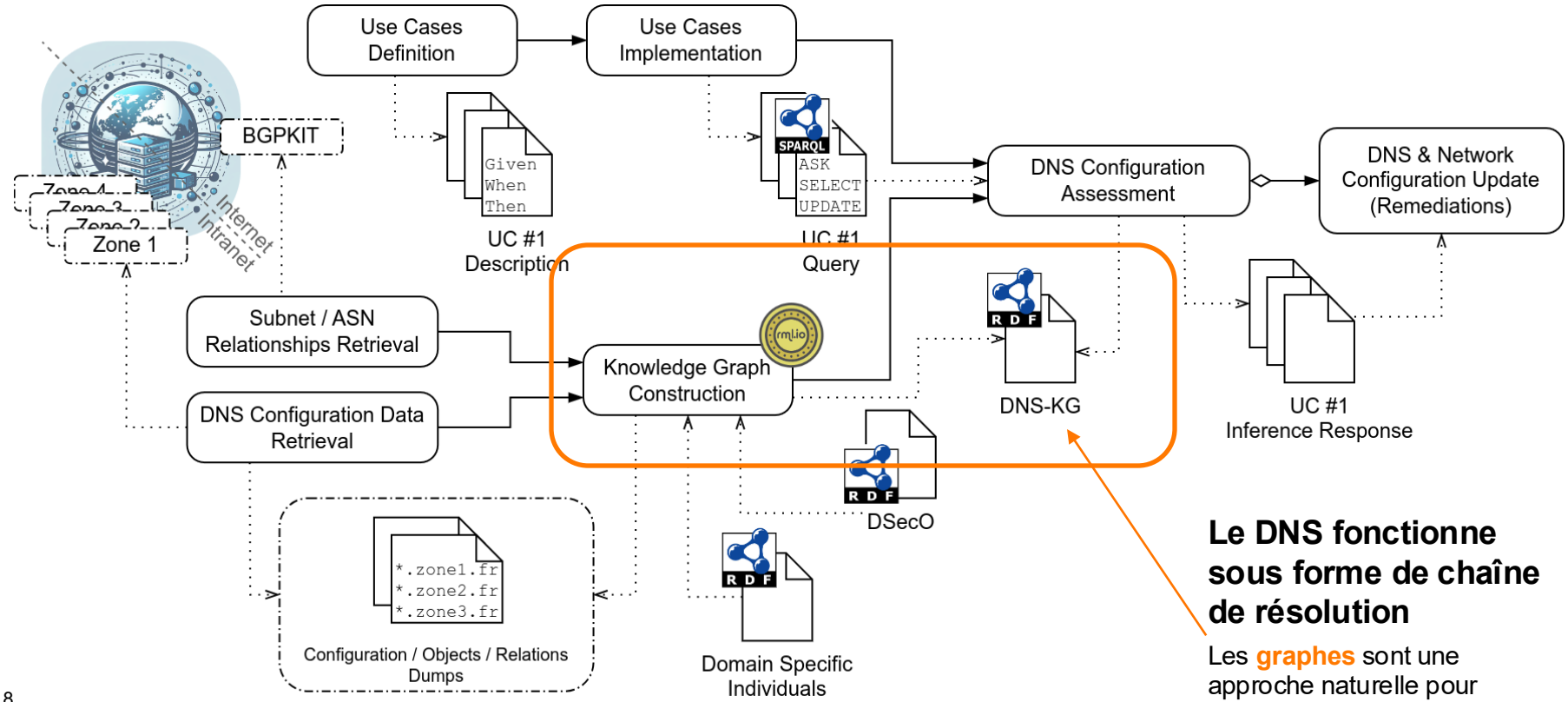
Sécuriser le DNS

L'approche DNS-KG / DSecO



Sécuriser le DNS

L'approche DNS-KG / DSecO



Le DNS fonctionne sous forme de chaîne de résolution

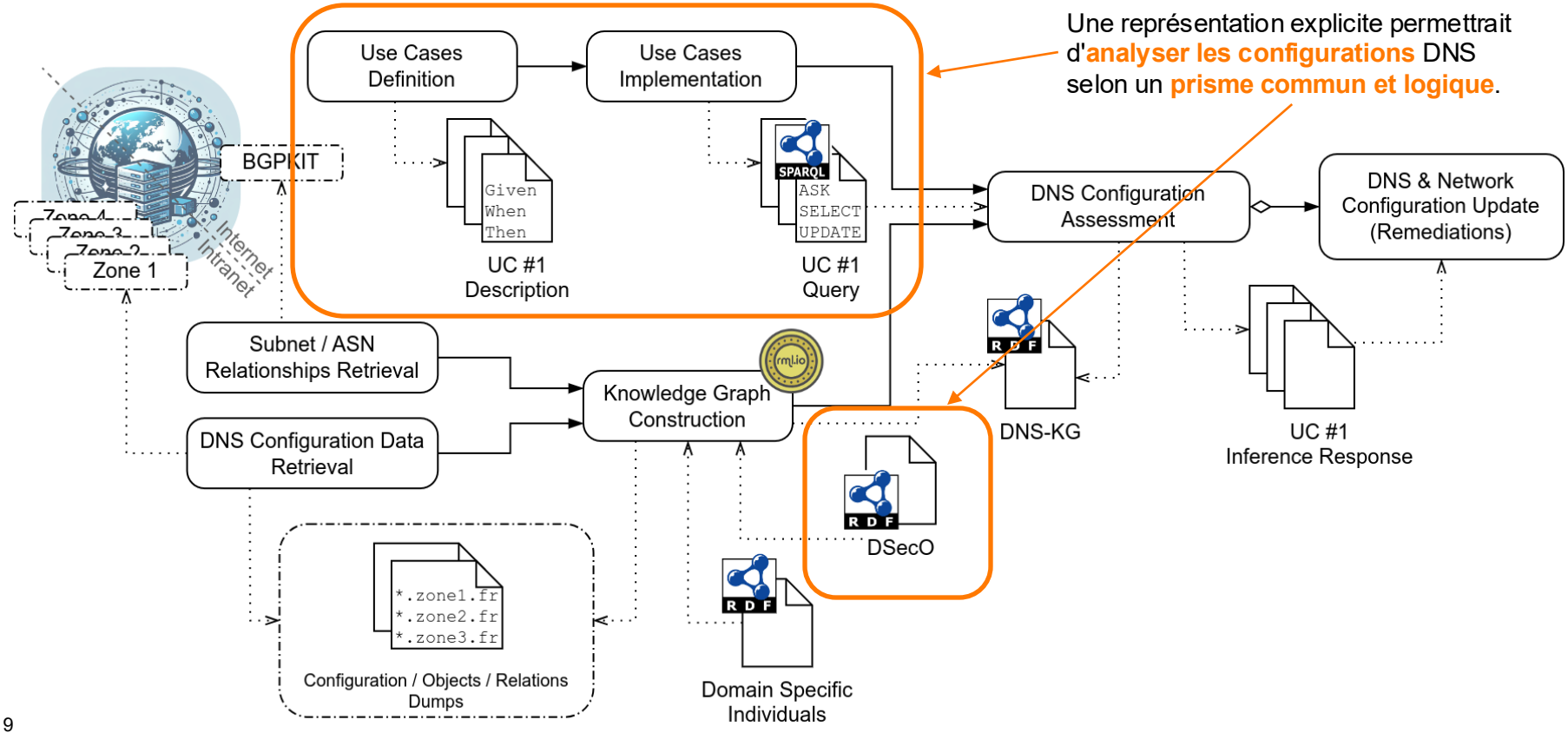
Les **graphes** sont une approche naturelle pour modéliser ce fonctionnement...

