

CAS 5.2 – Retour d'expérience Aix-Marseille Université

Pourquoi CAS V5 ?

Bonne pub de Misagh Moayyed aux EsupDays :

- Rendre les mises à jour plus faciles
- Un seul fichier de configuration cas.properties
- Fini les dizaines de fichiers xml à éditer !

Cependant..

Jolie documentation, mais difficile à exploiter. On reboucle souvent.

Pas de bug report : il faut soumettre des patches.

Migration pré-requis

Notre CAS était en version 4.0 et nous devions supporter ces services qui a priori pouvaient poser problème:

- Esup-filemanager qui est une portlet
 - dépend de uPortal pour clearPass (récupération du mot de passe)
- AGIMUS
 - dépend de la version du CAS car il gère la correspondance cookie TGC ↔ identifiant

clearPass CASV5

Changement total de stratégie

- Auparavant :
 - Une URL spécialisée avec un mode proxy interne au serveur CAS
- CAS V5 :
 - Le service et le CAS partagent un secret (clef publique/privée)
 - Le mot de passe (credential) est envoyé comme un attribut normal chiffré

Uportal 4 ne sait pas gérer ce mode !

clearPass esup-filemanager

Pré-requis : On ne veut pas toucher à uPortal4

Modifications d'esup-filemanager pour CAS V5

- CAS gère le mode proxy
- Uportal fait du proxy
- Esup-filemanager fait un appel proxy
 - Récupère le credential
 - Et peut le jouer sur les partages

Un grand merci à Vincent Bonamy pour ce développement

Agimus

On ne veut pas perdre nos statistiques

- Février 2018 annonce du portage pour CAS V5
- Installation simple

C'est parti, on va pouvoir migrer !

CAS V5 Configuration

CAS V5

Cette version est très riche et ne se contente pas du seul protocole CAS. Elle supporte entre autre :

- SAML – Shibboleth (testé avec ADFS)
- oAuth2
- OpenidConnect
- MFA (Multi factor authentication) Testé avec Google Authenticator
- Kerberos
- Délégation à une source externe (ex : FranceConnect)

CAS V5 - configuration

On doit définir dans **/etc/cas/cas.properties** le ou les backends d'authentification (chaînage possible par service)

- LDAP (authentification + attributs)

Et si besoin

- ActiveDirectory (comptes particuliers)
- Kerberos

L'attribut repository (nécessaire si on a plusieurs backends comme kerberos ou active directory)

Les attributs à renvoyer par défaut

La gestion des tickets (redis, memcached), etc..

CAS V5 – Les services

Ils sont définis par des fichiers JSON (recommandé)

- On indique le protocole (CAS, SAML, OIDC..)
- l'expression régulière pour l'URL
- Quels attributs renvoyer
 - Si on veut une demande de consentement (oui/non)
- Le thème éventuel
- Par quels backends d'authentification passer
 - On peut en citer plusieurs
- Et un ordre pour parcourir les services.
 - Ceci peut-être testé via le Dashboard (Attribute resolution & release)

Exemple de service

Service AMU de base :

```
{  
  "@class" : "org.apereo.cas.services.RegexRegisteredService",  
  "serviceId": "^(https|http)://[a-z0-9-]+((([.](xxx|yyy))))[.]univ[.]fr(:|/|$).*",  
  "name": "AMU",  
  "theme": "theme_amu",  
  "id": 2,  
  "description": "Amu",  
  "evaluationOrder":1000  
}
```

Sous-domaine

Attention aux
Expressions régulières


CASV5 Dashboard


CAS V5 Dashboard




CAS V5 Dashboard sessions

SSO Sessions Report


2
 Total Active Principals


3
 Usage Count Sessions


2
 Total SSO Sessions

[Remove All Sessions](#)

Show entries

	Principal	Ticket Granting Ticket	Authentication Date	Usage Count	
+	lalot	TGT-5-xrxods3Se-HljYRTJ2O2eqQBdoBn0s72t...	2018-06-17T14:38:31Z	0	Destroy
+	lalot	TGT-4-AihnA6COILNR0rYxrr-P-yqXFPNvm0KoZy...	2018-06-17T13:07:58Z	3	Destroy

Showing 1 to 2 of 2 entries

[Previous](#)
1
[Next](#)

Super en test, sauf que dans la réalité, on plante sur un timeout AJAX (trop de sessions)



Les nouveautés ou plutôt ce que nous avons testé

CAS V5 Service Management

Une application séparée :


- qui permet de gérer les services
- ceux-ci peuvent être sauvegardés dans différents backends.
- Le serveur CAS doit utiliser le même backend











Nous avons testé avec la base SQL et perdu beaucoup de temps (jpa autodestroy). Nombreuses bug pour supprimer, dupliquer les services.

