ALCASAR : projet libre et gratuit de gestion des accès à Internet

ALCASAR : a free and open source Internet access management project

www.alcasar.net

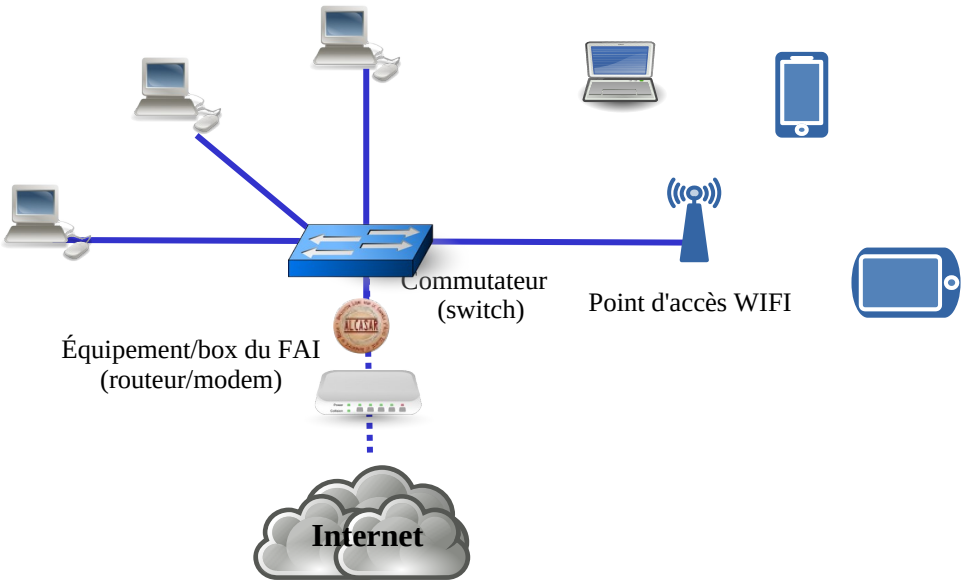
Documentation technique

Projet : Gestionnaire d'accès à Internet	Auteur : Rexy with helps by alcasar team
Objet : Documentation technique du projet	Version : 3.8.0
Mots clés : Contrôleur d'accès au réseau, Network Access Control (NAC), portail captif, captive portal, coovachilli	Date : juin 2026

Table des matières

1 - Architecture générale

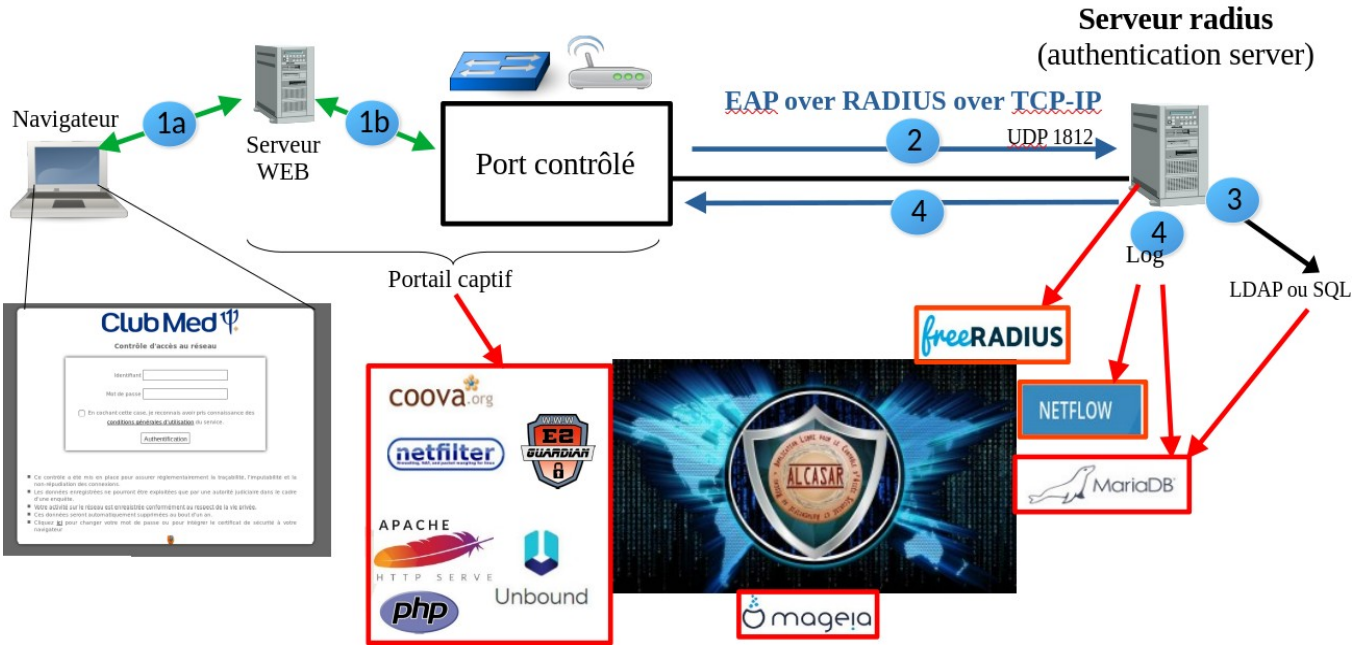
ALCASAR est positionné en coupure entre l'accès Internet et un réseau de consultation. Il permet d'authentifier les utilisateurs, de contrôler et filtrer les accès, de tracer les connexions effectuées, de protéger le réseau de consultation.



Le cœur d'ALCASAR est constitué des éléments traditionnels d'un contrôleur d'accès au réseau (NAC = Network Access Control) :

- Un serveur d'authentification AAA (freeradius) connecté à une base de données utilisateurs (mariadb) ;
- Une passerelle d'interception (portail captif) constituée d'un client radius (coovachilli), d'un serveur DHCP (coovachilli) et d'un serveur WEB (apache) ;
- Un système de filtrage constitué d'un pare-feu dynamique (netfilter + ipset), de quatre serveurs DNS (unbound) pour chaque profil (« blacklist », « whitelist », « blackout » et « sans filtrage »), et d'un proxy HTTP/HTTPS (E²Guardian) ;
- Un système de traçabilité et de visualisation des flux réseau (sonde Netflow et grapheur Nfsen-ng).

Le schéma suivant présente les briques principales d'ALCASAR par rapport à un schéma traditionnel 802.1X.

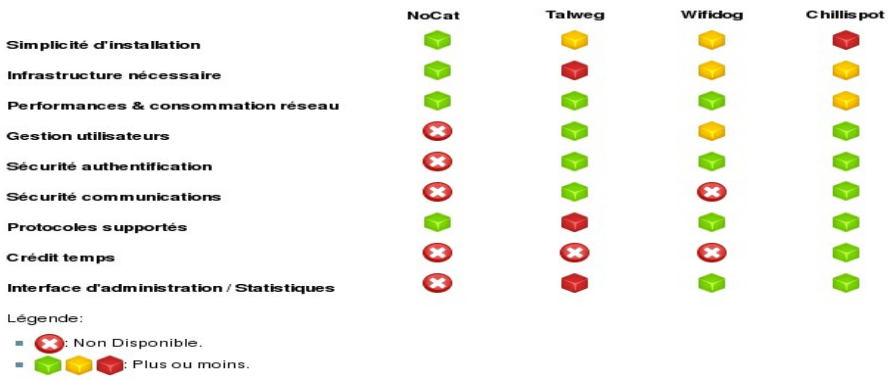


2 - Constituants

2.1 - La passerelle d'interception

Avec le pare-feu, la « passerelle d'interception » (ou portail captif) constitue le chef d'orchestre de cet ensemble. Pour choisir celle qui serait intégrée dans ALCASAR, les passerelles libres suivantes ont été évaluées au lancement du projet :

	NoCat	Talweg	Wifidog	Chillispot
Site WEB	nocat.net	talweg.univ-metz.fr	dev.wifidog.org	www.chillispot.org
Langage	C	C# sous mono	- C pour le programme principal - module PHP pour le serveur WEB	- C pour le programme principal - module CGI-BIN pour le serveur WEB (PERL ou C)
Description	NoCat est constitué de plusieurs éléments : « NoCatSplash » est le portail, « NoCatAuth » est utilisé pour l'authentification et « Splash Server » est le service permettant de générer les formulaires de connexion des utilisateurs. Le suivi de ce produit a été arrêté en 2005.	Talweg est un portail dont le contrôle d'accès au réseau est géré protocole par protocole. Tous les protocoles utilisables sur Internet ne sont pas encore intégrés.	WifiDog est composé de 2 modules : « Authentification Server » et « WifiDog Gateway ». Le serveur d'authentification doit être installé sur un serveur fixe alors que la passerelle peut être embarquée dans certains équipements réseau compatibles (routeur, passerelle ADSL, etc.).	Chillispot ne constitue que la partie centrale d'une architecture de type portail captif. Il implémente les 2 méthodes d'authentification (UAM et WPA). Il nécessite la connaissance et l'installation des autres services constitutifs du portail captif. Il est compatible radius.



Bien que cette liste ne soit pas exhaustive, la passerelle « Chillispot » a été choisie. Depuis, elle a été remplacée par le clone (fork) « CoovaChilli » https://coova.github.io/CoovaChilli/

Le code de ce projet est disponible sur Github (https://github.com/coova/coova-chilli/).

Avant chaque nouvelle version d'ALCASAR, le code source du projet « coova-chilli » est récupéré, compilé spécifiquement pour ALCASAR et empaqueté (RPM) pour être intégré à la distribution Linux exploitée par ALCASAR (Mageia).

Le tableau ci-après présente les constituants principaux d'ALCASAR et leurs évolutions au fil des versions :

2.2 - Liste des constituants et leur évolution

Pour couvrir l'ensemble des besoins d'ALCASAR, les produits libres suivants ont été intégrés. Leur choix est principalement dicté par leur niveau de sécurité et leur reconnaissance au sein de la communauté du logiciel libre.

Versions majeures d'ALCASAR / Constituants		<1.7	1.7	1.8	1.9	2.0	2.1-2.2	2.3-2.5	2.6	2.7	2.8	2.9	3.0	3.1	3.2	3.3-3.4	3.5	3.6	3.7	**3.8.0**
Système d'exploitation	Linux Mandriva	2007	2009	2010.0		2010	2010	2010	2010											
	Mageia									2	2	4.1	5.0	5.1	6	6	7.1	8	9	**9**
noyau Linux		2.6.1	2.6.2	2.6.31		2.6.33				3.4.4	3.4.5	3.14.4	4.4.13	4.4.74	4.9.5	4.14.89	5.10.46	5.15.126	6.6.88	**6.6.141**
Trace des connexions (sonde noyau NetFlow + collecteur)	IPT_Netflow	1.8											2.0	2.2	2.3		2.6	2.6.1	2.6.5	**2.6.6**
	Nfdump												1.6.13	1.6.15	1.6.19		1.6.23	1.7.3	**1.7.4**	
Passerelle d'interception	CoovaChilli	1.0	1.0.12		1.2.2		1.2.5	1.2.8	1.2.9	1.3.0			1.3.2	1.4.3			1.6			**1.8.0**
DNS	Unbound																1.10.1	1.17.1	**1.24.2**	
Serveur DHCP (mode bypass)	Dnsmasq					2.52-1				2.63		2.66	2.75	2.77		2.77-1				
	dhcpd server	3.0.4	3.0.7	4.1.0													4.4.1	4.4.2	**4.4.3**	
Serveur Web	Apache	2.2.3	2.2.9	2.2.14		2.2.15			2.2.2	2.2.2	2.2.2	2.4.7	2.4.10	2.4.27					**2.4.67**	
	Lighttpd														1.4.51		1.4.53	1.4.59		
PKI locale (chiffrement des flux)	OpenSSL					1.0.0.a				1.0.0.k		1.0.1p	1.0.2h	1.0.2k	1.0.2.n	1.0.2q	1.1.0l	1.1.1v	**3.0.20**	
Middleware	PHP	5.1.6	5.2.6	5.3.1		5.3.4	5.3.6		5.3.1	5.3.1	5.3.2	5.5.22	5.6.22	5.6.30	5.6.32	5.6.38	7.3.29	8.0.30	**8.2.31**	
Serveur d'authentification	FreeRadius	1.1.2	2.1.0	2.1.7		2.1.8		2.1.8.6		2.1.12		2.2.0	2.2.9	2.2.10	3.0.15		3.0.22		**3.0.27**	
Serveur de base de données utilisateurs	Mysql	5.0.2	5.0.6	5.1.4			5.1.5		5.1.6											
	MariaDB									5.5.25		5.5.41	10.0.24	10.0.31	10.1.29	10.1.35	10.3.29	10.5.22	**11.4.10**	
Cache WEB (proxy)	Squid	2.6	3.0.8	3.0.22		3.1.14				3.1.19							cf. e2guardian			
	tinyproxy											1.8.3								
Serveur de temps	ntpd	4.2	4.2.4						4.2.6			4.2.6p5			4.2.8					
	ntpsec																		**1.2.2**	
Journalisation	Ulogd	1.24									2.0.2	2.0.4	2.0.5				2.0.7		**2.0.8**	
Filtrage d'URL WEB + Proxy	SquidGuard	1.2.0																		
	Dansguardian		2.9.9	2.10.1								2.12.0								
	E²Guardian														3.5.0		5.3.2	5.3.4	**5.5.6**	
Statistiques de consultation	Awstat	2.5	2.5.3	6.9		6.95				7.0										
	Nfsen										1.3.7		1.3.8							
Détection d'intrusion	Fail2ban										0.8.6	0.8.13		0.9.5			0.10.4	**1.0.2**		
Chiffrement des fichiers journaux	Gnupg	1.4.5	1.4.9	1.4.10						1.4.12		1.4.16	1.4.19	1.4.22	1.4.23		2.2.19	2.2.36	**2.3.8**	
Connexion distante sécurisée	openssh-server	4.3	5.1	5.3		5.5				5.9		6.2	6.6	7.5			8.0	8.4	**9.3p1**	
Antimalware	LibClamav				0.96		0.97					0.98	0.99		0.100					
	clamd																0.103			
Serveur de courrier	Postfix									2.8.8		2.10.2	2.10.3	3.1.6			3.4.5	3.5.9	**3.8.4**	
Authentification Email	Cyrus SASL																**2.1.27**			
Gestionnaire de SMS	Gammu-smsd											1.32	1.34	1.35	1.39		**1.42**			
Générateur de fichiers PDF	wkhtmltopdf														**0.12.3**					
Intégration Let's Encrypt	Acme.sh														2.8.6		3.0.4	3.1.1	**3.1.3**	
Front-end (Framework)	Bootstrap														3.3.7		**4.6.1**			
Front-End (Librairie JS)	Jquery / ui														3.2/1.10		**3.6/1.13**			
Front-End (Info système)	Phpsysinfo	2.5.3															3.4.3		**3.4.5**	