Sécuriser le DNS

L'approche DNS-KG / DSecO

Un vocabulaire partagé pour structurer des données hétérogènes

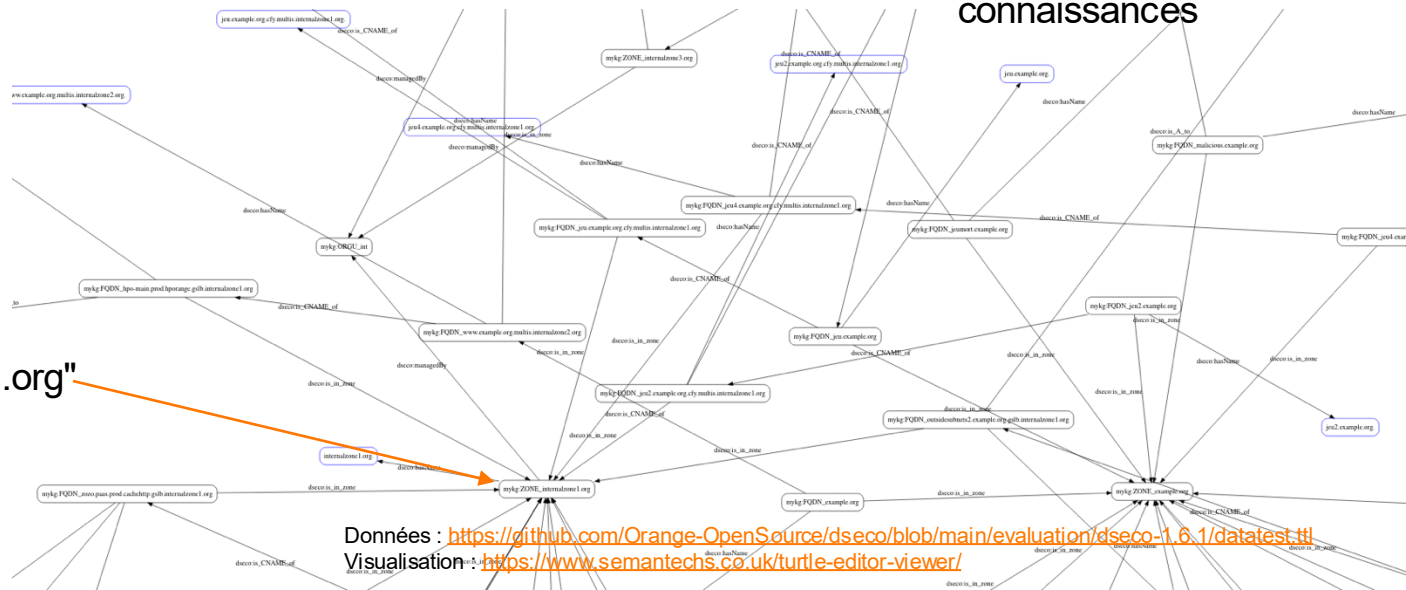
Une représentation explicite permettrait d'**analyser les configurations** DNS selon un **prisme commun et logique**.



Graphes de connaissances ? Représentation explicite ?
Mais de quoi parle-t-on ?

Les graphes de connaissances permettent d'utiliser des techniques d'analyse de données et d'inférence pour raisonner sur le contexte des objets représentés tout en gérant des données hétérogènes.

Des données de configuration DNS sous la forme d'un graphe de connaissances



Données : https://github.com/Orange-OpenSource/dseco/blob/main/evaluation/dseco-1.6.1/datatest.ttl
Visualisation : https://www.semantechs.co.uk/turtle-editor-viewer/

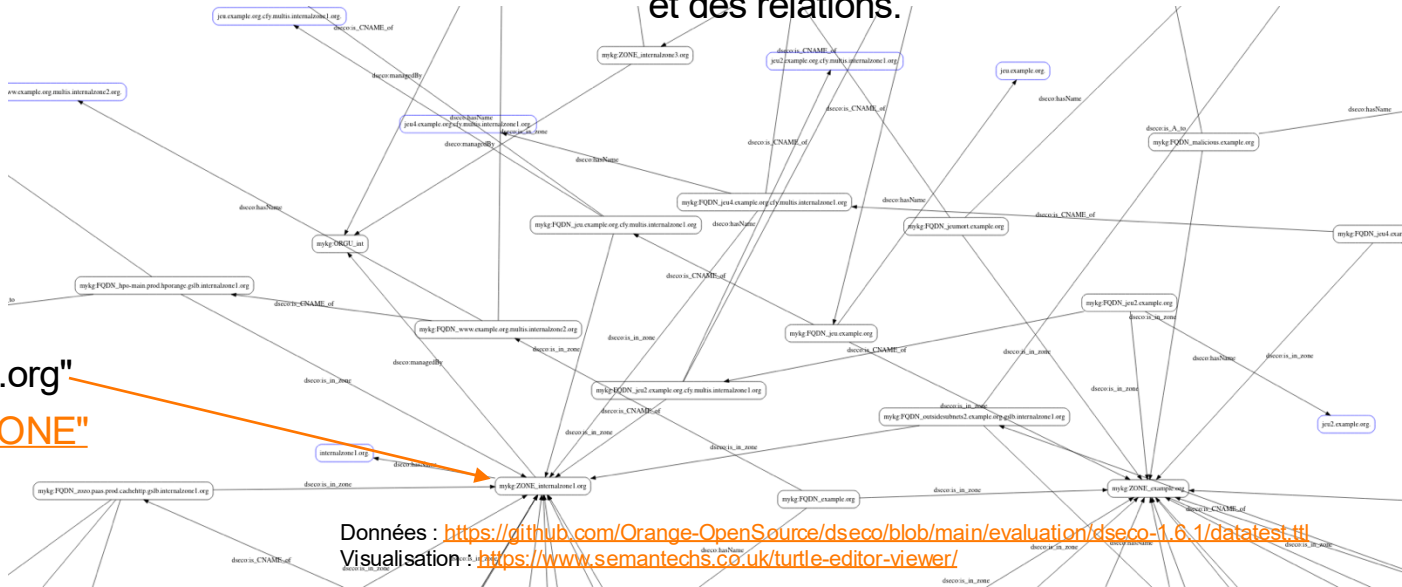
Graphes de connaissances ? Représentation explicite ?
Mais de quoi parle-t-on ?

Les graphes de connaissances permettent d'utiliser des techniques d'analyse de données et d'inférence pour raisonner sur le contexte des objets représentés tout en gérant des données hétérogènes.

Ontologies : représentations explicites d'un domaine de discours à travers des concepts et des relations.

Un nœud "internalzone1.org"

est une entité de type "ZONE"



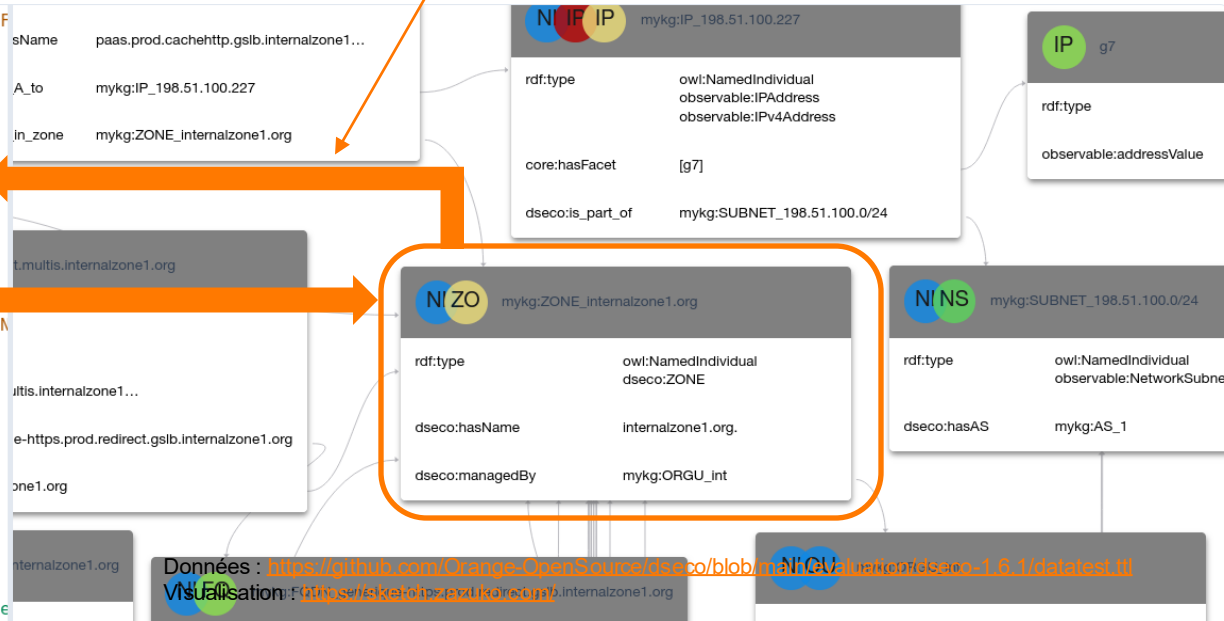
Données : https://github.com/Orange-OpenSource/dseco/blob/main/evaluation/dseco-1.6.1/datatest.ttl
Visualisation : https://www.semantechs.co.uk/turtle-editor-viewer/