Il vaut mieux utiliser le format JSON, le seul documenté.

Avantages / Inconvénients :

- + Plus facile pour générer les Json
- Le fichier Json est très verbose et pas très bien indenté
- Application lente à démarrer (autre instance tomcat conseillée)

 **Manage Services - default**

Name	Service Url	Description
 Alumni	https://a[redacted]u[.]fr(\$/).*	
 Ametice SAML	https://an[redacted]mu[.]fr	Test CAS SAML
 ENT de test uPortal4	https://up[redacted]fr(\$/).*	
 EntAMU-2	https://(e[redacted]v- amu.fr(\$/).*	
 NoubacNRS	^(http https)://n[redacted]rs[.]fr.*	
 NS1Univmed	^(http https)://ns[redacted]fr(: /\$).*	Pour le DNS maitre ns[redacted].fr
 Resa Fed3C CNRS	https://resa[redacted]fr(\$/).*	
 SciencesPoAix	^(http https)://[redacted] [redacted]x.fr(\$/).*	
 Tout-univ-aix-marseille.fr	^(https http)://[a-z][redacted] [redacted].fr(/ \$).*	
 ADFS UNIVAM	http://[redacted]services /trust	Test ADFS CAS SAML

Items per page: 10 1 - 10 of 20 < >

CAS V5 Kerberos

Inscrire le serveur CAS dans le domaine Kerberos

- Générer un keytab pour le serveur CAS pointé dans /etc/krb5.conf
- Dans cas.properties, mettre les clefs cas.authn.spnego

Côté Navigateurs :

- IE/Edge - Ajouter l'URL du serveur CAS à la zone Intranet locale
- Firefox - positionner **network.negotiate-auth.trusted-uris** au serveur CAS dans **about:config**

Et c'est magique !

- Machine dans le domaine
 - Utilisateur connecté
 - plus de mire CAS avec login/mdp

Attention, l'URL du serveur CAS ne doit pas être un CNAME !

CAS V5 Kerberos

Comment déployer, et comment vont réagir les machines hors domaine ?

- IE, Edge, Chrome ont un mode fallback en NTLM avec une vilaine page du style Auth Basic
- Firefox n'a pas ce problème, et on peut aussi le configurer via GPO

Restreindre Kerberos :

- Par IP : buggé mais résolu en 5.2.6
 - `cas.authn.spnego.ipsToCheckPattern=^(10.19.0.*|7.9.140.*)$`
- Par user-agent buggé et corrigé en 5.2.7+ par nos soins
 - `cas.authn.spnego.supportedBrowsers = Firefox`
 - On peut modifier le user-agent pour les navigateurs gérés dans le domaine. Mais ça ne marche que pour IE

Effet de bord :

- clearPass ne marche plus (plus de mot de passe..)
 - Patch dans esup-filemanager pour demander un login/mdp en cas d'échec

CAS V5 OpenidConnect

OpenID Connect est un protocole analogue à Shibboleth mais plus moderne

- Il utilise des JWT (JSON web token : flux JSON signé par le serveur)
- Utilisé et poussé en avant par Google, Facebook, Microsoft, FranceConnect

Testé avec succès pour contrôler des accès dans Apache (mod_auth_openidc)

- On reçoit les attributs en variables d'environnement comme shibboleth, mais en beaucoup plus simple

APACHE mod_auth_openidc

Il s'avère que c'est le moyen le plus pratique pour tester
openidc avec son mode debug

```
OIDCProviderMetadataURL https://cas.univ.fr/cas/oidc/.well-known/openid-configuration
OIDCClientID test-oidc-apache
OIDCClientSecret XXX
OIDCRemoteUserClaim sub
```

```
OIDCScope "openid email profile"
OIDCRedirectURI https://test.univ.fr/secureoidc/redirect_uri
# voir "jwks": "file:/etc/cas/oidc.jwks"
OIDCCryptoPassphrase XXX
# LogLevel info auth_openidc:debug
<Location /secureoidc>
  AuthType openid-connect
  Require user xxx
</Location>
```