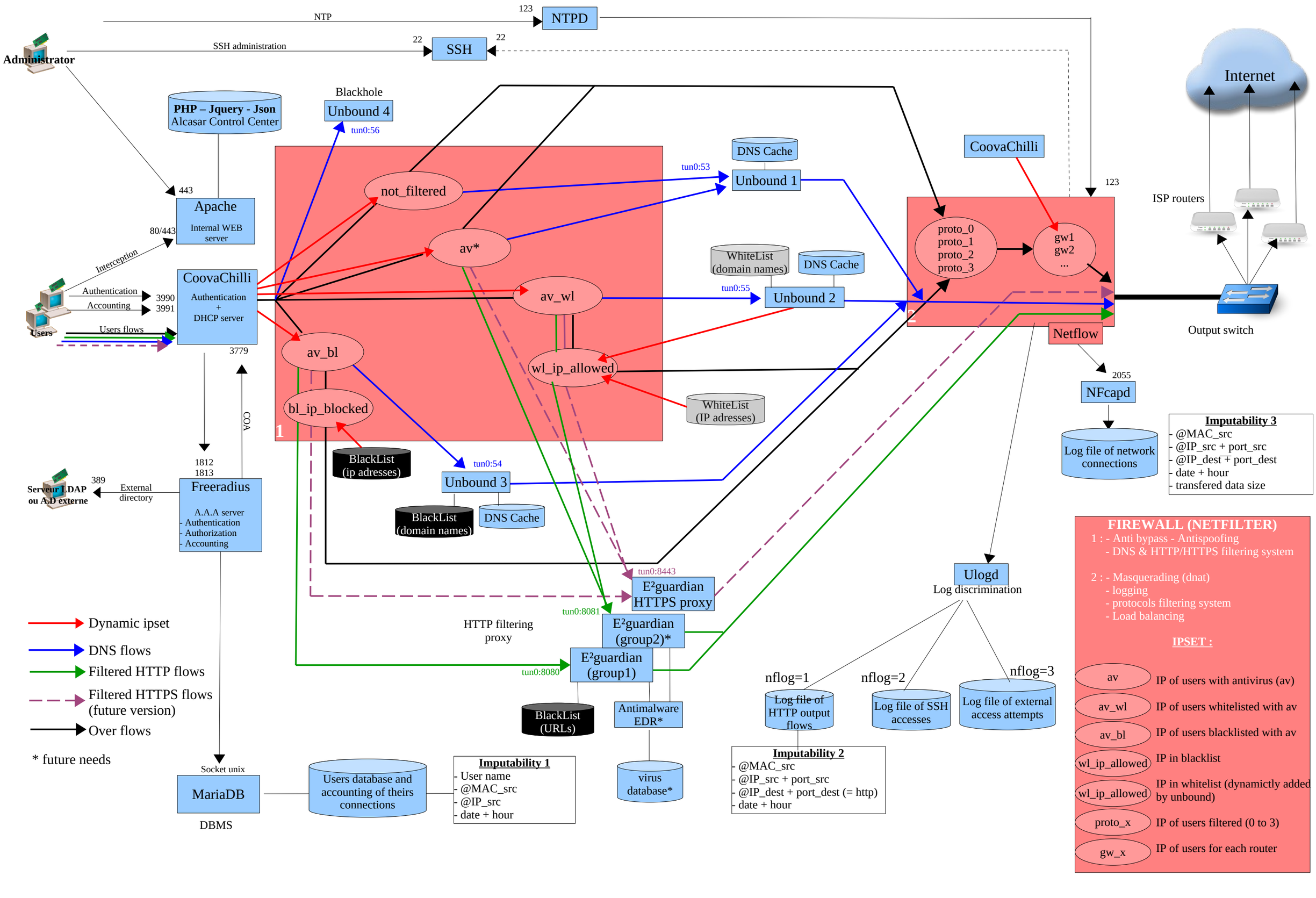
3 - Architecture technique :

ALCASAR peut être décomposé en cinq fonctions qui sont détaillées dans la suite du document :

- La fonction « interception / authentification / autorisation » est réalisée par CoovaChilli, Unbound, Apache et le couple (Freeradius, MariaDB). Pour l'authentification, la connexion à un annuaire externe (LDAP/A.D.) est possible ;
- La fonction « traçabilité / imputabilité des connexions » est constituée d'une sonde Netflow, des journaux du pare-feu géré par le daemon de journalisation Ulog et du couple (Freeradius, MariaDB) ;
- La fonction de « filtrage » (de domaine, d'URL, d'adresses IP et de protocoles réseau) est réalisé par le pare-feu (Netfilter) associé à des ipset dynamiques, 4 instances de Unbound et le proxy HTTP/HTTPS « E²Guardian) » ;
- La fonction « interface de gestion » appelée ALCASAR Control Center (ACC) est réalisée en PHP / HTML4 & 5/ Jquery. Elle est servie par Apache ;
- Des modules complémentaires ont pour objectif d'améliorer la sécurité globale du portail (anti-contournement, anti-usurpation MAC/IP, chiffrement des fichiers journaux, gestion des certificats, détection d'intrusion machine (HIDS), etc.) ou d'enrichir les possibilités du portail (installation, mise à jour, by-pass, archivage, accélération de la consultation (cache DNS, cache HTTP/HTTPS), etc.).

Le schéma d'architecture suivant présente les processus principaux d'ALCASAR et les flux de données transitant entre ceux-ci.

ALCASAR – ARCHITECTURE



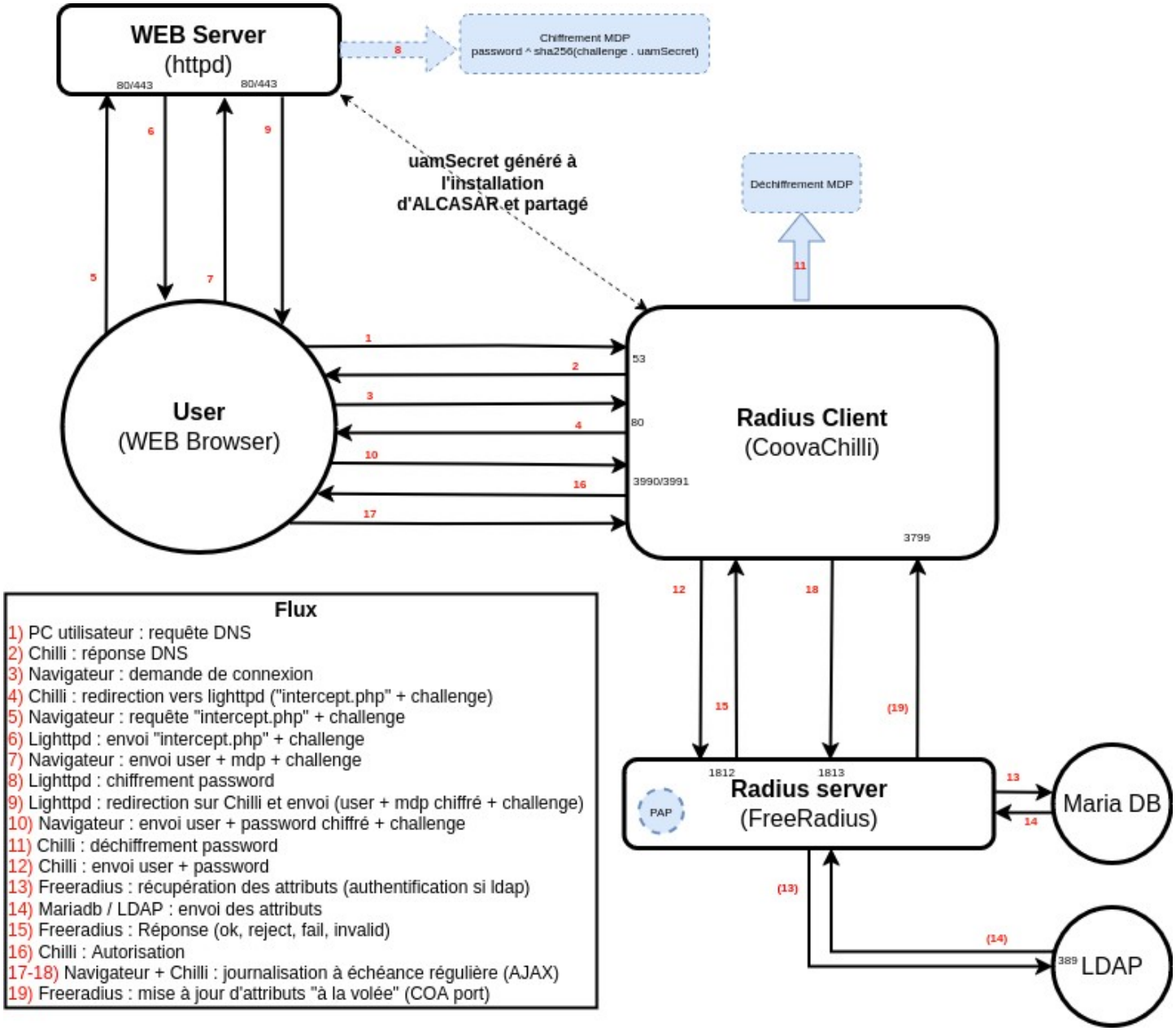
4 - Fonction « interception / authentification / autorisation »

Afin d'être le plus universel possible, ALCASAR s'appuie sur la méthode « 802.1X-UAM » (Universal Access Method). Cette méthode n'utilise que des protocoles standards ne nécessitant qu'un navigateur WEB pour authentifier un utilisateur. Les autres méthodes, nécessitent qu'un agent radius soit installé dans chaque équipement.

La fonction « interception / authentification » s'appuie sur les 4 briques logicielles suivantes :

- « CoovaChilli » (processus « chilli ») gère les fonctions de « portail captif », de « client radius » et de « serveur DHCP ». il est configuré pour exploiter la méthode 802.1x-UAM (Universal Access Method) ;
- « Apache » (processus « httpd ») est le serveur WEB présentant les pages web aux utilisateurs (et l'ACC aux administrateurs) ;
- « FreeRadius » (processus « radiusd ») est le serveur d'authentification, d'autorisation et de journalisation (Authentication, Authorisation & Accounting - AAA) de type radius ;
- « Mariadb » (processus « mariadbd ») est le système de gestion de bases de données stockant la base des utilisateurs (base « radius »). Mariadb gère aussi la base de données liée à la gestion des SMS (base « gammu »).

L'architecture de cette fonction est décrite dans le schéma suivant :

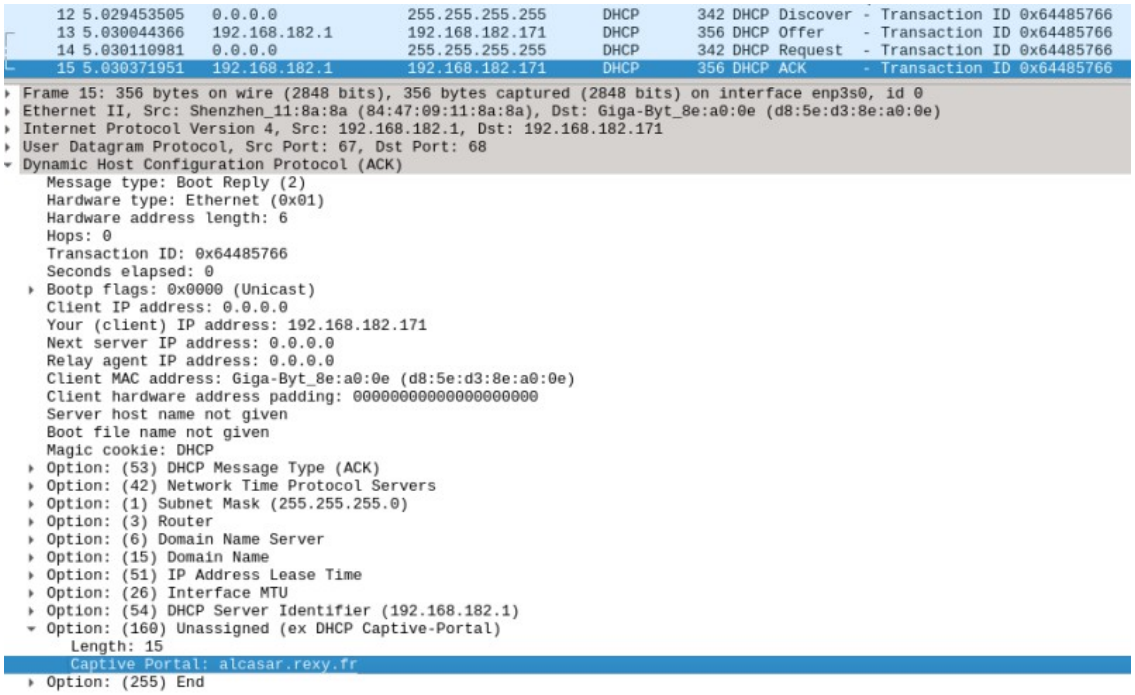


4.1 - Le portail captif et client radius

« CoovaChilli » est lancé via son unité de démarrage *(/etc/systemd/system/chilli.service).* Son fichier de configuration est *« /etc/chilli.conf »*. Le processus « chilli » est lancé en mode « daemon ». Ce dernier crée l'interface virtuelle « tun0 »[1] liée en « point à point » à l'interface physique connectée au réseau de consultation ($INTIF). Cette configuration permet de gérer le cache ARP en espace utilisateur autorisant ainsi le verrouillage des couples (@MAC, @IP) acquis en écoutant le réseau de consultation (gestion semi-statique du cache). Un empoisonnement du cache ARP par le réseau est alors impossible (« cache poisoning »). Dans certains cas, ce comportement peut être bloquant. Exemple : lorsqu'on reparamètre l'@IP d'un équipement qui a déjà été enregistré par ALCASAR. La commande *« chilli-query list »* permet d'afficher et de contrôler le cache ARP de « chilli ». Cette commande est utilisée par le centre de contrôle d'ALCASAR – ACC (menu « ACTIVITÉ ») pour gérer ces cas. Complémentaire à cette fonction d'anti « cache poisoning », ALCASAR utilise un module spécifique de sécurité (*alcasar-watchdog.sh*) qui permet d'éviter l'usurpation d'adresses MAC et d'adresses IP sur le réseau de consultation (cf. fonctions de sécurité).

4.1.1 - DHCP

La capture suivante montre l'étape d'attribution de la configuration IP aux équipements par le serveur DHCP intégré à « Coovachilli ». On remarque que la réponse DHCP intègre l'option 160 « Captive-Portal DHCP/RA » qui permet de fournir aux clients l'URI d'un portail captif (RFC 7710). En septembre 2020, les RFC 8908 et RFC 8910 impose le remplacement du numéro d'option (114 au lieu de 160). Le code de chilli a été modifié en conséquence en octobre 2025.

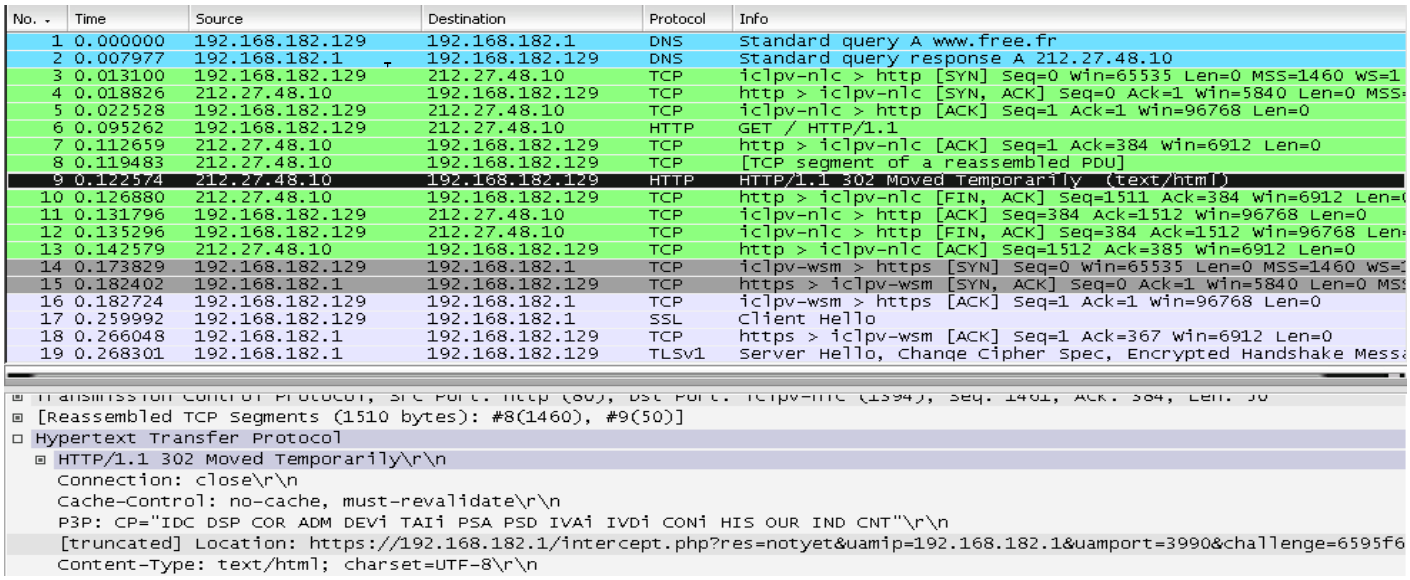


Comme ALCASAR présente un serveur de temps sur le réseau de consultation, l'option 42 (Network Time Protocol Server) du serveur DHCP a aussi été configuré.

1 - Les périphériques « Tap » et « Tun » des noyaux Linux sont des interfaces réseau virtuelles de niveau 2 (Ethernet) pour « Tap » ou 3 (IP) pour « Tun » permettant à des processus exécutés en espace utilisateur (userland) d'envoyer ou de recevoir des trames/paquets sur ces interfaces via les fichiers spéciaux (/dev/tapX ou /dev/tunX). Ces interfaces virtuelles peuvent être exploitées exactement comme les interfaces physiques qui fonctionnent en mode « noyau » (configuration, émission/réception, routage). Ainsi, ces interfaces virtuelles autorisent un traitement sur les trames à la réception ou avant l'émission de celles-ci. L'interface Tap est souvent utilisée dans la création de tunnels RPV/VPN afin d'encapsuler un flux dans un autre (cf. projet « OpenVPN »).