Graphes de connaissances ? Représentation explicite ?
Mais de quoi parle-t-on ?

Les graphes de connaissances permettent d'utiliser des techniques d'analyse de données et d'inférence pour raisonner sur le contexte des objets représentés tout en gérant des données hétérogènes.

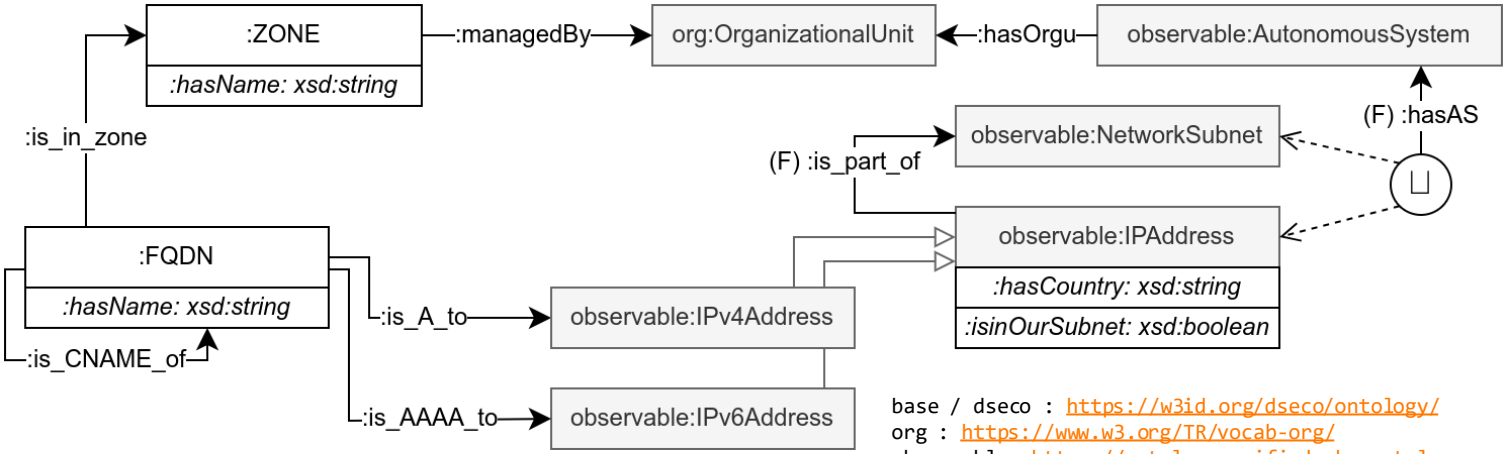
De la représentation graphique (conviviale pour l'humain) à la représentation en triplets sujet-prédicat-objet (adaptée aux ordinateurs), et vice versa.

```
mykg:FQDN_christoph2.internalzone2.org a owl:NamedIndividual, dseco:F
    dseco:is_in_zone mykg:ZONE_internalzone2.org ;
    dseco:hasName "christoph2.internalzone2.org." .

mykg:ZONE_yetanother_isp.fr a owl:NamedIndividual, dseco:ZONE ;
    dseco:hasName "yetanother_isp.fr." .

mykg:ZONE_internalzone1.org a owl:NamedIndividual, dseco:ZONE ;
    dseco:hasName "internalzone1.org." ;
    dseco:managedBy mykg:ORGU_int .

mykg:ZONE_example.org a owl:NamedIndividual, d    co:ZONE ;
    dseco:hasName "example.org." ;
    dseco:managedBy mykg:ORGU_top .

mykg:ZONE_webagencyintermediaire.com a owl:NamedIndividual, dseco:ZON
    dseco:hasName "webagencyintermediaire.com." .

mykg:ZONE_webagencymorte.fr a owl:NamedIndividual, dseco:ZONE ;
    dseco:hasName "webagencymorte.fr." .

mykg:ZONE_internalzone2.org a owl:NamedIndividual, dseco:ZONE ;
    dseco:hasName "internalzone2.org." ;
    dseco:managedBy mykg:ORGU_int .

mykg:ZONE_internalzone3.org a owl:NamedIndividual, dseco:ZONE ;
    dseco:hasName "internalzone3.org." ;
    dseco:managedBy mykg:ORGU_int .

mykg:IP_12.12.12.12 a owl:NamedIndividual, observable:IPAddress, obse
```



DSecO
Une ontologie pour la sécurité DNS

- Une ontologie légère en syntaxe RDFS/OWL
  - Namespace `dseco`
  - Deux classes, huit object properties, quatre data type properties
  - Réutilise les vocabulaires ORG et UCO
  - Expressivité ALURF(D)
- Dérivée de la RFC 1035 et de neuf cas de détection formulés par des experts SecOps Orange
- Permet la représentation des éléments de configuration DNS
- Disponible en open source : https://w3id.org/dseco/ (BSD-4)



base / dseco : https://w3id.org/dseco/ontology/
org : https://www.w3.org/TR/vocab-org/
observable: https://ontology.unifiedcyberontology.org/uco/observable/

De la faille à la requête
Exemples de détection automatisée

Domain hijacking

FQDN hors gestion Orange : détection de domaines à surveiller de près.