CAS V5 OpenidConnect : le service

```
{
  "@class": "org.apereo.cas.services.OidcRegisteredService",
  "serviceId": "https://test.univ.fr/secureoidc/redirect_uri",
  "name": "Test OpenidConnect",
  "id": "1521564200984",
  "scopes" : [ "java.util.HashSet", [ "profile","email" ] ],
  "clientSecret": "XXX",
  "clientId": "test-oidc-apache",
  "bypassApprovalPrompt": false,
  "generateRefreshToken": false,
  "jsonFormat": true,
  "jwks": "file:/etc/cas/oidc.jwks",
  "signIdToken": true,
  "encryptIdToken": false,
  "idTokenEncryptionAlg": "A256GCMKW",
  "subjectType": "public",
  "attributeReleasePolicy": {
    "@class" : "org.apereo.cas.services.ReturnMappedAttributeReleasePolicy",
    "allowedAttributes" : {
      ....
    }
  }
}
```

MFA

Authentication multi-facteur

Multi-Factor-Authentication

Ne pas reposer uniquement sur les login/mdp (phishing, mot de passe faible). On demande une preuve d'identité supplémentaire (Clefs U2F, Google Authenticator, ..)

Comment l'activer ? Dans la définition json du service, ajouter ceci :

```
multifactorPolicy:  
{  
  @class: org.apereo.cas.services.DefaultRegisteredServiceMultifactorPolicy  
  multifactorAuthenticationProviders: [  
    java.util.HashSet [mfa-gauth]  
  ]  
}
```

Remarque : On peut rendre la chose très pénible et ajouter d'autres facteurs

MFA (Google Authenticator)

Les clefs des utilisateurs sont créées automatiquement et sauvegardées à la première connexion :

- En base : `cas.authn.mfa.gauth.jpa.database.url=xxx`
- Ou au format json :
`cas.authn.mfa.gauth.json.location=xxx`

Remarques :

- En cas de perte, il faut une intervention d'un administrateur.
- Peut-on demander d'utiliser son smartphone personnel pour un usage professionnel, est-ce déployable et pour combien d'utilisateurs ?

Interrupt Authentication

La cerise sur le gâteau ! Pouvoir bloquer l'authentification sur des critères. Par exemple, suivre la procédure d'activation du compte, ou adresse mail personnelle non fournie, etc..

cas.interrupt.groovy.location=file:/etc/cas/config/interrupt.groovy

Le script reçoit :

- Les attributs LDAP
- L'uid (principal)
- Le service

On peut soit :

- bloquer le mode SSO, écrire un message, et avoir un bouton avec un lien.
- Ou laisser passer mais juste pour un service

Exemple: Forcer l'activation des comptes

```
def run(final Object... args) {  
  def uid = args[0]  
  def attributes = args[1]  
  def service = args[2] ? args[2] : 'null'  
  def logger = args[3]  
  def message = "Vous n'avez pas encore validé la charte informatique."  
  def links = [sesame:"https://sesame.univ.fr/"]  
  def block = true  
  def unblock = false  
  def ssoDisabled = false  
  if (!attributes.DateActivation){  
    if (service.contains("sesame.univ.fr")){  
      return new InterruptResponse("Cliquez sur continuer pour finaliser l'activation de votre  
compte",null, unblock, ssoDisabled)  
    }  
    else{ // juste un exemple de test  
      if ((attributes.eduPersonAffiliation) && (attributes.eduPersonAffiliation.contains("student"))){  
        links = [sesame:"https://sesame.univ.fr/ActivationEtud.php"]  
      }else{  
        links = [sesame:"https://sesame.univ.fr/Activation.php"]  
      }  
      return new InterruptResponse(message, links, block, ssoDisabled)  
    }  
  }  
}
```

Le script est relu dynamiquement.

Interrupt Authentication

The image shows a screenshot of a web application's authentication interruption dialog. The dialog has a light gray background and rounded corners. At the top, the title "Interruption de l'authentification" is displayed in a large, bold, black font. Below the title, a message reads "Bonjour, [nom]. L'authentification a été interrompue pour la raison suivante :" in a smaller black font. Underneath this message is a light blue rectangular box containing the text "Vous n'avez pas encore validé la carte informatique." in a light blue font. Below this box are two buttons: a blue button with the text "Sesame" and an orange button with the text "Annuler". A large, white, tilted rectangular box with a black border is overlaid on the dialog, containing the text "ATTENTION : En cas d'erreur dans le script, on laisse tout passer !" in a bold black font.