4.1.2 - Fonctionnement de l'interception/authentification

La capture suivante montre les différentes étapes constitutives de l'interception/authentification de type « UAM » implémenté dans ALCASAR :



No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.182.129	192.168.182.1	DNS	Standard query A www.free.fr
2	0.007977	192.168.182.1	192.168.182.129	DNS	Standard query response A 212.27.48.10
3	0.013100	192.168.182.129	212.27.48.10	TCP	iclpv-nlc > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
4	0.018826	212.27.48.10	192.168.182.129	TCP	http > iclpv-nlc [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=
5	0.022528	192.168.182.129	212.27.48.10	TCP	iclpv-nlc > http [ACK] Seq=1 Ack=1 Win=96768 Len=0
6	0.095262	192.168.182.129	212.27.48.10	HTTP	GET / HTTP/1.1
7	0.112659	212.27.48.10	192.168.182.129	TCP	http > iclpv-nlc [ACK] Seq=1 Ack=384 Win=6912 Len=0
8	0.119483	212.27.48.10	192.168.182.129	TCP	[TCP segment of a reassembled PDU]
9	0.122574	212.27.48.10	192.168.182.129	HTTP	HTTP/1.1 302 Moved Temporarily (text/html)
10	0.126880	212.27.48.10	192.168.182.129	TCP	http > iclpv-nlc [FIN, ACK] Seq=1511 Ack=384 Win=6912 Len=
11	0.131796	192.168.182.129	212.27.48.10	TCP	iclpv-nlc > http [ACK] Seq=384 Ack=1512 Win=96768 Len=0
12	0.135296	192.168.182.129	212.27.48.10	TCP	iclpv-nlc > http [FIN, ACK] Seq=384 Ack=1512 Win=96768 Len=
13	0.142579	212.27.48.10	192.168.182.129	TCP	http > iclpv-nlc [ACK] Seq=1512 Ack=385 Win=6912 Len=0
14	0.173829	192.168.182.129	192.168.182.1	TCP	iclpv-wsm > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=
15	0.182402	192.168.182.1	192.168.182.129	TCP	https > iclpv-wsm [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MS
16	0.182724	192.168.182.129	192.168.182.1	TCP	iclpv-wsm > https [ACK] Seq=1 Ack=1 Win=96768 Len=0
17	0.259992	192.168.182.129	192.168.182.1	SSL	Client Hello
18	0.266048	192.168.182.1	192.168.182.129	TCP	https > iclpv-wsm [ACK] Seq=1 Ack=367 Win=6912 Len=0
19	0.268301	192.168.182.1	192.168.182.129	TLSv1	Server Hello, Change Cipher Spec, Encrypted Handshake Mess

```
[Reassembled TCP Segments (1510 bytes): #8(1460), #9(50)]
Hypertext Transfer Protocol
  HTTP/1.1 302 Moved Temporarily\r\n
    Connection: close\r\n
    Cache-Control: no-cache, must-revalidate\r\n
    P3P: CP="IDC DSP COR ADM DEVi TAIi PSA PSD IVAi IVDi CONi HIS OUR IND CNT"\r\n
    [truncated] Location: https://192.168.182.1/intercept.php?res=notyet&uamip=192.168.182.1&uamport=3990&challenge=6595f6
    Content-Type: text/html; charset=UTF-8\r\n
```

Lorsqu'un équipement de consultation tente de se connecter sur Internet (www.free.fr dans la capture ci-dessus) :

- **1)** [trame 1] La requête DNS de l'équipement est reçue par le serveur DNS principal d'ALCASAR (unbound). Les tentatives de connexion vers d'autres serveurs DNS sont bloquées par le pare-feu interne. Cela permet de prévenir le contournement du DNS d'ALCASAR ainsi que les tunnels DNS.
- **2)** unbound résout le domaine localement s'il est dans son cache, sinon il génère sa propre requête vers les serveurs DNS Internet définis lors de l'installation d'ALCASAR. Les réponses sont retournées par unbound à l'équipement de consultation [trame 2].
- **3)** Quand une requête WEB (80 ou 443) est envoyée à l'@IP destination [trame 3] par l'équipement de consultation, elle est interceptée par « chilli » qui vérifie si un utilisateur est déjà « autorisé » sur cet équipement :
  - Si tel est le cas, « chilli » laisse transiter toutes les trames de l'équipement quel que soit le protocole. Le pare-feu dynamique prend alors le relai. Il oriente les flux de l'utilisateur (DNS, WEB, etc.) vers les chaînes de traitement associées à son profil (cf. fonction de filtrage).
  - **4)** Si tel n'est pas le cas, « chilli » simule une connexion WEB standard [trames 4 à 6] et répond à la requête de l'équipement [trames 7 à 9] par une trame HTTP de redirection de service *(« HTTP/1.0 302 Moved Temporarily »)* contenant l'URL de la page d'authentification définie par la primitive *« uamserver »* du fichier *« /etc/chilli.conf »* (« http://@ip_alcasar/intercept.php ») [cf. détail de la trame 9]. Cette session TCP se termine [trames 10 à 13]. Tant que l'utilisateur n'est pas authentifié, aucun flux réseau ne peut sortir de son équipement vers Internet.
  - **5) & 6)** Le navigateur initie alors une connexion avec le serveur WEB d'ALCASAR (Apache) afin de récupérer la page « intercept.php » [trame 14 et suivantes].
  - **7)** L'utilisateur renseigne les champs d'authentification (identifiant + mot de passe) qui sont envoyés à Apache pour être traités **8)**. Dans le cas de la capture présentée, ces flux sont chiffrés (HTTPS).
  - **9)** Apache retourne le résultat au navigateur afin que ce dernier redirige ces informations au processus « chilli » (port 3990[2]).
  - **10)** « chilli » les récupère afin de requêter le serveur radius **11) & 12) & 13) & 14) & 15)**. La communication entre « chilli » et « freeradius » exploite le protocole « radius ». Les paramètres de

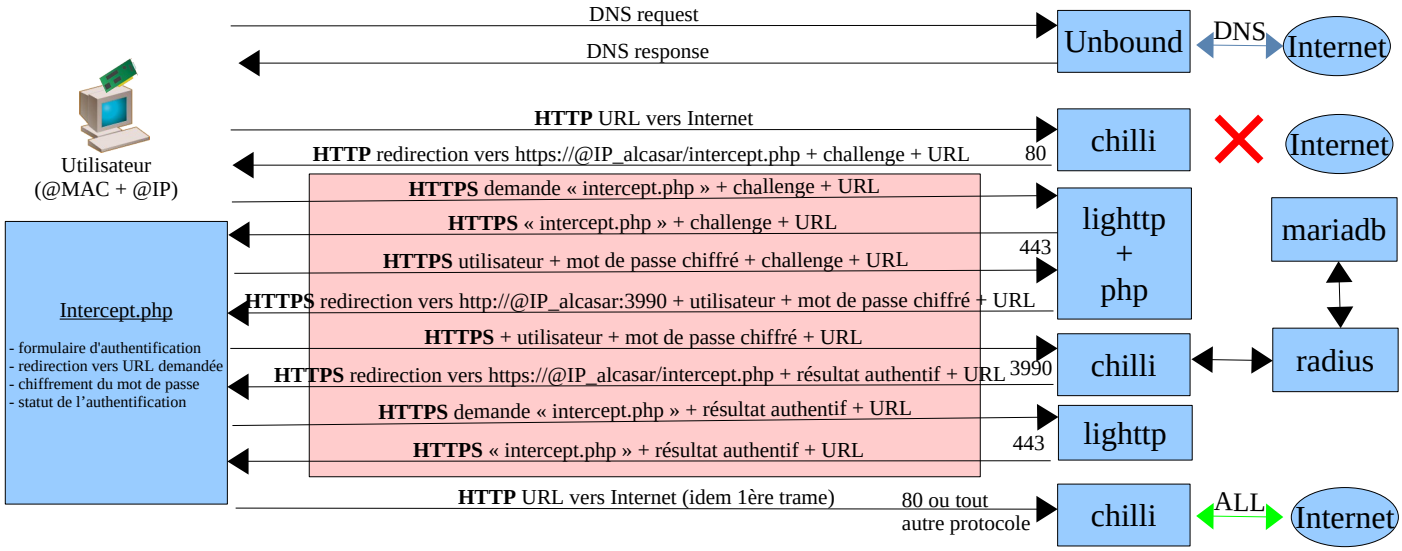
2 - « chilli » écoute sur un port défini par la primitive « uamport » de son fichier de configuration *« /etc/chilli.conf »* (3990 par défaut). Le format des requêtes envoyées sur ce port détermine l'action demandée (ex. « @IP:3990/prelogin » pour une demande de connexion, « @IP:3990/logout » pour une demande de déconnexion. La requête contient bien entendu l'ensemble des paramètres nécessaires au traitement de la demande (@MAC, challenge, identifiant, etc.).

cette communication sont définis à la fois dans le fichier *« /etc/raddb/client.conf »* et via les directives « *hs_radius* », « *hs_radius2* » *et* « *hs_radsecret* » du fichier *« /etc/chilli.conf »*.

- **16)** Le résultat de cette requête est retourné au navigateur afin d'être traité par les scripts javascript de la page « intercept.php » (échec ou réussite de la connexion).
- Dans son rôle de client radius, « chilli » gère les compteurs liés aux attributs d'autorisation de l'utilisateur (temps de connexion, débit, volume, etc.). Il les affiche sur la page de connexion de l'utilisateur (requêtes « ajax ») **17)**. Il les enregistre régulièrement pour la journalisation **18)**.

- Pour la déconnexion, les navigateurs Web génèrent une requête adéquate sur le port d'écoute de « chilli » (3990).

Cette phase d'interception peut être schématisée comme suit :



Cette double redirection (chilli → Apache → chilli) a été maintenue dans ALCASAR afin de rester conforme aux architectures d'authentification où les 3 services en charge de l'authentification/autorisation à savoir le client radius (« chilli »), le serveur radius et le service WEB peuvent être hébergés sur des serveurs différents.

Exception à l'authentification

« chilli » à la possibilité de laisser transiter des trames spécifiques vers Internet sans authentification préalable. Cette possibilité est exploitée dans ALCASAR pour permettre les mises à jour automatiques. Les paramètres « uamallowed » et « uamdomain » pointent vers deux fichiers contenant la liste des adresses IP (ou adresse IP de réseaux) ou des noms de domaine joignables sans authentification (*« /usr/local/etc/alcasar-uamallow »* et *« /usr/local/etc/alcasar-uamdomain »*).

4.1.3 - Action lors des connexions / déconnexions

« Chilli » possède la capacité d'appeler un programme externe lors du déclenchement d'un évènement interne (hook). ALCASAR exploite cette capacité sur les évènements suivants :

- « nouvel utilisateur authentifié » : appel du programme « alcasar-conup.sh » ;
- « déconnexion d'un utilisateur » : appel du programme « alcasar-condown.sh » ;
- « nouvelle @MAC détectée sur le LAN » : appel du programme « alcasar-macup.sh ».

Ces scripts permettent notamment de rediriger les flux des utilisateurs vers les processus de traitement adéquats en fonction de leurs profile de filtrage (filtrages DNS, URL, protocoles, etc.). À cet effet, plusieurs « ipset » ont été créés (cf. §6.2.2) afin que le parefeu puisse gérer de manière dynamique les @ip et @MAC des utilisateurs (cf. schéma des flux au §3). La commande « ipset -L -t ») permet de lister tous les « ipset » :

- ipset contenant les @IP des utilisateurs en fonction de leurs attributs : « not_filtered », « av », « av_bl », « av_wl » ;
- ipset contenant les @IP de la blacklist et de la whitelist : « bl_ip_blocked », « wl_ip_allowed » ;
- ipset contenant les protocoles réseau filtrés : « proto_0 », « proto_1 », « proto_2 », « proto_3 » ;

- ipset contenant les sites autorisés sans authentification : « site_direct ».

4.1.4 - Protection contre l'écoute

ALCASAR intègre 3 protections permettant d'éviter le vol d'identifiants par un pirate qui serait connecté sur réseau de consultation :

1. La première protection consiste à empêcher les techniques de MiTM LAN (ARP spoofing). Cela a été rendu possible en gérant le cache ARP de la carte réseau LAN (INTIF) de manière semi-statique. L'empoisonnement du cache ARP d'ALCASAR est ainsi rendu impossible.
2. La deuxième protection est assurée via l'activation du chiffrement des flux d'authentification HTTPS (via l'ACC ou via la commande « *alcasar-https.sh -on|-off* »). Cette activation d'HTTPS impose cependant de gérer les certificats afin que les navigateurs des utilisateurs acceptent cette connexion chiffrée. Cela implique :
   - soit d'installer un certificat officiel sur ALCASAR issu d'une autorité de certification reconnue (letsencrypt ou autre) ;
   - soit d'installer sur tous les navigateurs le certificat de l'autorité de certification (CA) ayant signé le certificat interne (certificat autosigné créé lors de l'installation ou regénéré via la commande « alcasar-CA.sh »).
3. Conscient que l'activation du HTTPS présente des contraintes fortes en termes de gestion de certificats, nous sommes en train de développer une solution exploitant un chiffrement des identifiants via un algorithme asymétrique associé à un bi-clés géré par ALCASAR. Le fonctionnement est le suivant : TODO

4.2 - Base de données des utilisateurs

Le SGBDR « mariaDB » est exploité pour gérer les bases de données d'ALCASAR. Deux bases de données ont été créées : la base « gammu » pour l'exploitation du module d'auto-inscription par SMS et la base « radius » pour la gestion des utilisateurs. Le modèle conceptuel de données (MCD) de cette base est entièrement compatible avec le service d'authentification Radius. Sa structure est mise en place lors de l'installation d'ALCASAR en exploitant un script SQL (« radius-db-vierge.sql »).

```
# Ajout d'une base radius vierge
mariadb -u$DB_USER -p$radiuspwd $DB_RADIUS < $DIR_CONF/radiusd-db-vierge.sql
```

Les Modèles Conceptuels de Données (MCD) de ces deux bases sont les suivants :

BASE DE DONNÉES RADIUS V3.x*
(gestion des utilisateurs)

Légende

En rouge : Clé Primaire. AI : Auto-Increment
** Clé unique * : Clé secondaire

Tables normalisées Radius

Tables complémentaires ALCASAR

Tables inexploitées actuellement

1. badusers
id int(10)
UserName * varchar(30)
Date * datetime
Reason varchar(200)
Admin varchar(30)

12. userinfo
id int(10) - AI
username * varchar(64)
name varchar(200)
mail varchar(200)
department * varchar(200)
workphone varchar(200)
homphone varchar(200)
mobile varchar(200)

2. mtotacct
mtotacctid bigint(21) - AI
username * varchar(64)
acctdate * date
connnum bigint(12)
conntotduration bigint(12)
connmaxduration bigint(12)
connminduration bigint(12)
inputoctets bigint(12)
outputoctets bigint(12)
nasipaddress * varchar(15)

10. totacct
totacctid bigint(21) - AI
username * varchar(64)
acctdate * date
connnum bigint(12)
conntotduration bigint(12)
connmaxduration bigint(12)
connminduration bigint(12)
inputoctets bigint(12)
outputoctets bigint(12)
nasipaddress * varchar(15)

4. radacct
radacctid bigint(21) - AI
acctsessionid * varchar(64)
acctuniqueid ** varchar(32)
username * varchar(64)
realm varchar(64)
nasipaddress * varchar(15)
nasportid varchar(32)
nasporttype varchar(32)
acctstarttime * datetime
acctupdatetime datetime
acctstoptime * datetime
acctinterval * int(12)
acctsessiontime * int(12)
acctauthentic varchar(32)
connectinfo_start varchar(128)
connectinfo_stop varchar(128)
acctinputoctets bigint(20)
acctoutputoctets bigint(20)
calledstationid varchar(50)
callingstationid varchar(50)
acctterminatecause varchar(32)
servicetype varchar(32)
framedprotocol varchar(32)
framedipaddress * varchar(15)
framedipv6address * varchar(45)
framedipv6prefix * varchar(45)
framedinterfaceid * varchar(45)
delegateipv6prefix * varchar(45)
class * varchar(64)

8. radpostauth
id int(11) - AI
username * varchar(64)
pass varchar(64)
reply varchar(32)
authdate timestamp(6)
class * varchar(64)