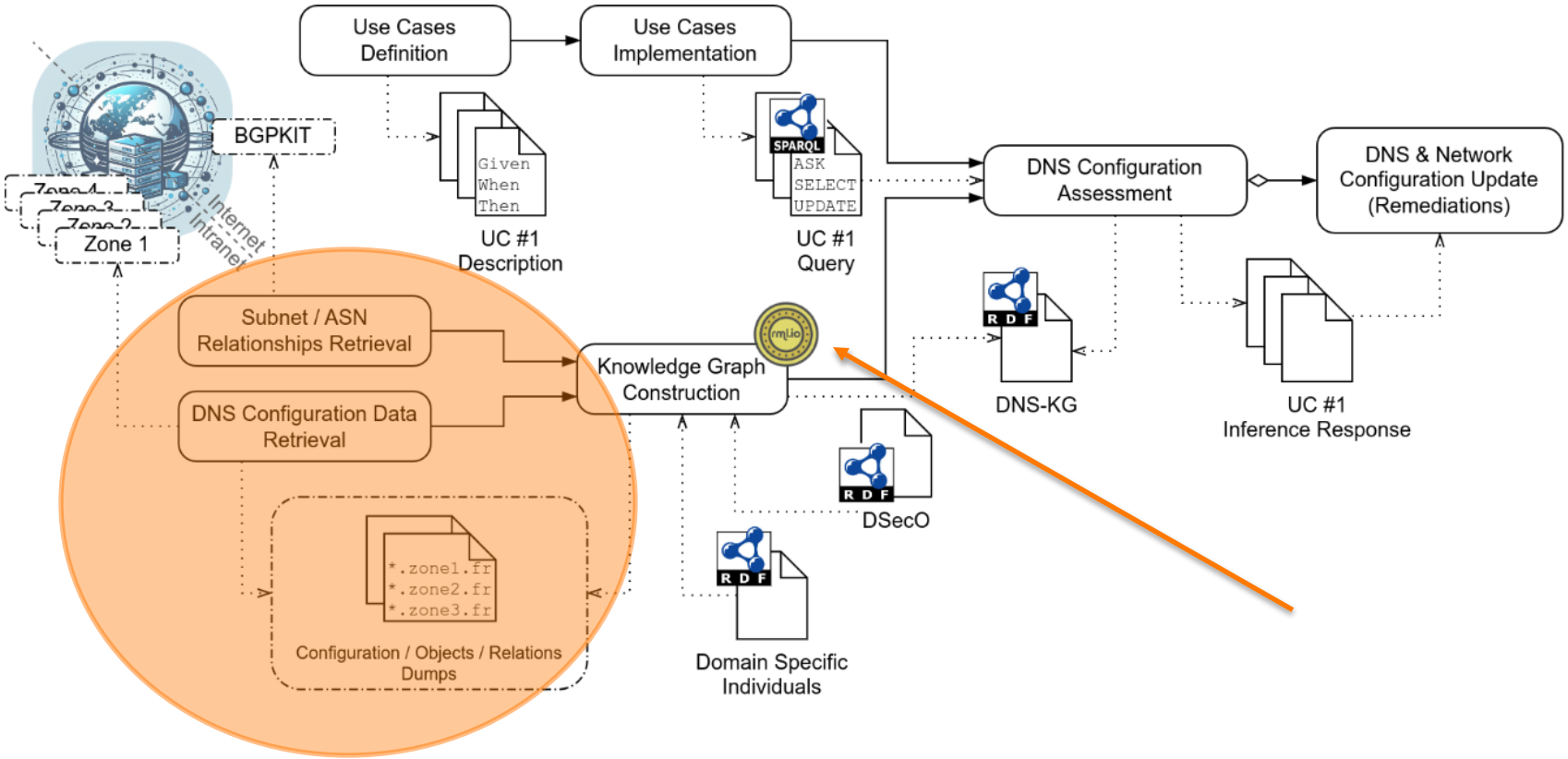
IP hijacking

A/AAAA hors gestion Orange : détection des IPs à surveiller de près.

Autres cas

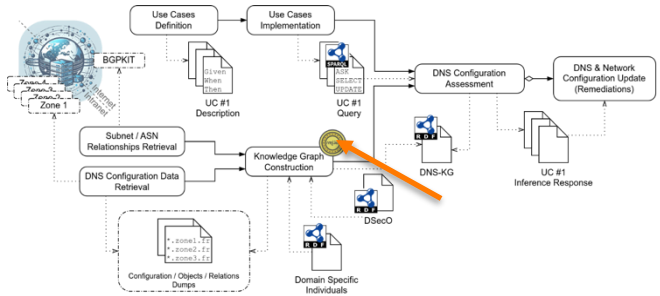
Mauvaises configurations, problèmes de conformité

Knowledge Graph Construction
Principes



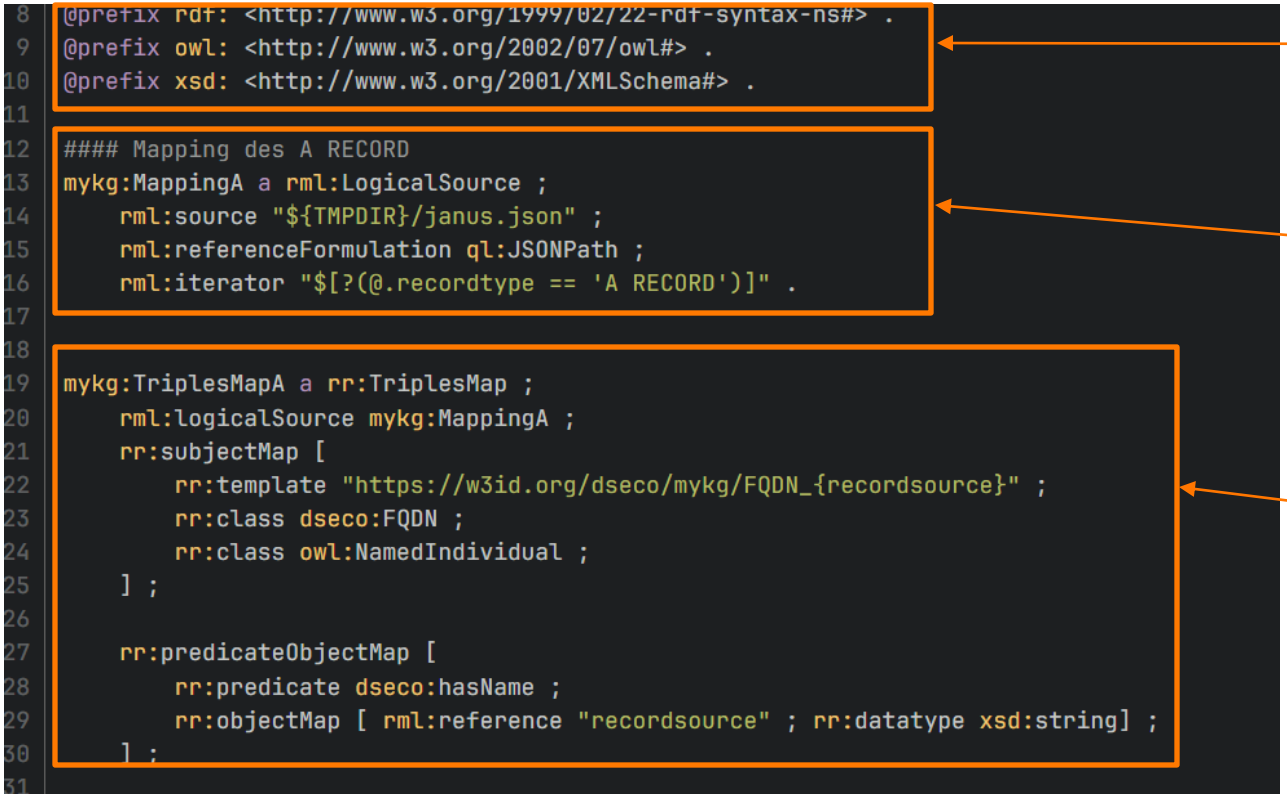
Knowledge Graph Construction
Principes

- Initialement: retrievers en Python qui faisaient chacun un petit boulot
- Problèmes: processus long (à écrire et à exécuter), peu modulaire, intégration de nouvelles sources compliquée
- Passage en RML (RDF Mapping Language)
- Avantages: relativement facile à écrire, modularité, intégration de nouvelles données facilitée



Spécifications RML : RML.io

Exemple de mapping RML



Partie préfixes

Partie source mapping

Partie TriplesMap

Exemple de mapping RML

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

#### Mapping des A RECORD
mykg:MappingA a rml:LogicalSource ;
    rml:source "${TMPDIR}/janus.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$[?(@.recordtype == 'A RECORD')]" .