Passage en production

Mise en production

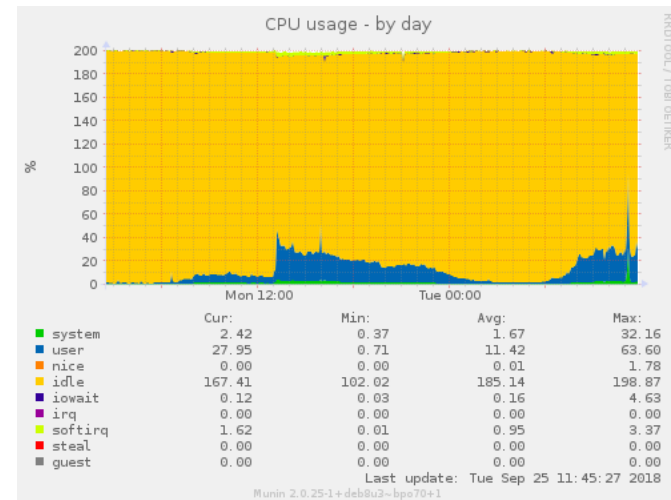
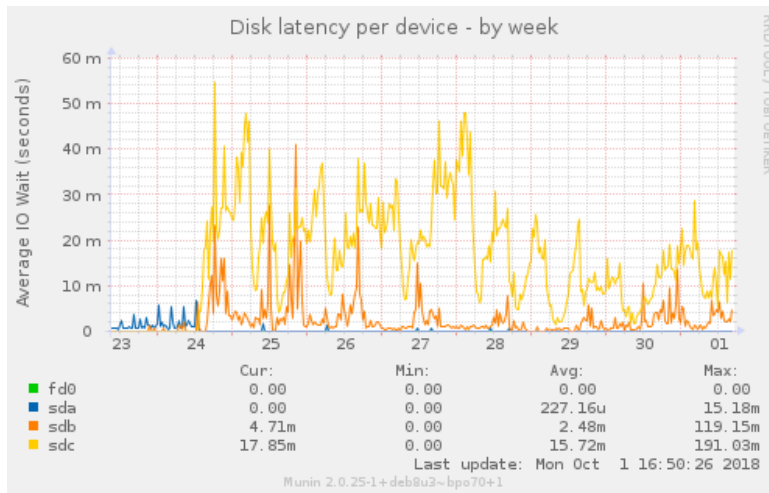
Plusieurs essais

- Fin juin
 - Il manquait quelques services mais..
 - Date délicate (inscriptions, vacances)
- 11/09 (30 minutes)
 - Charge importante (ajout RAM + CPU)
 - Problème avec une application
 - Logs montrant des erreurs diverses (finalement pas inquiétant)
- Fin septembre
 - Idem : problème openssl, mise à jour d'un vieux serveur applicatif de ubuntu 11.04 vers 12.04 (CAS en A+)
 - Le login renvoyé était ce que l'utilisateur tapait.
 - Redmine : faire en sorte de renvoyer les attributs en V2
 - `cas.view.cas2.success=protocol/2.0/casServiceValidationSuccess-AMU`

Hardware

Consommation un peu plus importante

- 2 processeurs (application démarre plus vite)
- Plus de mémoire (Redis 1Go)
- Consommation I/O plus importante



Corrections

Lire le code source
RTFS

- Dashboard plante sur timeout Ajax. Le script lit toutes les sessions sur Redis et ramène les attributs LDAP. 70000 sessions et 110Mo de données :-(
 - `cas.http-client.asyncTimeout=PT60S` (5 secondes sinon)
- Cache des attributs : on ne voulait pas activer de cache.
 - Stockés avec les sessions, on n'a pas réussi à dissocier
 - Limitation des attributs à conserver:mapper les attributs

RTFS

- `cas.authn.attributeRepository.ldap[0].attributes.uid=uid`
`cas.authn.attributeRepository.ldap[0].attributes.mail=mail`

...

- Protocole V2 et attributs. Syntaxe thymeleaf non triviale :
 - `<span th:utext="'<cas:' + ${attr.key} + '>'" th:remove="tag"><span th:utext="'</cas:' + ${attr.key} + '>'" th:remove="tag">`

Bénéfices

- Tickets gérés par REDIS (mieux que memcached)
 - Redémarrage en 40 secondes (2 CPU) sans perte de sessions (y compris avec reboot du serveur)
- Relecture des services dynamiquement
- Le Dashboard (test des URL, attributs, sessions)
- Kerberos
- Interrupt Authentication (killer feature)
- Et possibilité d'utiliser d'autres protocoles que le CAS tout en étant dans le même SSO

Conclusion

Remerciements au GT-Auth
et à Vincent Bonamy pour leur aide

Où peut-on déposer notre configuration pour
aider la communauté?