5. radcheck
id int(11) - AI
username * varchar(64)
attribute varchar(64)
op char(2)
value varchar(253)

9. radreply
id int(11) - AI
username * varchar(64)
attribute varchar(64)
op char(2)
value varchar(253)

11. radusergroup
id int(11) – AI
username * varchar(64)
groupname varchar(64)
priority int(11)

7. radgroupreply
id int(11) - AI
groupname * varchar(64)
attribute varchar(64)
op char(2)
value varchar(253)

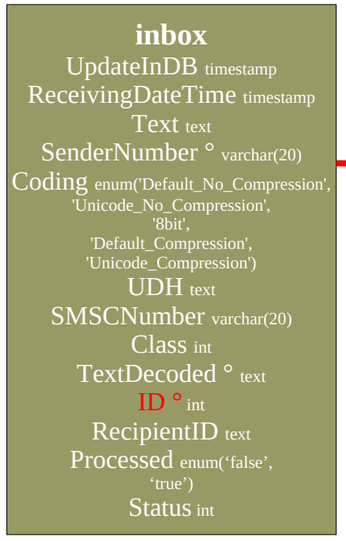
6. radgroupcheck
id int(11) - AI
groupname * varchar(64)
attribute varchar(64)
op char(2)
value varchar(253)

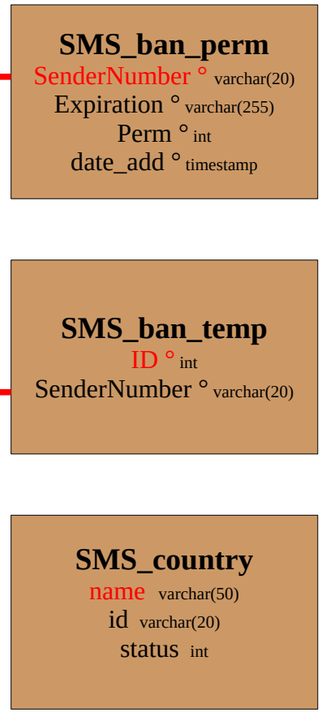
3. nas
id int(10) - AI
nasname * varchar(128)
shortname varchar(32)
type varchar(30)
ports int(5)
secret varchar(60)
community varchar(50)
description varchar(200)

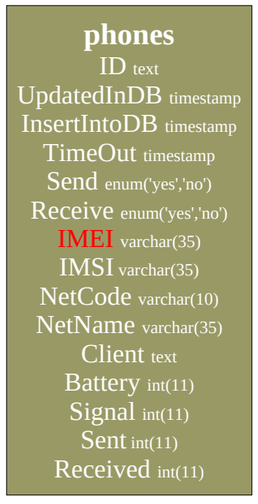
* dans les versions < 2.0 : la table « radusergroup » s'appelait « usergroup » et le champs « groupname » de la table « raddacct » n'existait pas
* à partir de la version 2.6 : le champs 'username' de la table 'userinfo' change de type pour être compatible avec les autres tables (bascule de varchar(30) en varchar(64))
* à partir de la version 3.7 : - dans la table ‘radacct’ : ajout de ‘acctupdatetime’, ‘acctinterval’. ‘framedipv6address’, ‘framedipv6prefix’, ‘delegateipv6prefix’ et ‘class’
  - dans la table ‘radacct’ : Suppression de ‘groupname’, ‘acctstartdelay’, ‘acctstopdelay’ et ‘xascendsessionsvrkey’ (a nécessité la modification des requêtes coova (cf. queries.conf))
  - dans la table ‘radauthpost’ : ajout de ‘class’
  - ajout de la table ‘nas’

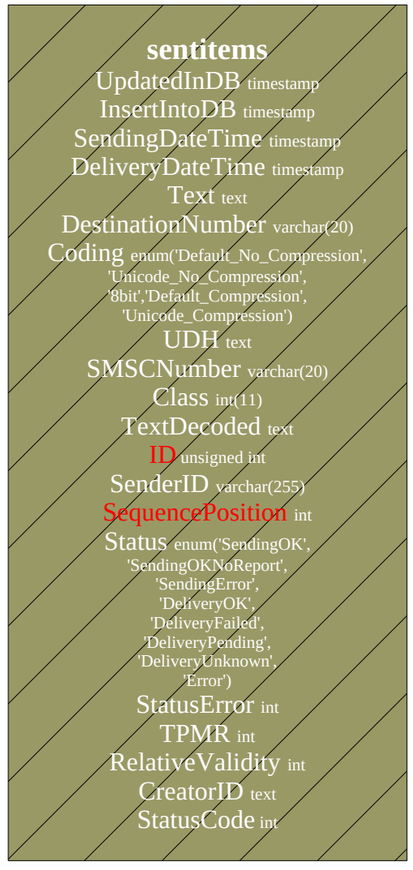
! l’attribut « id » de la table « radusergroup » n’est pas renseigné par coova → génère une erreur (Column « id » in SELECT is ambiguous). cf. query.conf.
! l’attribut « acctuniqueid » de la table « radacct » n’est pas renseigné par coova (il est donc remplacé par ‘’). Il ne peut donc pas être « unique ».

BASE DE DONNÉES GAMMU V1.39
(gestion des SMS)
Version du schéma : 17









Légende

En rouge : Index – Clé Primaire

Table créée par Gammu-smsd

Table ajoutée pour gérer les SMS

Table inexploitée par ALCASAR

° : Colonne exploitée

4.2.1 - Débogage

En mode texte

Utilisez les login/mot_de_passe de votre fichier *« /root/ALCASAR-password.txt »*

Accès à l'administration globale

```
mariadb -uroot -p$(grep '^db_root=' /root/ALCASAR-passwords.txt | cut -d'=' -f2-)
```

ex : pour afficher la liste des bases de données : « SHOW DATABASES ; »

« exit » pour quitter ; « help » pour voir les autres commandes.

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| gammu              |
| information_schema |
| mysql              |
| performance_schema |
| radius             |
+--------------------+
5 rows in set (
```

Accès aux différentes bases (ex : avec la base « radius »)

```
mariadb -uradius -p$(grep '^db_password=' /root/ALCASAR-passwords.txt | cut -d'=' -f2-) radius
```

ex : Pour afficher toutes les tables : « SHOW TABLES ; »

ex : Pour afficher les attributs d'une table : DESCRIBE <table_name>

ex : Pour afficher le contenu d'une table : « SELECT * FROM <table_name> ; »

```
+-------------------+
| Tables_in_radius  |
+-------------------+
| mtotacct          |
| radacct           |
| radcheck          |
| radgroupcheck     |
| radgroupreply     |
| radpostauth       |
| radreply          |
| radusergroup      |
| totacct           |
| userinfo          |
+-------------------+
10 rows in set (0.01 sec)
```

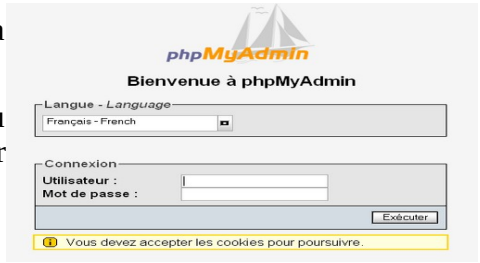
Journalisation des évènements

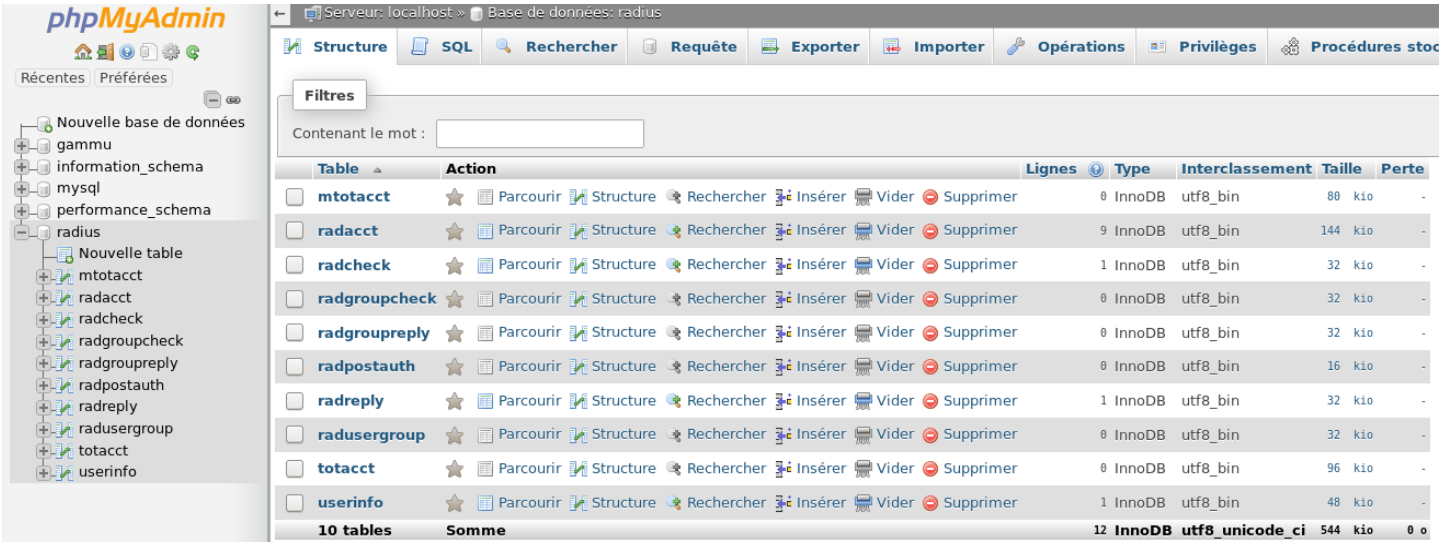
Il est possible d'activer la journalisation afin d'afficher toutes les requêtes reçues par le serveur « mariadb ». Pour cela, ajoutez les lignes « *general_log_file=/var/log/mysqld/mariadb.log* » et « *general_log=1* » dans la section [mysqld] du fichier de conf « */etc/my.cnf* ». *relancez le serveur : systemctl restart mariadb. Affichez les log :* tail -f /val/log/mysqld/mariadb.log

En mode graphique

Il est possible d'afficher de manière graphique (et pédagogique) le contenu des bases exploitées pas ALCASAR. Voici la procédure exploitant l'interface WEB « phpmyadmin » (à supprimer après avoir été exploité).

- Récupérez la dernière archive de « phpmyadmin » et copiez-la dans le répertoire /var/www/html/. Installez « unzip » (urpmi « unzip ») et décompressez « phpmyadmin » (unzip nom_de_l'archive). Renommez le nom du répertoire de l'archive en « phpmyadmin » (mv nom_répertoire phpmyadmin) ;
- connectez-vous à la base à partir de votre station de consultation à l'URL : « https://alcasar.lan/phpmyadmin » ;
- récupérez les mots de passe du compte d'administration (root) ou du compte gestionnaire de la base « radius » dans le fichier *« /root/ALCASAR-passwords.txt »* ;
- identifiez-vous sur le SGBD soit en « root » soit en « radius » ;
- Vous pouvez maintenant accéder aux bases (« gammu » ou « radius ») ainsi qu'aux contenus des tables.





Quand vous avez terminé avec « phpmyadmin », désinstallez-le (rm -rf phpmyadmin »).

Exemple d'actions (base « radius »)

```
SET NAMES utf8
```

Ajout d'un groupe d'utilisateurs (group-test)

Vérification de doublon

```
SELECT DISTINCT groupname FROM radusergroup WHERE username = 'group-test'
SELECT attribute,value ,op FROM radgroupcheck WHERE groupname = 'group-test'
SELECT attribute,value ,op FROM radgroupreply WHERE groupname = 'group-test'
SELECT username FROM radusergroup WHERE groupname = 'group-test' ORDER BY username
```

Insertion

```
INSERT INTO radusergroup (username,groupname) VALUES ('group-test','group-test')
```

Ajout d'attributs

```
INSERT INTO radgroupcheck (attribute,value,groupname ,op) VALUES ('Simultaneous-Use','2','group-test' ,':=')
INSERT INTO radgroupcheck (attribute,value,groupname ,op) VALUES ('Login-Time','Mo0800-1900,Tu0800-1900,We0800-1900,Th0800-1900,Fr0800-1900','group-test' ,':=')
```

Ajout d'un utilisateur ("rexy") + affectation à un groupe + ajout d'attribut

Vérification de doublon

```
SELECT attribute,value ,op FROM radcheck WHERE username = 'rexy'
SELECT attribute,value ,op FROM radreply WHERE username = 'rexy'
SELECT DISTINCT groupname FROM radusergroup WHERE username = 'rexy'
SELECT username FROM userinfo WHERE username = 'rexy'
```

Insertion

```
INSERT INTO radcheck (attribute,value,username ,op) VALUES ('Crypt-Password','$5$fdzbkw8g$e21ZDZ/OHV2B6iqseee5B/XJ5vzOBk5j/HjQ7eK.ayB','rexy' ,':=')
INSERT INTO userinfo (username,name,mail,department,homephone,workphone,mobile) VALUES ('rexy','','','','','')
```

Affectation du groupe

```
INSERT INTO radusergroup (username,groupname) VALUES ('rexy','group-test')
```

Ajout d'un attribut particulier

```
INSERT INTO radcheck (attribute,value,username ,op) VALUES ('Expiration','27 November 2020','rexy' ,':=')
```

Suppression d'un utilisateur ("rexy")

```
DELETE FROM radreply WHERE username = 'rexy'
DELETE FROM radcheck WHERE username = 'rexy'
DELETE FROM radusergroup WHERE username = 'rexy'
DELETE FROM userinfo WHERE username = 'rexy'
```

Ajout d'une adresse MAC (équipement de confiance)

Vérification de doublon

```
SELECT attribute,value ,op FROM radcheck WHERE username = '08-00-27-39-30-92'
SELECT attribute,value ,op FROM radreply WHERE username = '08-00-27-39-30-92'
SELECT DISTINCT groupname FROM radusergroup WHERE username = '08-00-27-39-30-92'
SELECT * FROM userinfo WHERE username = '08-00-27-39-30-92'
```

Insertion

```
INSERT INTO radcheck (attribute,value,username ,op) VALUES ('Crypt-Password','$5$35l1gtzc$nKevQ39XlUBclsRu98zSCEhQfyoWV0FtugsVFPWlNS7','08-00-27-39-30-92' ,':=')
INSERT INTO userinfo (username,name,mail,department,homephone,workphone,mobile) VALUES ('08-00-27-39-30-92','','','','','')
```

Ajout d'attributs

```
INSERT INTO radreply (attribute,value,username ,op) VALUES ('Alcasar-Filter','3','08-00-27-39-30-92' ,'=')
INSERT INTO radreply (attribute,value,username ,op) VALUES ('Alcasar-Protocols-Filter','2','08-00-27-39-30-92' ,'=')
INSERT INTO radreply (attribute,value,username ,op) VALUES ('Alcasar-Status-Page-Must-Stay-Open','1','08-00-27-39-30-92' ,'=')
```

4.3 - Le serveur Radius

Le service « freeradius » (radiusd) est utilisé comme unité d'authentification, d'autorisation et de journalisation (Authentication Authorization Accounting – AAA). Le schéma fonctionnel est présenté en P3.

L'authentification exploite par défaut la base de données locale (mariadb). Un module LDAP additionnel a été intégré afin de pouvoir valider le couple login/MDP avec celui d'un annuaire externe compatible LDAP (Active Directory©, OpenLDAP, etc.).

L'autorisation et la journalisation exploitent uniquement la base de données local. Cela est lié au fait que les nombreux attributs affectés aux utilisateurs d'ALCASAR n'existent pas dans un annuaire A.D. ou un annuaire externe.

Les fichiers de configuration du serveur sont dans le répertoire *« /etc/raddb ».* La configuration de ce serveur est définie dans le fichier *« radiusd.conf »*. Le fichier *« client.conf »* contient les paramètres du seul client radius configuré par défaut (le daemon « chilli »). Deux fichiers de configuration spécifiques pour ALCASAR sont situés dans le répertoire « sites-available » (« alcasar » et « alcasar-with-ldap »). En fonction de la configuration, l'un ou l'autre est lié symboliquement au fichier « alcasar » situé dans le répertoire « sites-enabled ». Les modules radius (rlm : Radius Loadable Modules) exploités par ALCASAR sont situés dans le répertoire « mods-enabled ». Ce sont des liens symboliques pointant vers des fichiers du répertoire « mods-available ». Les modules suivants sont exploités pour ALCASAR :

- « sql » : défini les paramètres de connexion à la base de données des utilisateurs (mariadb) ;
- « pap » : permet de valider le mot de passe chiffré de l'utilisateur (Password Authentication Protocol) ;
- « attr_filter » : permet de filtrer (mise en forme) certains attributs avant de les traiter ;
- « expr » : utiliser pour effectuer des calculs numériques (*, /, +, sha256, etc.) et pour convertir des caractères (ex : « toupper » et « tolower ») ;
- « expiration » : permet de gérer la date d'expiration d'un compte (attribut : « Expiration ») ;
- « logintime » : permet de gérer le « temps de session max » (attribut : « Session-Timeout ») ;
- « sqlcounter » : permet de calculer les notions de temps ou de volume cumulé. Ils sont définis plus loin dans ce chapitre (Ch. 4.3.3) ;
- « ldap » : permet la connexion vers un annuaire externe (chargé uniquement si configuré)

4.3.1 - Débogage

Il est possible de tester l'authentification et l'autorisation d'un utilisateur sur le serveur radius en ligne de commande :

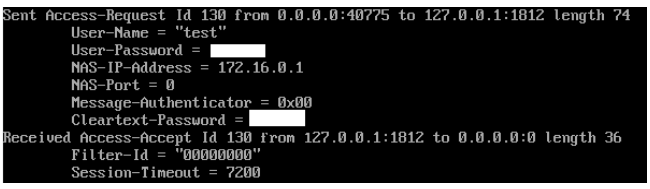
echo "User-Name = test,User-Password=test" | radclient -x localhost:1812 auth $(grep '^secret_radius=' /root/ALCASAR-passwords.txt | cut -d'=' -f2-)

ou

radtest 'USERNAME' 'PASSWORD' 127.0.0.1 0 $(grep '^secret_radius=' /root/ALCASAR-passwords.txt | cut -d'=' -f2-)