mykg:TriplesMapA a rr:TriplesMap ;
    rml:logicalSource mykg:MappingA ;
    rr:subjectMap [
        rr:template "https://w3id.org/dseco/mykg/FQDN_{recordsource}" ;
        rr:class dseco:FQDN ;
        rr:class owl:NamedIndividual ;
    ] ;

    rr:predicateObjectMap [
        rr:predicate dseco:hasName ;
        rr:objectMap [ rml:reference "recordsource" ; rr:datatype xsd:string] ;
    ] ;
```

Attention à bien mettre le même nom pour le mapping

Exemple de mapping RML

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

#### Mapping des A RECORD
mykg:MappingA a rml:LogicalSource ;
    rml:source "${TMPDIR}/janus.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$[?(@.recordtype == 'A RECORD')]" ;

mykg:TriplesMapA a rr:TriplesMap ;
    rml:logicalSource mykg:MappingA ;
    rr:subjectMap [
        rr:template "https://w3id.org/dseco/mykg/FQDN_{recordsource}" ;
        rr:class dseco:FQDN ;
        rr:class owl:NamedIndividual ;
    ] ;

    rr:predicateObjectMap [
        rr:predicate dseco:hasName ;
        rr:objectMap [ rml:reference "recordsource" ; rr:datatype xsd:string] ;
    ] ;
```

L'itérateur permet de

boucler sur les

enregistrements

remplissant la/les

condition(s)

Exemple de mapping RML

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

#### Mapping des A RECORD
mykg:MappingA a rml:LogicalSource ;
    rml:source "${TMPDIR}/janus.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$[?(@.recordtype == 'A RECORD')]" .

mykg:TriplesMapA a rr:TriplesMap ;
    rml:logicalSource mykg:MappingA ;
    rr:subjectMap [
        rr:template "https://w3id.org/dseco/mykg/FQDN_{recordsource}" ;
        rr:class dseco:FQDN ;
        rr:class owl:NamedIndividual ;
    ] ;

    rr:predicateObjectMap [
        rr:predicate dseco:hasName ;
        rr:objectMap [ rml:reference "recordsource" ; rr:datatype xsd:string] ;
    ] ;
```

- Attention au nom des objets et leur ordre
- Chaque terme dans le fichier de mapping à son importance

Exemple de mapping RML

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

#### Mapping des A RECORD
mykg:MappingA a rml:LogicalSource ;
    rml:source "${TMPDIR}/janus.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$[?(@.recordtype == 'A RECORD')]" .

mykg:TriplesMapA a rr:TriplesMap ;
    rml:logicalSource mykg:MappingA ;
    rr:subjectMap [
        rr:template "https://w3id.org/dseco/mykg/FQDN_{recordsource}" ;
        rr:class dseco:FQDN ;
        rr:class owl:NamedIndividual ;
    ] ;

    rr:predicateObjectMap [
        rr:predicate dseco:hasName ;
        rr:objectMap [ rml:reference "recordsource" ; rr:datatype xsd:string] ;
    ] ;
```

Création d'objets issus de la terminologie de dseco

Knowledge Graph Construction
Validation du build

- On obtient un graphe de connaissances avec nos données, stocké sous forme de fichier texte manipulable par n'importe quel utilisateur
- Validation du graphe avant interrogation : on effectue une validation syntaxique et logique à l'aide d'un raisonneur automatique (Pellet)
- Exemple : un individu ne peut pas être à la fois un FQDN et à la fois une IP ...

```
Explanation(s):
1)   FQDN_auto.example.org type IPAddress
     FQDN_auto.example.org type FQDN
     DisjointClasses(OrganizationalUnit
                     AutonomousSystem
                     IPAddress
                     NetworkSubnet
                     COMPONENT
                     CVE
                     FQDN
                     ZONE)
```

Knowledge Graph Construction
Enrichissement du graphe

- On obtient une ontologie valide et cohérente
- Rajout d'une étape d'enrichissement (cette partie pourrait être optionnelle)
  - On insère de nouvelles informations (aka. knowledge)
  - On utilise plusieurs requêtes SPARQL INSERT

```
PREFIX mykg: <https://w3id.org/dseco/mykg/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

INSERT {
    mykg:FQDN_resolving a owl:Class .
}
WHERE {
    FILTER NOT EXISTS { mykg:FQDN_resolving a owl:Class }
}
```

```
PREFIX dseco: <https://w3id.org/dseco/ontology/>
PREFIX mykg: <https://w3id.org/dseco/mykg/>

INSERT { ?Fqdn_orig a mykg:FQDN_resolving . }
WHERE {
    ?Fqdn_orig a dseco:FQDN .
    ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_end .
    ?Fqdn_end dseco:is_A_to ?_IP .
}
```

Knowledge Graph Construction
Enrichissement du graphe

```
PREFIX mykg: <https://w3id.org/dseco/mykg/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

INSERT {
    mykg:FQDN_resolving a owl:Class .
}
WHERE {
    FILTER NOT EXISTS { mykg:FQDN_resolving a owl:Class }
}
```

But: rajouter la classe FQDN_resolving

```
PREFIX dseco: <https://w3id.org/dseco/ontology/>
PREFIX mykg: <https://w3id.org/dseco/mykg/>

INSERT { ?Fqdn_orig a mykg:FQDN_resolving . }
WHERE {
    ?Fqdn_orig a dseco:FQDN .
    ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_end .
    ?Fqdn_end dseco:is_A_to ?_IP .
}
```

But: rajouter des données métier

Knowledge Graph Construction
Expérimentations & résultats

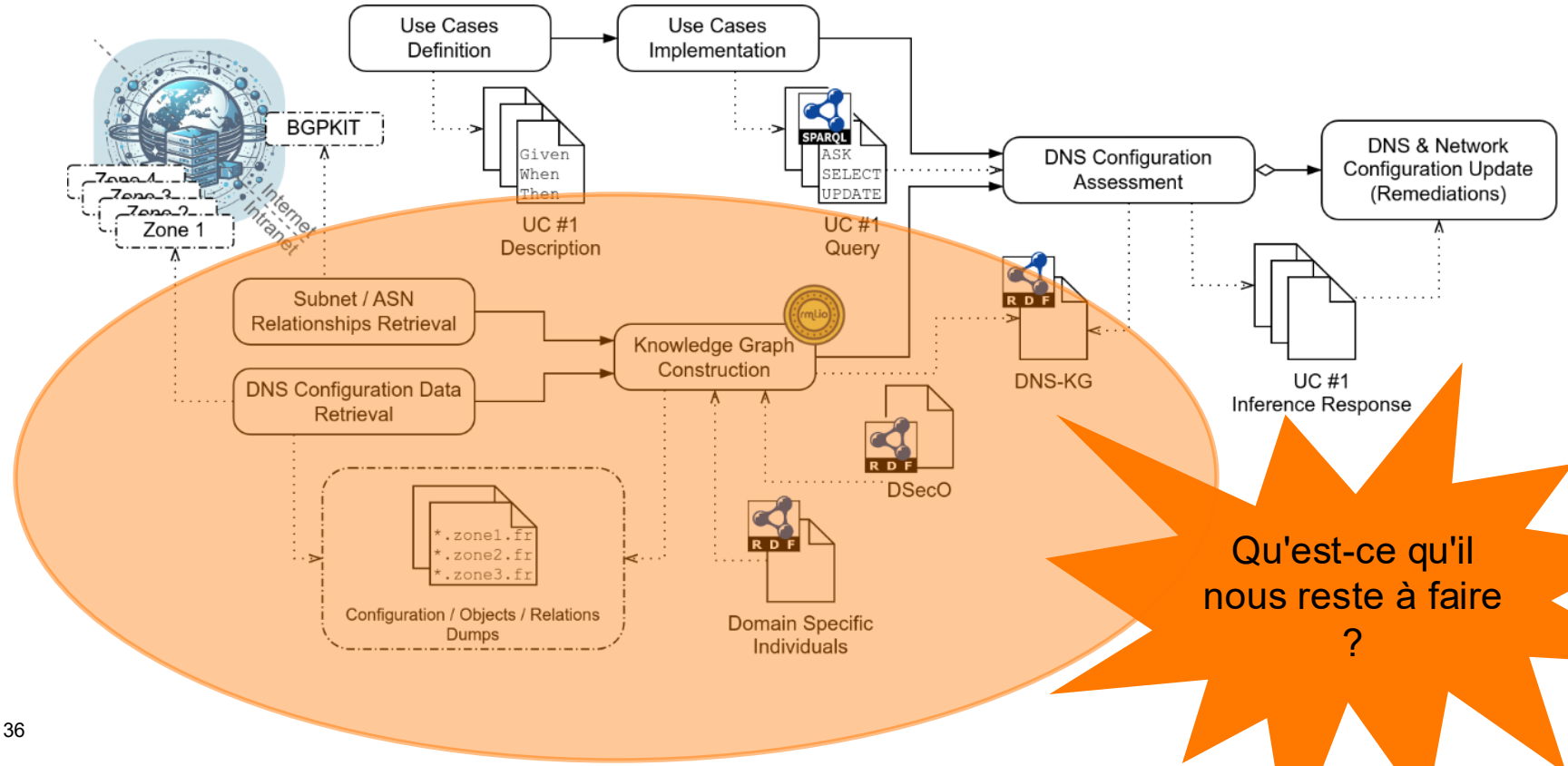
- On a généré une ontologie finale de 12 MiB de texte brut issus de nos fichiers de zone DNS, de nos SBOMs et de fichiers issus du Web
- Temps de génération : 10 minutes (séquentiel)
- Environ 160 000 individus créés
- Périmètre Orange Groupe