```
# radtest 'test' 'test' 127.0.0.1 0 $(grep '^secret_radius=' /root/ALCASAR-passwords.txt | cut -d'=' -f2-)
Sent Access-Request Id 135 from 0.0.0.0:39859 to 127.0.0.1:1812 length 74
        User-Name = "test"
        User-Password = "test"
        Cleartext-Password = "test"
        Received Access-Accept Id 129 from 127.0.0.1:1812 to 127.0.0.1:39123 length 38
        Alcasar-Filter = HAVP
        Alcasar-Protocols-Filter = Web
        Session-Timeout = 555222
```

Il est possible d'analyser le fonctionnement du serveur radius. Pour cela, arrêtez le DAEMON (systemctl stop radiusd) et relancez-le en mode « débug » *(radiusd -X)*. Vous pouvez alors analyser les requêtes SQL lorsque des



utilisateurs se connectent, se déconnectent, échouent à la connexion, etc.

4.3.2 - Les attributs Radius exploités par ALCASAR

ALCASAR exploite plusieurs attributs standards radius (Crypt_Password, Simultaneous-Use, Wispr-Redirection-URL, Login-Time, Expiration, etc.). De plus, pour ses besoins spécifiques, les 6 attributs particuliers suivants ont été créés (cf. */etc/raddb/conf/radiusd/dictionary.alcasar*) :

- Alcasar-Filter (integer) : Niveau de filtrage DNS/IP/WEB
  - 1=None (sans filtrage),
  - 2=HAVP (filtrage antimalware → actuellement inactif faute d'intérêt avec les flux https),
  - 3=BL (filtrage de type Blacklist),
  - 4=WL (filtrage de type Whitelist)
- Alcasar-Protocols-Filter (integer) : Niveau de filtrage protocolaire (cf. 6.2.1)
  - 1=None (sans filtrage) ;
  - 2=Web (accès uniquement à http et https) ;
  - 3=Commons (accès aux ports 20,21,22,25,80,110,115,143,443,465,587,993,995,2525) ;
  - 4=Custom (accès aux protocoles définis dans l'ACC).
- Alcasar-Imputability-Warning (integer) :
  - 1=Yes (quand un admin a consulté des fichiers de traçabilité. Les utilisateurs en seront informés lors de leur prochaine connexion).
- Alcasar-Status-Page-Must-Stay-Open (integer) : L'utilisateur doit-il laisser sa page de statut ouverte ?
  - 1=Yes (il reste connecté tant que sa page de statut est active. Vérifié toutes les 3' par le Watchdog
  - 2=No (il sera déconnecté à minuit par le watchdog).
- Alcasar-Expire-After (integer) : Temps de connexion autorisé après le premier login (en seconde)
- Alcasar-Reconnect-Timeout (integer) : Usage futur

4.3.3 - Compteurs SQL exploités par ALCASAR

Ces compteurs permettent de calculer des valeurs cumulatives. Ils sont lancés au moment de l'authentification d'un utilisateur pour autoriser sa connexion.

Déclaration des compteurs (section « authorize » de *« /etc/raddb/sites-enabled/alcasar »*)

```
authorize {
        sql
        noresetcounter
        dailycounter
        monthlycounter
        expiration
        logintime
        pap
}
```

Définition des compteurs SQL (fichier *« /etc/raddb/mods-enabled/sqlcounter »*).

Temps de connexion déjà effectué sur la journée (attribut « Max-Daily-Session »)

```
sqlcounter dailycounter {
      sql_module_instance = sql
      counter_name = Daily-Session-Time
      check_name = Max-Daily-Session
      reply_name = Session-Timeout
      key = User-Name
      reset = daily
      query = "SELECT IFNULL((SELECT SUM(acctsessiontime - GREATEST((%%b - UNIX_TIMESTAMP(acctstarttime)), 0)) FROM radacct WHERE username='%{${key}}' AND
UNIX_TIMESTAMP(acctstarttime) + acctsessiontime > '%%b'
), 0)"
}
```

Temps de connexion effectué sur le mois en cours (« monthlycounter » = attribut « Max-Monthly-Session »)

```
sqlcounter monthlycounter {
```

```
	sql_module_instance = sql
	counter_name = Monthly-Session-Time
	check_name = Max-Monthly-Session
	reply_name = Session-Timeout
	key = User-Name
	reset = monthly
	query = "SELECT IFNULL((SELECT SUM(acctsessiontime - GREATEST((%%b - UNIX_TIMESTAMP(acctstarttime)), 0)) FROM radacct WHERE username='%{${key}}' AND
UNIX_TIMESTAMP(acctstarttime) + acctsessiontime > '%%b'
), 0)"
}
```

Temps autorisé cumulé après le premier login » (attribut « Max-All-Session »)

```
sqlcounter noresetcounter {
	sql_module_instance = sql
	counter_name = Max-All-Session-Time
	check_name = Max-All-Session
	key = User-Name
	reset = never
	query = "SELECT IFNULL(SUM(AcctSessionTime), 0) FROM radacct WHERE username='%{${key}}'"
}
```

Temps autorisé pour une session (attribut « Alcasar-Expire-after »)

```
sqlcounter expire_on_login {
	sql_module_instance = sql
	counter_name = Alcasar-Expire-After-Initial-Login
	check_name = Alcasar-Expire-After
	key = User-Name
	reset = never
	query = "SELECT IFNULL((SELECT TIME_TO_SEC(TIMEDIFF(NOW(), acctstarttime)) FROM radacct WHERE username='%{${key}}' ORDER BY acctstarttime LIMIT 1), 0)"
}
```

Volume de données autorisé pour une journée (attribut « CoovaChilli-Max-Total-Octets-Daily »)

```
sqlcounter counterCoovaChilliMaxTotalOctetsDaily {
	sql_module_instance = sql
	counter_name = CoovaChilli-Max-Total-Octets-Daily
	check_name = CoovaChilli-Max-Total-Octets-Daily
	counter_type = data
	reply_name = CoovaChilli-Max-Total-Octets
	key = User-Name
	reset = daily
	query = "SELECT IFNULL((SUM(AcctInputOctets + AcctOutputOctets)), 0) FROM radacct WHERE username='%{${key}}' AND UNIX_TIMESTAMP(AcctStartTime) +
AcctSessionTime > '%%b'"
}
```

Volume de données autorisé pour un mois (attribut « CoovaChilli-Max-Total-Octets-Monthly »)

```
sqlcounter counterCoovaChilliMaxTotalOctetsMonthly {
	sql_module_instance = sql
	counter_name = CoovaChilli-Max-Total-Octets-Monthly
	check_name = CoovaChilli-Max-Total-Octets-Monthly
	counter_type = data
	reply_name = CoovaChilli-Max-Total-Octets
	key = User-Name
	reset = monthly
	query = "SELECT IFNULL((SUM(AcctInputOctets + AcctOutputOctets)),0) FROM radacct WHERE username='%{${key}}' AND UNIX_TIMESTAMP(AcctStartTime) +
AcctSessionTime > '%%b'"
}
```

Volume max de données autorisé (attribut « CoovaChilli-Max-Total-Octets »)