TABLE I
DNS-KG DATASET STATISTICS.

	Toy example	Real-world
Class	Count	Count
dseco:FQDN	43	22'404
dseco:ZONE	7	297
observable:IPAddress	12	7'817
observable:IPAddressFacet	12	7'817
observable:IPv4Address	11	7'079
observable:IPv6Address	1	738
observable:NetworkSubnet	5	13
observable:AutonomousSystem	2	162
observable:AutonomousSystemFacet	2	162
mykg:FQDN_resolving	17	7'803
mykg:FQDN_orange_resolving	0	4'036
Property	Count	Count
dseco:hasName	50	21'905
dseco:hasAS	5	6'339
dseco:hasOrgu	2	162
dseco:is_A_to	12	12'971
dseco:is_AAAA_to	0	873
dseco:is_CNAME_of	27	9'838
dseco:is_in_zone	41	21'608
dseco:is_part_of	11	906
dseco:managedBy	4	297
uco:asHandle	2	162
uco:addressValue	12	7'817
dseco:isinOurSubnet	6	906
dseco:is_CNAME_of_CNAME_of	15	5'130
mykg:isUsingOurCDN	0	42

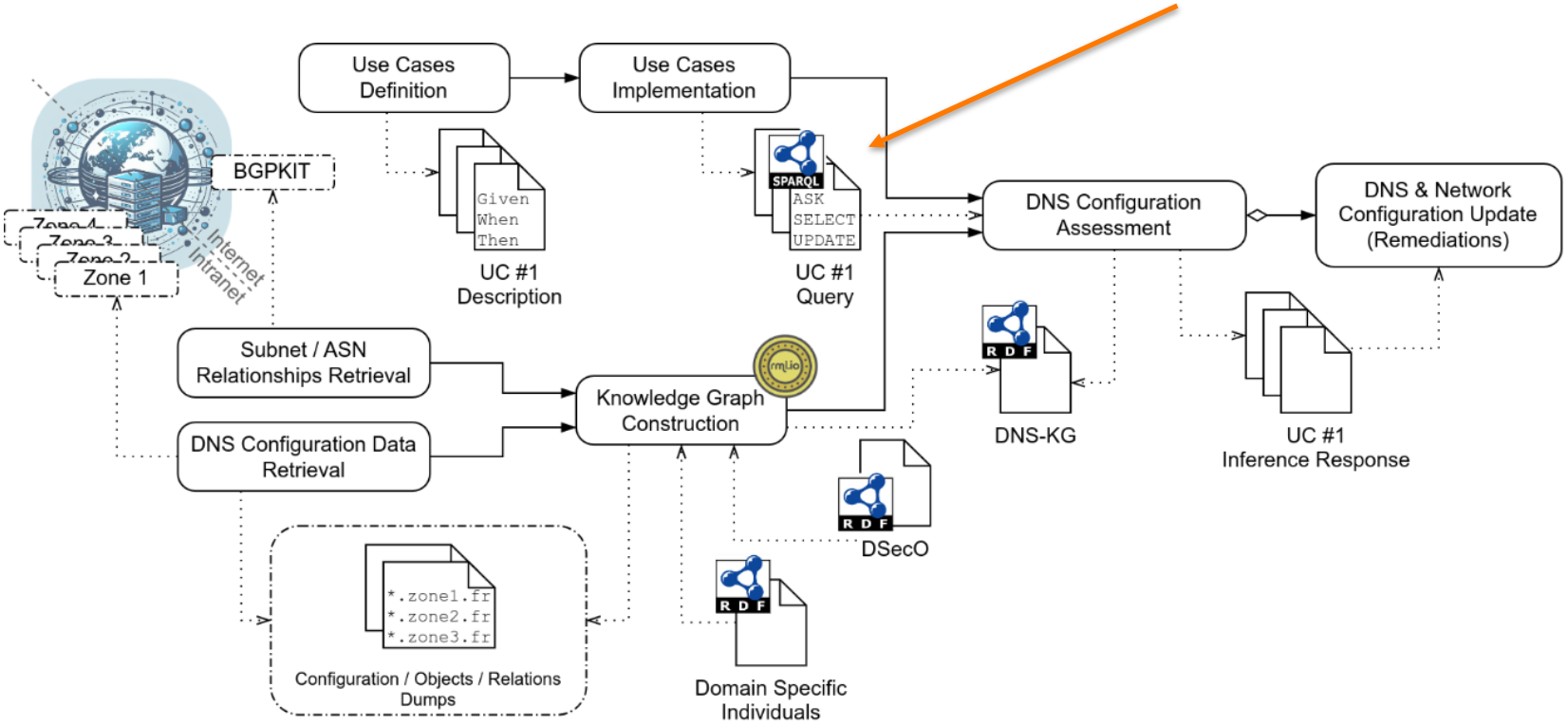
De la faille à la requête

Implémentation des cas de détection



Requêtes

Exploiter le graphe (analyse des indices clés)



De la faille à la requête

Implémentation des cas de détection

```
SELECT ?nameOrig ?name_intermediate_or_end
WHERE {
      ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_intermediate_or_end .
      # attention here, difference with others, we do not write that fqdn_intermediate_or_end is a "end fq

      ?Fqdn_orig dseco:hasName ?nameOrig .
      ?Fqdn_orig dseco:is_in_zone ?Zorig .
      ?Zorig dseco:hasName "example.org."^^xsd:string .

      ?Fqdn_intermediate_or_end dseco:hasName ?name_intermediate_or_end .
      ?Fqdn_intermediate_or_end dseco:is_in_zone ?zone_intermediate_or_end .

      # Exclude if managed by ORGU_top or ORGU_int
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_top .
      }
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_int .
      }
}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will d
ORDER BY ?nameOrig ?name_intermediate_or_end
```

```
SELECT ?nameOrig ?IP_End
WHERE {
      ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_end .
      FILTER NOT EXISTS {
        ?Fqdn_end dseco:is_CNAME_of ?_OtherFQDN
      }

      ?Fqdn_orig dseco:hasName ?nameOrig .
      ?Fqdn_orig dseco:is_in_zone ?Zorig .
      ?Zorig dseco:hasName "example.org."^^xsd:string .

      ?Fqdn_end dseco:hasName ?nameEnd .
      ?Fqdn_end dseco:is_A_to ?IP_End .

      # The interesting part of the query
      FILTER NOT EXISTS {
          ?IP_End dseco:isinOurSubnet "true"^^xsd:boolean .
      }


}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will do the same.
ORDER BY ?nameOrig ?nameEnd
```