```
sqlcounter counterCoovaChilliMaxAllTotalOctets {
	sql_module_instance = sql
	counter_name = CoovaChilli-Max-All-Total-Octets
	check_name = CoovaChilli-Max-Total-Octets
	counter_type = data
	reply_name = CoovaChilli-Max-Total-Octets
	key = User-Name
	reset = never
	query = "SELECT IFNULL((SUM(AcctInputOctets + AcctOutputOctets)),0) FROM radacct WHERE username='%{${key}}'"
}
```

4.3.4 - Module A.D./LDAP externe

Freeradius peut interroger une base externe via le protocole LDAP. Les paramètres de connexion sont renseignés dans le fichier de configuration central d'ALCASAR (/usr/local/etc/alcasar.conf). Le script « alcasar-ldap.sh » se charge de configurer les modules LDAP de freeradius (*« /etc/raddb/mods-available/ldap-alcasar »).* Un lien symbolique lie le fichier *« /etc/raddb/mods-enabled/ldap »*) vers ce fichier quand « ldap » est activé. Dans le même ordre d'idée, un lien symbolique lie le fichier *« /etc/raddb/sites-available/alcasar-with-ldap »* vers le fichier *« /etc/raddb/sites-enabled/alcasar »*. Les paramètres du fichier sont les suivants :

Paramètres	Définition	Remarques
server	Nom du serveur LDAP (server = "ldap.example.com" ou server = "@IP"	Le port de connexion par défaut est 389. Pour le changer : @ serveur:port
basedn	Base de recherche des utilisateurs à authentifier	Voir l'exemple ci-dessous dans le cas d'Active Directory.
filter	Recherche de l'identifiant ou attribut pour l'authentification	Pour un ldap standard : filter = "(uid=%{Stripped-User-Name:-%{User-Name}})" Pour Active Directory : filter = "(samAccountName=%{Stripped-User-Name:-%{User-Name}})"
base_filter	Filtre de recherche ldap complémentaire	Exemples : – par défaut, vide – base_filter="(objectclass=radiusprofile)" – base_filter="(memberof=groupe_alcasar)"
identity	Compte possédant des droits en lecture sur l'annuaire.	Vide = connexion anonyme (LDAP) Obligatoire pour Active Directory (sur le serveur AD, créer un compte standard avec délégation qui sera utilisé par ALCASAR pour l'interroger l'annuaire à distance).
password	Mot de passe associé au compte avec des droits de lecture sur l'annuaire ldap.	Vide = connexion anonyme (LDAP). Obligatoire pour Active Directory.

Par rapport à l'exemple d'annuaire présenté dans le document d'exploitation, les paramètres de ce fichier seraient les suivants :

basedn = "cn=Users,dc=lab-ad,dc=lan"

filter = "(samAccountName=%{Stripped-User-Name:-%{User-Name}})"

identity= "cn=superman,dc=lab-ad,dc=lan"

password = "*******"

Il est possible d'analyser la connexion avec le serveur d'annuaire externe à partir du système ALCASAR après avoir installé le paquetage « openldap-clients » (*urpmi openldap-clients*).

La commande « *alcasar-ldap.sh --test* » permet d'afficher la liste des utilisateurs de l'annuaire distant. Un mode verbeux permet de déboguer via la commande « *alcasar-ldap.sh –test -d* ». Ce script lance la commande « *ldapsearch -vWx -H"ldap://@ip_A.D" -b"cn=Users,dc=lab-ad,dc=lan" -D"superman@lab-ad.lan"*».

Le groupe par défaut affecté aux utilisateurs authentifiés est défini par une variable « default_user_profile » définie dans le fichier « */etc/raddb/mods-config/sql/main/mysql/queries.conf »*. L'idée est de laisser les utilisateurs authentifiés par un annuaire externe dans ce groupe (il faut le créer dans l'ACC) et d'affecter les autres utilisateurs (ceux qui sont gérés par ALCASAR) dans d'autres groupes.

Il est aussi possible d'affecter des attributs ALCASAR à un seul utilisateur authentifié A.D/LDAP. Pour cela il faut créer un utilisateur ALCASAR ayant le même nom que celui qui est dans l'annuaire externe. Exemple d'utilisation : tous les élèves d'une école sont gérés dans un annuaire. Il est possible de limiter la bande passante ou les créneaux de connexion pour l'élève ayant abusé de téléchargements. Il suffit de créer un compte sur ALCASAR avec le nom de cet élève et de lui affecter des attributs particuliers.

5 - Fonction « traçabilité et imputabilité »

Plusieurs systèmes de traçabilité-imputabilité ont été évalués afin de définir celui qui serait exploité dans ALCASAR. Le tableau suivant résume le résultat de cette évaluation :

	NAGIOS	MRTG	CACTI	MUNIN	NETFLOW
Licence GPL					
Compatible Mageia					
Charge supplémentaire sur le système					
Espace de stockage nécessaire					
Facilité d'utilisation					
Interface d'administration					
Représentation des données sous forme de graphes					
Accessibilité aux données antérieures					
Contenu des fichiers journaux pour l'imputabilité					
Affichage de la charge par ports					

Satisfait : Convenable : Non satisfait :

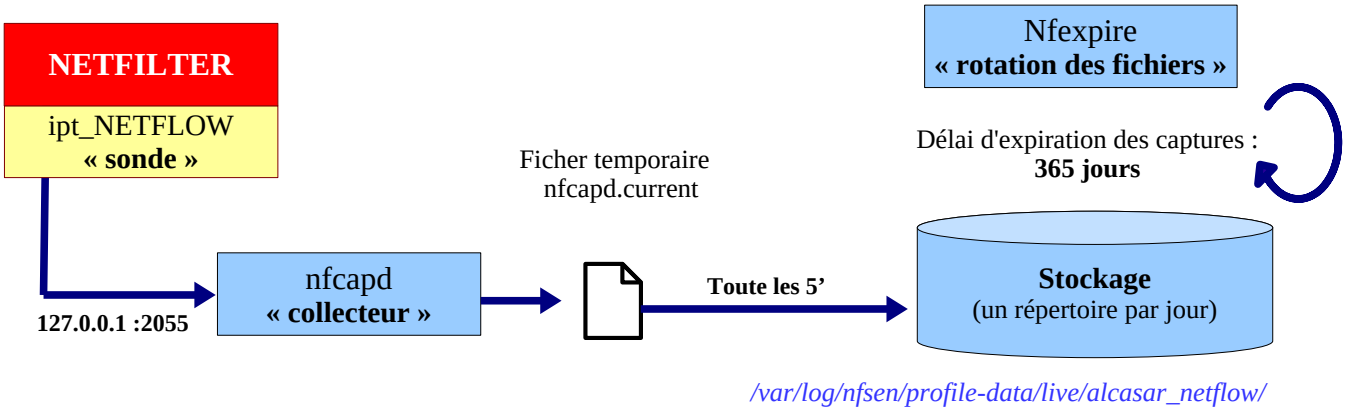
Après cette analyse, il a été décidé d'exploiter l'architecture « Netflow » (RFC3954). Sur ALCASAR, nous avons mis en œuvre cette architecture via le module noyau « ipt_netflow » qui génère les flux et le projet « nfdump » qui les collecte et les gère.

5.1 - Journalisation principale

La traçabilité des connexions est gérée par deux canaux de journalisation d'évènements.

1. Le canal général concerne tous les flux des utilisateurs à l'exception des flux HTTP des utilisateurs dont le proxy E2guardian a filtré les URLs (cf. 2e canal). Les traces sont générées par une sonde NetFlow qui est spécifiquement compilé pour le noyau Linux exploité par ALCASAR (projet « ipt_netflow »). Les flux netflow générés par cette sonde sont manipulés (collecte, interprétation, suppression, etc.) par les outils du projet libre « nfdump ».

Le schéma de fonctionnement de cette sonde est le suivant :



Les flux NetFlow sont générés par la sonde noyau (ipt_NETFLOW) via des règles du parefeu Netfilter (cible « -j Netflow »). Ces flux sont envoyés sur la socket 127.0.0.1:2055 où ils sont récupérés par le collecteur nfcapd qui génère un fichier toutes les 5' dans les répertoires « /var/log/nfsen/profile_data/live/alcasar_netflow/année/mois/jour/ ».

La commande « nfexpire » est lancée par « crontab » tous les jours à minuit afin de supprimer les fichiers de plus d'un an :

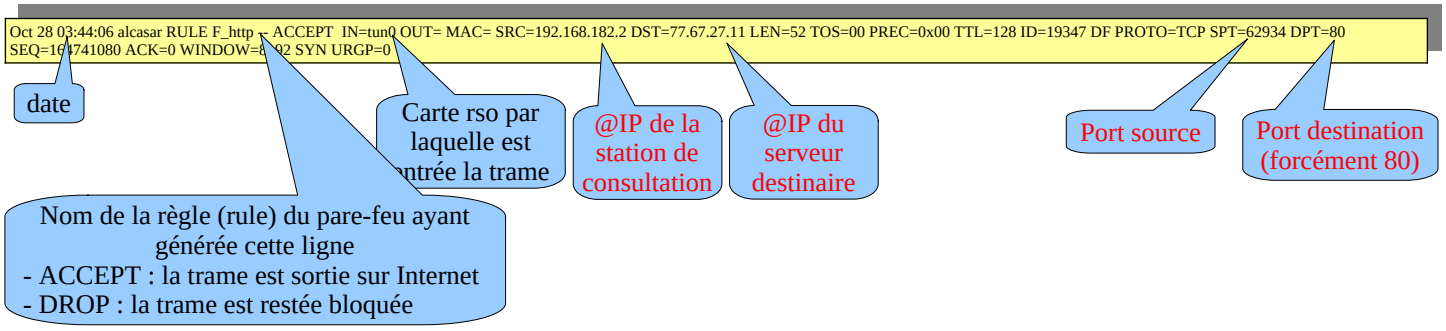
```
Remove netflow files older than one year (daily --> see "cron.daily")
@daily root /usr/bin/nfexpire -e /var/log/nfsen/profiles-data/live/alcasar_netflow -t 365d
```

Chaque semaine, une archive est constituée dans le répertoire « */var/SAVE* » sous le nom « traceability-ALL-<date><heure>.tar.gz » (cf. §5.3).

Les fichiers Netflow ne sont pas directement lisibles. Pour cela, il faut exploiter un **interpréteur Netflow** (ex. : « Nfdump ») comme suit : « nfdump -R <fichier_au_format_netflow> -o extended -a ». Une fois interprétée, chaque ligne est composée des champs suivants :

	Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	pps	bps	Bpp	Flows
•	2020-05-19 23:03:02.978	0.000	TCP	192.168.182.24:53955	→	216.239.38.120:443	....S.	0	1	60	0	0	60	1
•	2020-05-19 23:15:10.898	133.038	TCP	192.168.182.24:49570	→	216.239.38.120:5228	....S.	0	5	300	0	18	60	3
•	2020-07-04 14:45:29.870	0.000	TCP	192.168.182.7:48399	→	104.85.39.74:80	......S.	0	1	60	0	0	60	1

2. Le deuxième canal contient les traces des <u>flux HTTP des utilisateurs possédant un attribut de filtrage</u> (blacklist ou whitelist). Ces traces sont générées par une règle « Ulog » du pare-feu sur le flux transitant dans le système de filtrage D'ALCASAR (E²Guardian). Ces traces sont écrites dans le fichier « /var/log/firewall/traceability.log ». Chaque semaine, ce fichier est copié dans l'archive de traçabilité sous le nom « traceability-HTTP-<date><heure>.tar.gz » (cf. §5.3) avant d'être purgé. Dans ce fichier chaque ligne est composée des champs principaux suivants :



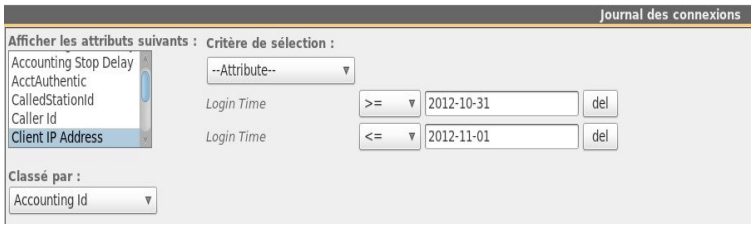
5.2 - journalisation accessoire

- Un canal Ulog génère les fichiers « /var/log/firewall/ssh.log » liés aux flux d'administration à distance via le protocole ssh.
- Un canal Ulog génère les fichiers « /var/log/firewall/ext-access.log » liés aux tentatives de connexions depuis Internet (fonction « bastion »). Pour ce canal, une protection est mise en place afin de ne pas charger trop le système en cas d'attaque par saturation (flooding).
- Le proxy de filtrage d'URL « E²Gurdian » génère des logs dans le répertoire « */var/log/e2guardian* » sous le nom :« *access.log* ». Ils présentent les URL ayant été bloquées.