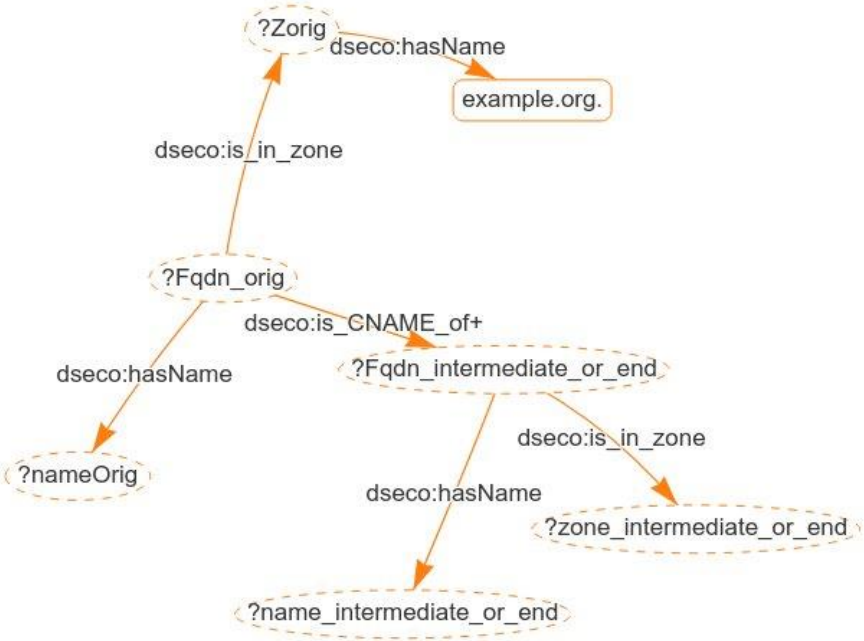
Requête

Analyse des indices clés + cas domain hijacking

Domain hijacking

FQDN hors gestion Orange : détection de domaines à surveiller de près.

- Format: SPARQL
- Langage: triplets en langage naturel
- Langage proche du SQL
- Temps de requêtage: quelques secondes pour obtenir les résultats



Requête domain hijacking

```
SELECT ?nameOrig ?name_intermediate_or_end
WHERE {
      ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_intermediate_or_end .
      # attention here, difference with others, we do not write that fqdn_intermediate_or_end is a "end fqdn"

      ?Fqdn_orig dseco:hasName ?nameOrig .
      ?Fqdn_orig dseco:is_in_zone ?Zorig .
      ?Zorig dseco:hasName "example.org."^^xsd:string .

      ?Fqdn_intermediate_or_end dseco:hasName ?name_intermediate_or_end .
      ?Fqdn_intermediate_or_end dseco:is_in_zone ?zone_intermediate_or_end .

      # Exclude if managed by ORGU_top or ORGU_int
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_top .
      }
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_int .
      }
}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will do the same.
ORDER BY ?nameOrig ?name_intermediate_or_end
```

Requête domain hijacking

```
SELECT ?nameOrig ?name_intermediate_or_end
WHERE {
    ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_intermediate_or_end .
    # attention here, difference with others, we do not write that fqdn_intermediate_or_end is a "end fqdn"

    ?Fqdn_orig dseco:hasName ?nameOrig .
    ?Fqdn_orig dseco:is_in_zone ?Zorig .
    ?Zorig dseco:hasName "example.org."^^xsd:string .

    ?Fqdn_intermediate_or_end dseco:hasName ?name_intermediate_or_end .
    ?Fqdn_intermediate_or_end dseco:is_in_zone ?zone_intermediate_or_end .

    # Exclude if managed by ORGU_top or ORGU_int
    FILTER NOT EXISTS {
        ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_top .
    }
    FILTER NOT EXISTS {
        ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_int .
    }
}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will do the same.
ORDER BY ?nameOrig ?name_intermediate_or_end
```

Requête domain hijacking

```
SELECT ?nameOrig ?name_intermediate_or_end
WHERE {
      ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_intermediate_or_end .
      # attention here, difference with others, we do not write that fqdn_intermediate_or_end is a "end fqdn"

      ?Fqdn_orig dseco:hasName ?nameOrig .
      ?Fqdn_orig dseco:is_in_zone ?Zorig .
      ?Zorig dseco:hasName "example.org."^^xsd:string .

      ?Fqdn_intermediate_or_end dseco:hasName ?name_intermediate_or_end .
      ?Fqdn_intermediate_or_end dseco:is_in_zone ?zone_intermediate_or_end .

      # Exclude if managed by ORGU_top or ORGU_int
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_top .
      }
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_int .
      }
}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will do the same.
ORDER BY ?nameOrig ?name_intermediate_or_end
```

Requête domain hijacking

```
SELECT ?nameOrig ?name_intermediate_or_end
WHERE {
      ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_intermediate_or_end .
      # attention here, difference with others, we do not write that fqdn_intermediate_or_end is a "end fqdn"

      ?Fqdn_orig dseco:hasName ?nameOrig .
      ?Fqdn_orig dseco:is_in_zone ?Zorig .
      ?Zorig dseco:hasName "example.org."^^xsd:string .

      ?Fqdn_intermediate_or_end dseco:hasName ?name_intermediate_or_end .
      ?Fqdn_intermediate_or_end dseco:is_in_zone ?zone_intermediate_or_end .

      # Exclude if managed by ORGU_top or ORGU_int
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_top .
      }
      FILTER NOT EXISTS {
          ?zone_intermediate_or_end dseco:managedBy mykg:ORGU_int .
      }
}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will do the same.
ORDER BY ?nameOrig ?name_intermediate_or_end
```

Requête domain hijacking

Des couples *nameOrig* ↦ *name_intermediate_or_end* à risque sont identifiés

```
[onto] acawet ~/ontologie/dseco/ontology/dseco-latest (try_to_optimize_the_code)$
libontology-query --ontology unittests/datatest_rml.ttl  --query unittests/queries
/uc_domain_hijacking/test.sparql
[+] Ontology loaded successfully from unittests/datatest_rml.ttl
[*] Query Results:
jeu.example.org jeu.webagencymorte.fr
jeu2.example.org jeu.webagencymorte.fr
jeu2.example.org jeu2.example.org.webagencyintermediaire.com
jeu4.example.org jeu4.example.org.webagencyintermediaire.com
malicious.example.org malicious.webagencymorte.fr
[+] Query executed successfully.
[onto] acawet ~/ontologie/dseco/ontology/dseco-latest (try_to_optimize_the_code)$
```

jeu.example.org est géré par example.org mais passe par un CNAME

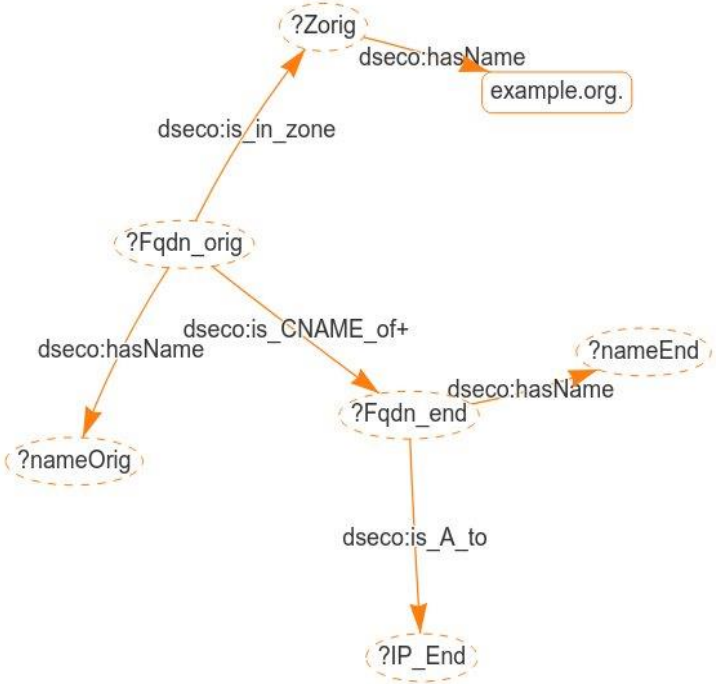
jeu.webagencymorte.fr géré par webagencymorte.fr

Requête
Cas IP hijacking

IP hijacking

A/AAAA hors gestion Orange : détection des IPs à surveiller de près.