- Le détecteur d'intrusion (IDS) « fail2ban » génère des logs dans « /var/log/fail2ban ».

5.3 - Constitution de l'archive de traçabilité

Tous les lundis à 5h35, le gestionnaire de tâche « cron » lance le script « alcasar-archive.sh ». Ce script crée un fichier contenant une archive pour chaque canal de journalisation (cf. §5.1) et la base des utilisateurs. Il copie ce fichier sous le nom « traceability-<date>-<heure>.tar.gz » dans le répertoire « /var/Save/archive/ » afin d'être visible dans l'interface de gestion (ACC).

Pour imputer chaque trame, il faut extraire à partir du fichier de la base de données « radius.sql » (de la même semaine) le nom de l'utilisateur connecté sur la station de consultation possédant l'adresse IP source. Cette dernière information peut être récupérée directement à partir de l'interface graphique d'ALCASAR (menu « statistique » + « connexions »). Exemple pour chercher les utilisateurs connectés dans la journée du 31/10/2012 :



6 - Fonction « filtrage »

L'architecture d'ALCASAR rend le contournement du filtrage très compliqué. Celui-ci est toujours possible par l'ouverture d'un tunnel (VPN) à destination d'un équipement maitrisé situé sur Internet dont l'internaute connait l'adresse IP. Pour fonctionner, ce tunnel doit faire transiter l'ensemble des protocoles de la station de consultation (dont le DNS).

Quoi qu'il en soit, ce type de tunnel ne permet pas de contourner l'authentification. Ainsi, ALCASAR trace et impute les trames de ce tunnel. En cas de problème, et si l'enquête détermine que la sortie du tunnel est impliquée, le portail pourra être sollicité pour déterminer quel utilisateur a créé ce tunnel.

Dans le cas de la WhiteList, ce contournement par VPN n'est pas possible. En effet, ALCASAR tient à jour de manière dynamique les @IP autorisées en fonction des appels DNS préalablement réalisés par l'internaute.

6.1 - Filtrage de protocoles réseau

Cette couche est gérée à l'aide du pare-feu intégré (NetFilter).
Le fichier de configuration principal qui conditionne le fonctionnement d'ALCASAR est *« /usr/local/bin/alcasar-iptables.sh »*. Il est déconseillé de le modifier afin d'éviter des effets de bords sur le fonctionnement.
Toutefois, des règles de pare-feu peuvent être surchargées pour autoriser certaines situations particulières (comme l'accès SSH à un équipement de votre LAN depuis Internet par exemple). Les règles pouvant « surcharger » les règles de base sont définies dans le fichier *« /usr/local/etc/alcasar-iptables-local.sh »* qui est appelé par le fichier de configuration principal du pare-feu. Ce fichier comporte plusieurs ensembles de règles d'exemple qui correspondent à des configurations que des utilisateurs d'ALCASAR ont eu besoin. Il est possible d'activer ces ensembles de règles en les adaptant et en les dé-commentant.

Pour forcer tous les utilisateurs à passer par le service DNS du portail, le pare-feu effectue une redirection du port 53 vers l'@IP locale. Cela permet de couper court aux éventuels tunnels DNS (sur le port 53 uniquement).

6.2 - Filtrage de noms de domaines, d'URLs et d'adresses IP

Ce filtrage s'appuie sur l'excellente liste de l'Université de Toulouse qui est organisée en répertoires. Chaque répertoire porte le nom d'une catégorie (adulte, secte, shopping, etc.). Chaque répertoire peut contenir 1 à 3 fichiers contenant la liste des « noms de domaine », des « URLs » et des « @IP » de cette catégorie. Un quatrième fichier permet de savoir si la catégorie est « noire » (blacklist) ou blanche (whitelist) ou les deux. ALCASAR traite cette liste afin de l'exploiter selon 3 techniques différentes :

1. filtrage de noms de domaine : ALCASAR s'appuie sur ses serveurs de DNS internes (« unbound ») ;
2. filtrage d'URLs : ALCASAR s'appuie sur le proxy HTTP/HTTPS « E²Guardian » ;
3. filtrage d'adresses IP : C'est le pare-feu d'ALCASAR qui traite les @IP de la liste.

6.2.1 - Filtrage par utilisateur/groupe

Quand un utilisateur réussi sont processus d'authentification, le daemon 'chilli' lance le script « /usr/local/bin/alcasar-conup.sh ». Ce script récupère l'ensemble des attributs de l'utilisateur. En fonction des attributs de filtrage « Alcasar-Filter » et « Alcasar-Protocols-Filter », le script positionne l'adresse IP de l'utilisateur dans l'IPSET correspondant à son niveau de filtrage « IP/DNS/WEB » ou « protocolaire ».

Quand l'utilisateur se déconnecte, le daemon 'chilli' lance le script « /usr/local/bin/alcasar-condown.sh » qui retire l'adresse IP de l'utilisateur de l'IPSET correspondant.

Les 4 IP_SET suivants ont été créés pour gérer les différentes possibilités de filtrage « IP/DNS/WEB » :

- IPSET=« not_filtered » pour les utilisateurs sans filtrage (Alcasar-Filter=1) ;
- IPSET=« av » pour les utilisateurs filtrés avec un antimalware → réservé pour un usage futur (Alcasar-Filter=2) ;

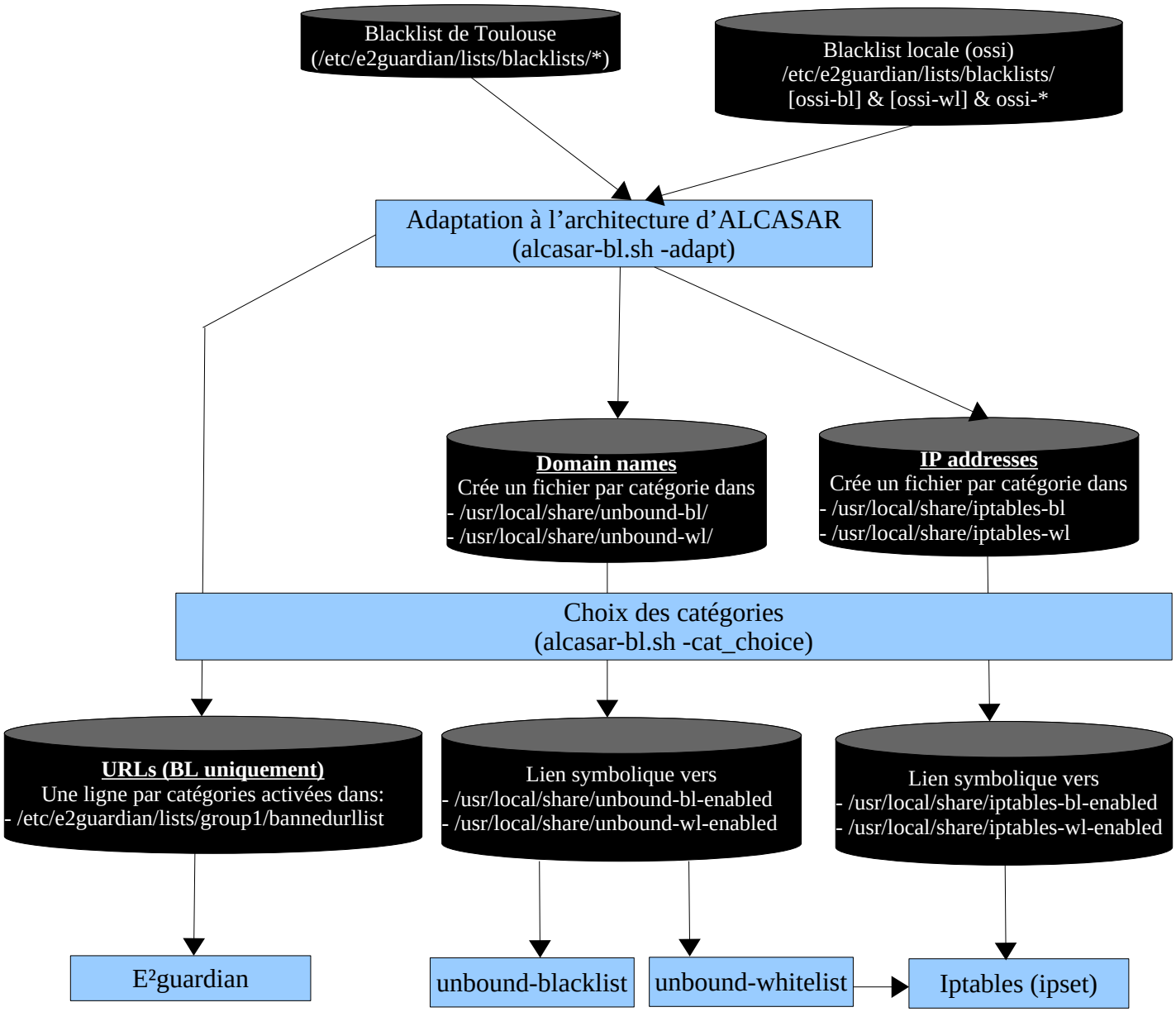
- IPSET=« av_bl » pour les utilisateurs filtrés par la liste noire (Alcasar-Filter=3) ;
- IPSET= « av_wl » pour les utilisateurs filtrés par la liste blanche (Alcasar-Filter=4)

Les 4 IP_SET suivants ont été créés pour gérer les différentes possibilités de filtrage de protocoles :

- IPSET=« proto_0 » sans filtrage (Alcasar-Protocols-Filter = 1) ;
- IPSET=« proto_1 » protocoles http et https (Alcasar-Protocols-Filter = 2) ;
- IPSET=« proto_2 » protocoles 20,21,22,25,80,110,115,143,443,465,587,993,995,2525 (Alcasar-Protocols-Filter = 3) ;
- IPSET= « proto_3 » protocoles « custom » définis dans l'ACC (Alcasar-Protocols-Filter = 4).

6.2.2 - Traitement de la liste de Toulouse

Afin de permettre cette triple exploitation, le script « alcasar-bl.sh » effectue le traitement suivant lors de l'installation, de la mise à jour ou du choix des catégories de la Blacklist :



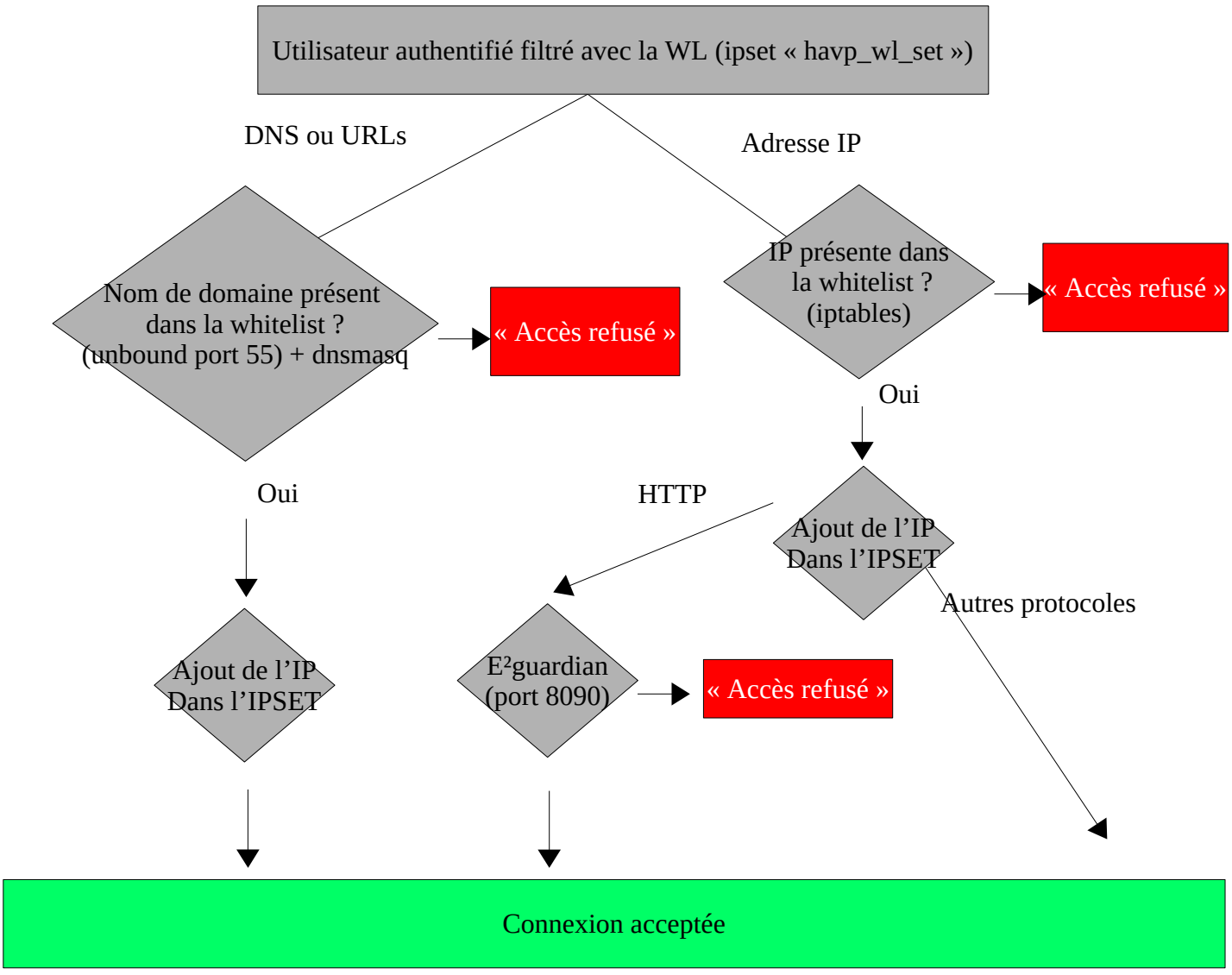
6.2.3 - Double filtrage de la WhiteList (WL)

Un utilisateur filtré avec la WL ne peut consulter que des sites ou des IP définis préalablement. La WL se décompose en plusieurs catégories de noms de domaine pouvant être sélectionnées et mises à jour dans l'ACC. La liste complète des noms de catégorie est située dans *« /usr/local/etc/alcasar-wl-categories »*. La liste des seules catégories activées est située dans *« usr/local/etc/alcasar-wl-categories-enabled »*. Ce sont des liens symboliques pointant vers la liste complète.

Les adresses IP des utilisateurs « whitelistés » connectés (authentifiés) sont stockées dans l'IPSET « av_wl ». Lorsqu'un utilisateur se connecte au portail avec cet ipset, ALCASAR redirige les requêtes DNS de cet utilisateur vers le daemon « unbound-whitelist » (port UDP 55). Si le site en question est présent dans la liste des sites *« /usr/local/share/unbound-wl/*.conf »*, le daemon peut résoudre le nom de domaine et récupérer l'@IP du site qui est retourné à l'utilisateur. Cette @IP est alors ajoutée à l'IPSET « whitelist_ip_allowed » afin d'autoriser l'utilisateur à l'atteindre. Cet IPSET est vidé quand tous les utilisateurs « whitelistés » se sont déconnectés (cf. script *alcasar-flush_ipset_wl.sh* lancé par le daemon crond toutes les 5' (*/etc/crond.d/alcasar-watchdog.sh*)).

Cette méthode permet d'éviter qu'un utilisateur « whitelisté » puisse contourner le filtrage en se connectant sur un site directement au moyen de son adresse IP (sans résolution préalable de domaine).

Il est possible via l'ACC d'ajouter manuellement de nouveau site/@IP. Les sites seront ajoutés dans *« /usr/local/share/unbound-wl/ossi-wl.conf »*. Un lien symbolique sera alors créé pour que ces ajouts soient pris en compte ici *« /usr/local/share/unbound-wl-enabled/ossi-wl »*. Les adresses IP seront ajoutées dans *« /usr/local/share/ossi-ip-wl »*. Le script permettant cet ajout se trouve dans *« /usr/local/bin/alcasar-bl.sh »*. Voici un schéma récapitulatif du fonctionnement du double filtrage de la WL :



6.2.4 - Filtrage avec la BlackList (BL)

En utilisant ses serveurs DNS, ALCASAR va pouvoir déterminer si le site demandé par l'utilisateur est interdit. Si tel est le cas, l'utilisateur sera renvoyé vers l'@IP du portail (page de filtrage). Ce filtrage offre l'avantage de pouvoir interdire un nom de domaine, quel que soit le protocole demandé (HTTP, HTTPS, FTP, etc.).

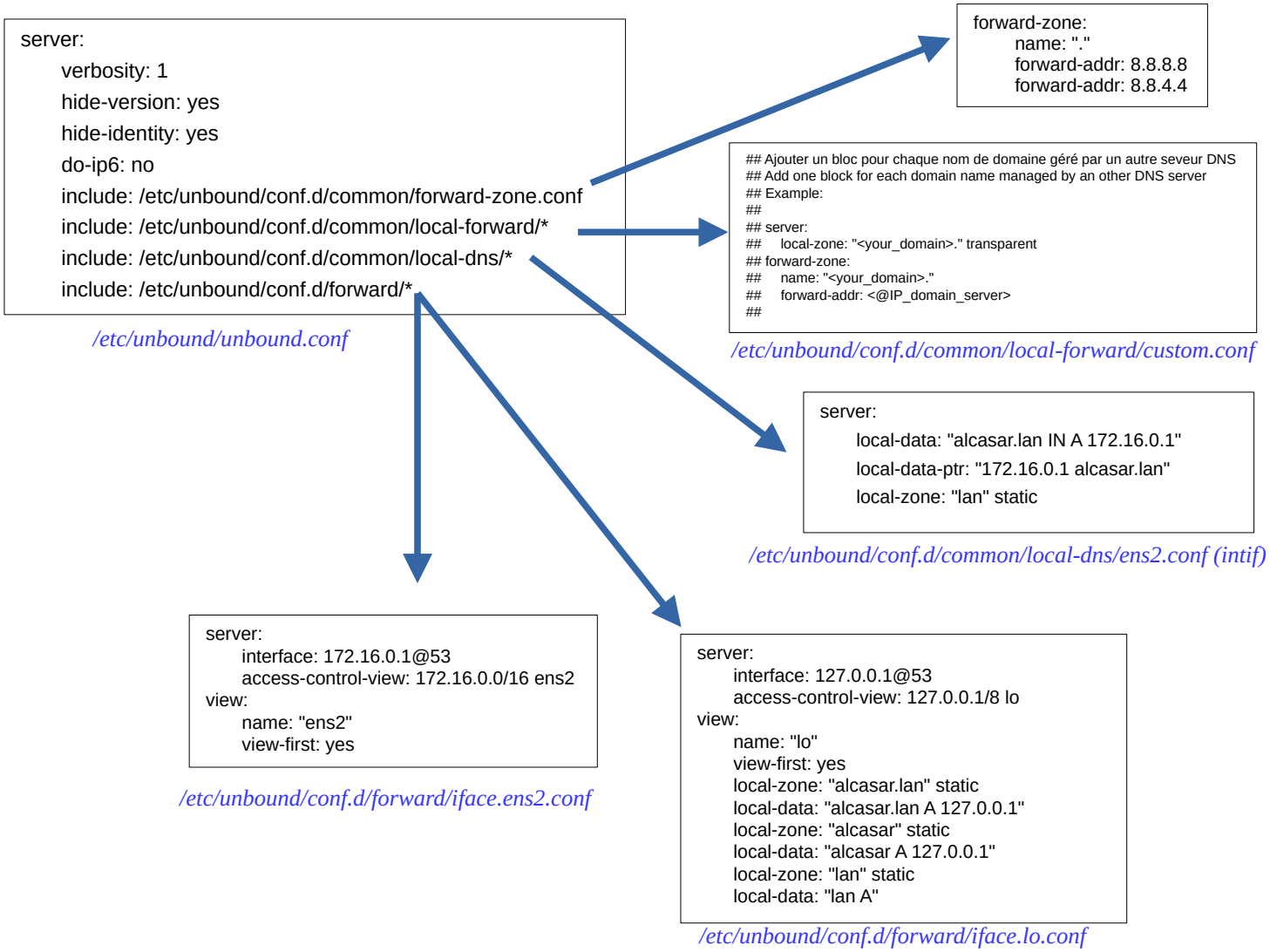