- Utilisation d'une propriété issue de la phase d'enrichissement
- Temps de requêtage: quelques secondes pour obtenir les résultats



Requête IP hijacking

Enrichissement : on déclare nos sous réseaux

```
PREFIX dseco: <https://w3id.org/dseco/ontology/>
PREFIX observable: <https://ontology.unifiedcyberontology.org/uco/observable/>
BASE <https://w3id.org/dseco/mykg/>

INSERT { ?ip dseco:isinOurSubnet true . }
WHERE {
    ?ip a observable:IPAddress .
    ?ip dseco:is_part_of ?SUBNET .
    VALUES ?SUBNET
        {
            <https://w3id.org/dseco/mykg/SUBNET_203.0.113.0/24>
            <https://w3id.org/dseco/mykg/SUBNET_198.51.100.0/24>
        }
}
```

Requête IP hijacking

```
SELECT ?nameOrig ?IP_End
WHERE {
      ?Fqdn_orig dseco:is_CNAME_of+ ?Fqdn_end .
      FILTER NOT EXISTS {
        ?Fqdn_end dseco:is_CNAME_of ?_OtherFQDN
      }

      ?Fqdn_orig dseco:hasName ?nameOrig .
      ?Fqdn_orig dseco:is_in_zone ?Zorig .
      ?Zorig dseco:hasName "example.org."^^xsd:string .

      ?Fqdn_end dseco:hasName ?nameEnd .
      ?Fqdn_end dseco:is_A_to ?IP_End .

      # The interesting part of the query
      FILTER NOT EXISTS {
          ?IP_End dseco:isinOurSubnet "true"^^xsd:boolean .
      }


}
# to ensure that the order is maintained, this way testing the ontology or the inferred ontology will do the same.
ORDER BY ?nameOrig ?nameEnd
```

Requête IP hijacking

Des couples *nameOrig* ↦ *IP_End* à risque sont identifiés

```
[onto] acawet ~/ontologie/dseco (try_to_optimize_the_code)$ libontology-query --ontology ontology/
dseco-latest/unittests
/datatest_rml.ttl --query ontology/dseco-latest/unittests/queries/uc_ip_hijacking/test.sparql
[+] Ontology loaded successfully from ontology/dseco-latest/unittests/datatest_rml.ttl
[*] Query Results:
jeu.example.org https://w3id.org/dseco/mykg/IP_12.12.12.12
jeu2.example.org https://w3id.org/dseco/mykg/IP_12.12.12.12
malicious.example.org https://w3id.org/dseco/mykg/IP_199.1.1.3
outsidesubnets2.example.org https://w3id.org/dseco/mykg/IP_199.1.1.2
[+] Query executed successfully.
```

Jeu.example.org a une IP finale 12.12.12.12 qui n'appartient pas à notre subnet

De la faille à la requête
Retour d'expérience sur le principe

Avantages

- Temps de requête : quelques secondes pour obtenir les résultats.
- Aucun code à écrire, seulement lancer des requêtes SPARQL pré-écrites.
- Langage humain plus qu'un langage technique.
- Les requêtes sont universelles : on peut les partager au sein de la communauté.

Inconvénients

- Dépend de la qualité de la donnée.
- Il peut y avoir des faux positifs (manque de données et/ou trous dans le graphe).

DSecO, un projet en open source (BSD-4-Clause)
Démarrage rapide

https://w3id.org/dseco/

Des étapes simples …

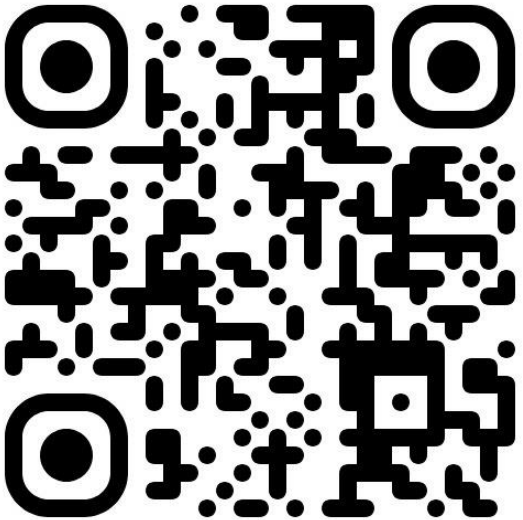
- Cloner le projet

```
git clone https://github.com/Orange-OpenSource/dseco.git
```

- Instancier & requêter

```
sh ./ontology/dseco-1.7.0/unittests/unittests_all.sh
```

- Analyser les réponses



Ontologie,
documentation
et exemples inclus

Conclusions & perspectives

Problème

Sécuriser le DNS face à des failles multiples et évolutives.

Approche

Représentation explicite des données de configuration dans un graphe de connaissances, intégration automatisée des données, requêtes sans code.

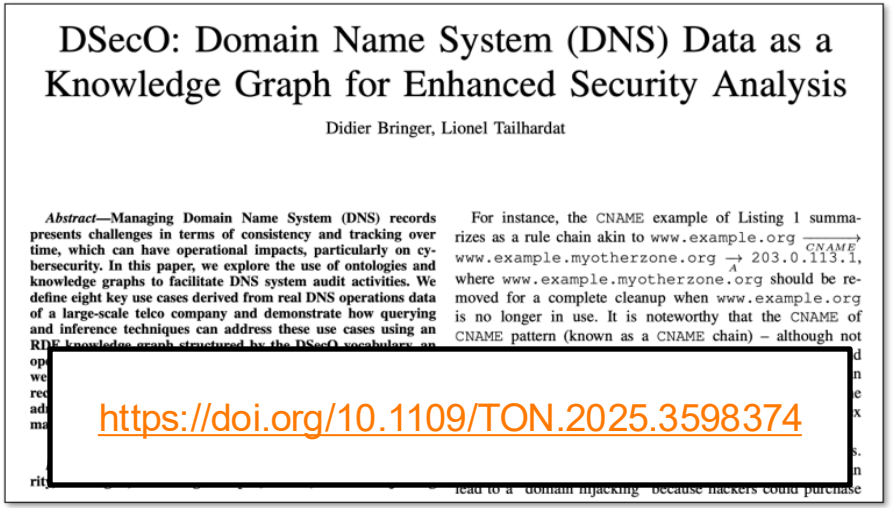
Résultats

Nettoyage massif de configurations complexes, gain de temps et de confiance dans l'analyse, adoption facilitée par les équipes opérationnelles.

Travaux futurs

Extension à d'autres sources (Gitlab, SBOMs, SAST, WAF, etc.), enrichissement de l'ontologie …

N'hésitez pas à utiliser et contribuer au projet, c'est en open source !



DSecO: Domain Name System (DNS) Data as a Knowledge Graph for Enhanced Security Analysis

Didier Bringer, Lionel Tailhardat

***Abstract*—Managing Domain Name System (DNS) records presents challenges in terms of consistency and tracking over time, which can have operational impacts, particularly on cybersecurity. In this paper, we explore the use of ontologies and knowledge graphs to facilitate DNS system audit activities. We define eight key use cases derived from real DNS operations data of a large-scale telco company and demonstrate how querying and inference techniques can address these use cases using an**

For instance, the CNAME example of Listing 1 summarizes as a rule chain akin to www.example.org $\xrightarrow{CNAME}$ www.example.myotherzone.org $\xrightarrow[A]{}$ 203.0.113.1, where www.example.myotherzone.org should be removed for a complete cleanup when www.example.org is no longer in use. It is noteworthy that the CNAME of CNAME pattern (known as a CNAME chain) – although not

https://doi.org/10.1109/TON.2025.3598374