Tout comme la WL, la BL est organisée en catégories. ALCASAR permet de sélectionner ces catégories via l'interface de gestion (ACC). La liste des noms de catégorie (*/usr/local/etc/alcasar-bl-categories*) et la liste des catégories activées sont situées dans le répertoire de configuration d'ALCASAR (*/usr/local/etc/alcasar-bl-categories-enabled*).

6.2.5 - Configuration DNS

Dans cette fonction de filtrage, le serveur DNS joue un rôle principal. Ainsi, en fonction de son Ipset (cf. §6.2.2), l'utilisateur est redirigé sur un serveur DNS lié à son attribut de filtrage. 4 serveurs DNS (unbound) reçoivent ainsi les requêtes DNS des utilisateurs en fonction de leur redirection. Les schémas suivants montrent l'architecture de ces 4 serveurs DNS. Ces schémas s'appuient sur la configuration d'un ALCASAR d'exemple ayant les paramètres suivants : DNS=8.8.8.8 et 8.8.4.4 ; @IP_LAN=172.16.0.1/24 ; INTIF=ens2 ; nom de domaine par défaut (alcasar.lan).

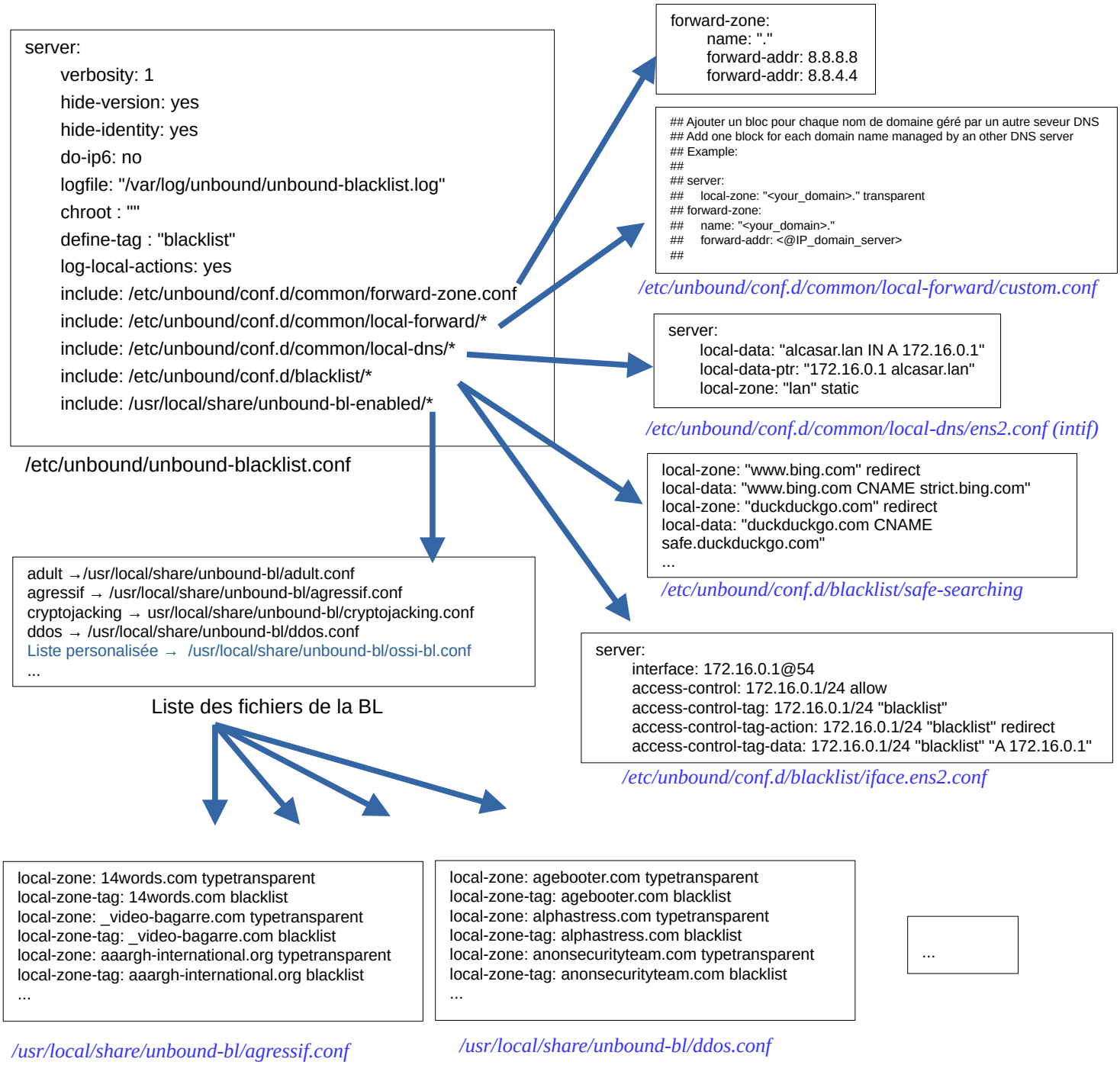
<u>Unbound forward (port 53)</u>

Cette instance de DNS reçoit les requêtes des utilisateurs sans filtrage.



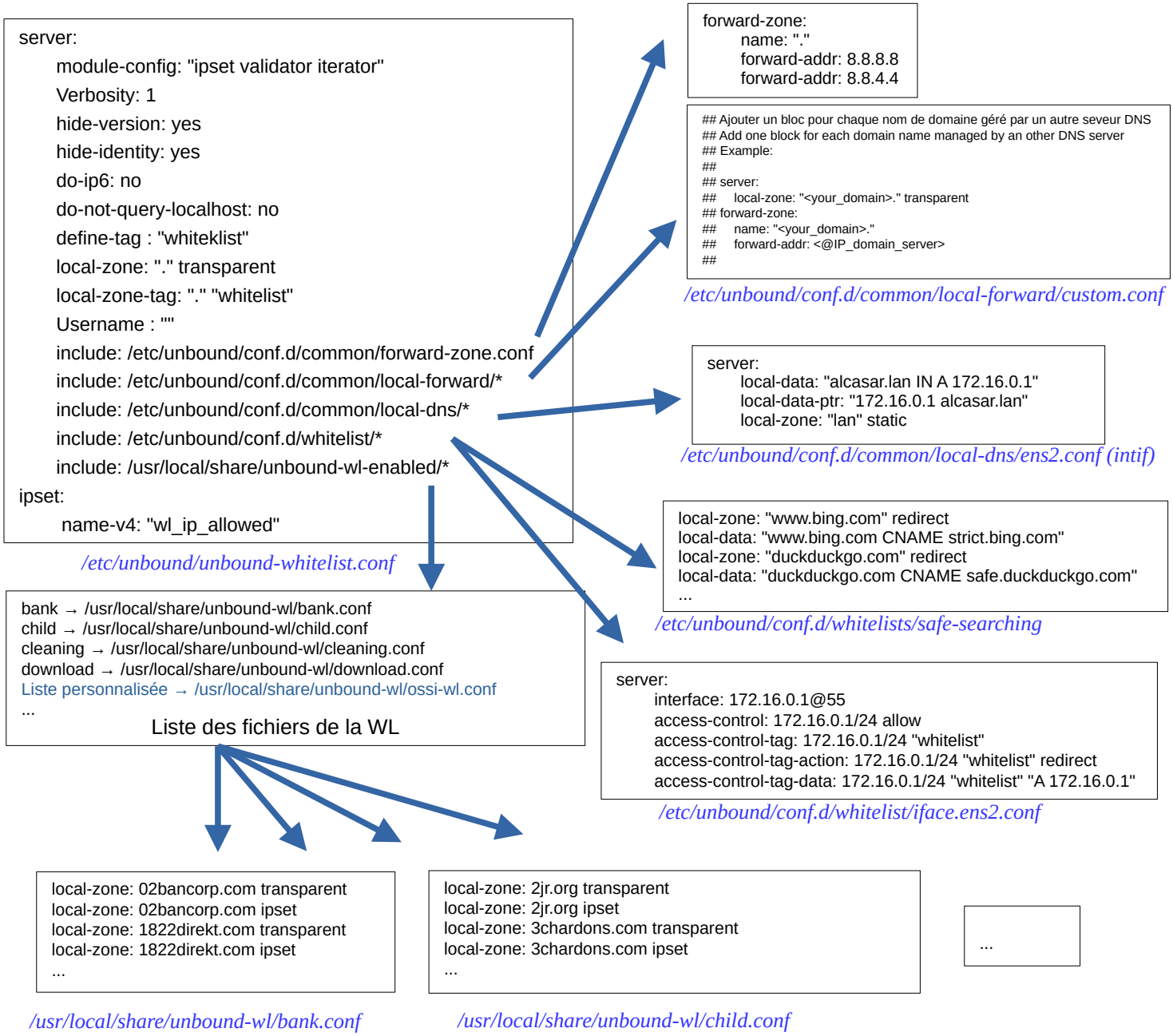
Unbound blacklist (port 54)

Cette instance de DNS reçoit les requêtes des utilisateurs « blacklistés ».



Unbound whitelist (port 55)

Cette instance de DNS reçoit les requêtes des utilisateurs « whitelistés ». Le module « ipset » est exploité pour alimenter dynamiquement netfilter (ipset : « wl_ip_allowed »). Pour cela, l'instance doit être lancée avec les droits « root ».

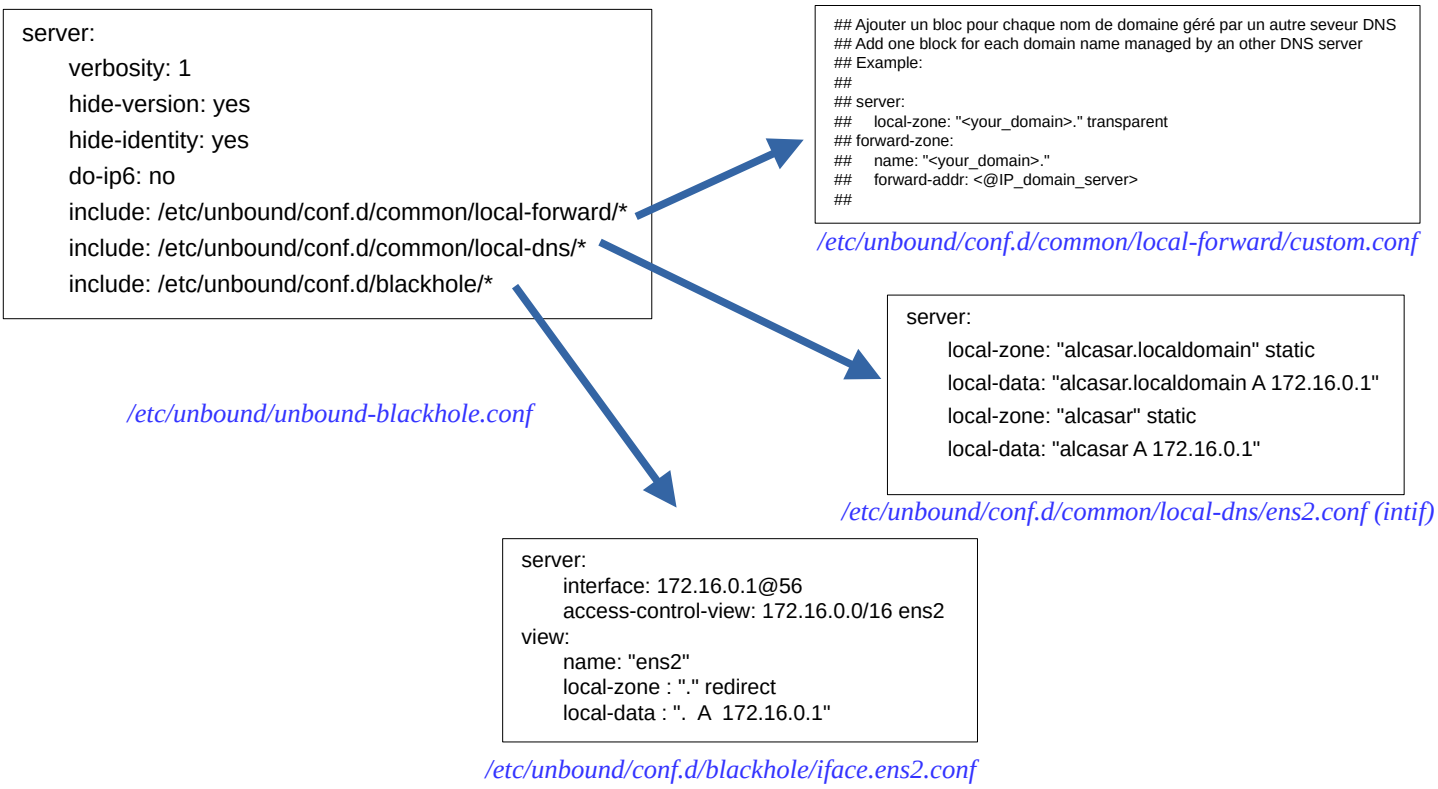


Particularité : Afin de supprimer les fenêtres de détection des portails captifs intégrées aux navigateurs, les domaines suivants doivent être whitelistés :

- Firefox : detectportal.firefox.com → déjà dans la liste « liste_bu »
- Chrome/Chromium : gstatic.com → déjà dans la liste « liste_bu »
- Edge : www.msftconnecttest.com (www.msftconnecttest.com/connecttest.txt)

Unbound blackhole (port 56)

Cette instance de DNS sert à rediriger toutes les requêtes des utilisateurs vers l'@IP d'ALCASAR. Cela n'arrive que quand le watchdog (alcasar-watchdog.sh) détecte que l'accès WAN est hors de service.



7 - Fonction ALCASAR Control Center

L'interface WEB de gestion d'ALCASAR (ACC = ALCASAR Control Center) est réalisée en PHP + javascript + jquery + bootstrap. Les possibilités de cette interface sont décrites dans la documentation d'exploitation.
L'ACC est situé dans le répertoire *« /var/www/html/acc »*. Le serveur WEB est « Apache ».

Le fichier de configuration de ce serveur est *« /etc/httpd/conf/httpd.conf »*.

L'ACC est protégé en accès par le module d'authentification « htdigest » d'Apache dont le fichier de configuration est *« /etc/httpd/conf/vhosts.d/alcasar.conf »*

Le répertoire *« /usr/local/etc/digest/ »* contient les fichiers des identifiants des administrateurs (et des empreintes de leurs mots de passe) en fonction de leur profil (*key_all*, *key_admin*, *key_manager* et *key_backup*).

8 - Modules complémentaires

8.1 - Import de comptes

Dans le cadre de la gestion des comptes d'authentifications, il est possible d'importer une liste de comptes attachés à un groupe prédéfini. Cette fonctionnalité accessible depuis l'interface de gestion génère un fichier <import-user>.pwd pour chaque importation et ajoute les utilisateurs dans le groupe (optionnel) de la base de données. Pour l'instant, seul le groupe peut-être attaché aux identifiants ; c'est-à-dire qu'aucun renseignement supplémentaire n'est importable pour le moment.

Le script *« import_user.php »* du répertoire *« /var/www/html/acc/manager/htdocs »* permet d'importer le fichier au format csv ou txt et le script *« import_file.php »* permet de ...

L'importation d'un fichier génère un fichier associé comportant les mots de passe en clair des utilisateurs importés. Ce dernier est téléchargeable pour être distribué aux utilisateurs. Afin de les supprimer périodiquement, une tâche, planifiée toutes les 30min lance le script *« alcasar-import-clean.sh »* qui cherche et supprime les fichiers datant de plus de 24h00.

8.2 - Inscription par SMS

8.2.1 - Fonctionnement global

Ce module permet aux utilisateurs de s'auto-inscrire sur ALCASAR en envoyant un mot de passe par SMS sur à ALCASAR. Un compte est alors créé dont le nom de login est le Nro de GSM de l'utilisateur. Ce module fonctionne grâce au projet « Gammu » (plus précisément Gammu_smsd) qui permet de stocker en base de données (cf. schéma de cette base en P11) les SMS reçus par un MODEM-GSM (ou clé-3G) connecté sur le port USB. Pour un bon fonctionnement, le code PIN de la carte SIM doit être renseigné dans le fichier de configuration de Gammu-smsd : *« /etc/gammu_smsd_conf ».*

Fonctionnement :

L'administrateur peut lancer *gammu-smsd* à partir de l'ACC (menu « Authentification/Inscription par SMS ». Il est possible de suivre le journal d'évènements : « tail -f /var/log/gammu-smsd/gammu-smsd.log »

À chaque lancement, le script « /usr/local/bin/alcasar-sms.sh --start » vérifie que le groupe « sms » est bien créé. Il le crée le cas échéant.

En fonctionnement, gammu-smsd dialogue avec le modem-GSM. Les SMS reçus par le modem-GSM sont alors récupérés puis stockés dans la table « inbox » de la base de données « gammu ».
Sur chaque réception de SMS, gammu-smsd, lance le script « /usr/local/bin/alcasar-sms.sh --new_sms ». Ce script permet de traiter le SMS selon le schéma de la page suivante.

Actuellement, l'administrateur à la possibilité :

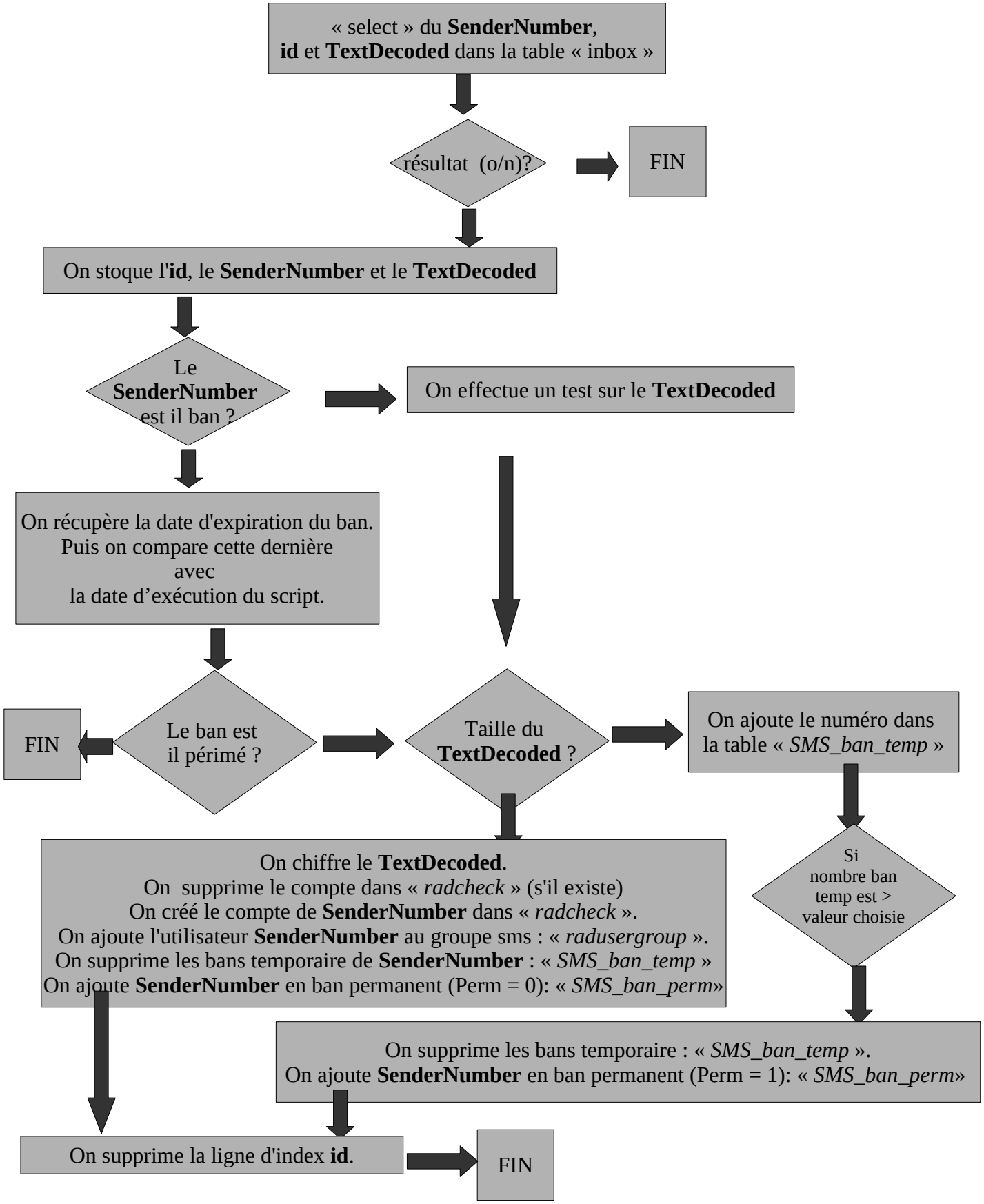
- *Lancer et arrêter gammu-smsd*
- *Renseigner le code PIN de la carte SIM présente dans le modem-GSM.*
- *Renseigner la durée d'une session pour les comptes auto enregistrés.*
- *Renseigner le nombre de bannissements temporaire permis avant le bannissement permanent.*
- *Renseigner la durée d'un bannissement permanent (en jours).*

On retrouve aussi sur cette page d'administration un tableau récapitulatif des comptes bannis :

- *soit pour une raison de Compte existant*
- *soit pour une raison d'excès d'envois de SMS au serveur (Flood).*

L'administrateur peut alors supprimer les numéros bannis. Cette action :

- *supprime le numéro de l'expéditeur du SMS de la table des bannissements permanents (« SMS_ban_perm »),*
- *supprime le numéro de la table du groupe « SMS » (« radusergroup »)*
- *supprime le numéro de la table des comptes radius (« radcheck »).*


« select » du **SenderNumber**, **id** et **TextDecoded** dans la table « inbox »
résultat (o/n)?
FIN
On stoque l'**id**, le **SenderNumber** et le **TextDecoded**
Le **SenderNumber** est il ban ?
On effectue un test sur le **TextDecoded**
On récupère la date d'expiration du ban. Puis on compare cette dernière avec la date d'exécution du script.
FIN
Le ban est il périmé ?
Taille du **TextDecoded** ?
On ajoute le numéro dans la table « *SMS_ban_temp* »
Si nombre ban temp est > valeur choisie
On chiffre le **TextDecoded**.
On supprime le compte dans « *radcheck* » (s'il existe)
On créé le compte de **SenderNumber** dans « *radcheck* ».
On ajoute l'utilisateur **SenderNumber** au groupe sms : « *radusergroup* ».
On supprime les bans temporaire de **SenderNumber** : « *SMS_ban_temp* »
On ajoute **SenderNumber** en ban permanent (Perm = 0): « *SMS_ban_perm*»
On supprime les bans temporaire : « *SMS_ban_temp* ».
On ajoute **SenderNumber** en ban permanent (Perm = 1): « *SMS_ban_perm*»
On supprime la ligne d'index **id**.
FIN

8.2.2 - Dialogue avec le modem-GSM – déblocage avec le code PUK

« gammu-smsd » dialogue avec le modem-GSM au moyen de commandes « AT ». Ces commandes permettent de faire un grand nombre d'actions, de la composition d'un numéro de téléphone à l'envoi de SMS, en passant par la récupération de l'état du modem. La syntaxe de ces commandes suit le schéma suivant : « AT+commande ». Les deux premiers caractères (AT) sont l'abréviation du mot 'ATtention'. Ils permettent d'avertir le modem, afin qu'il prenne en compte la commande qui suit. La syntaxe « AT^commande » correspond aux commandes « étendues » pour les modem-GSM de type 3G.

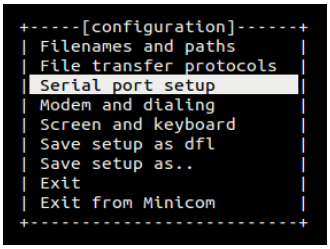
- Liste de commandes AT: *http://www.activexperts.com/sms-component/at/etsi/*
- Liste de commandes AT spécifique Huawei : rechercher « HUAWEI UMTS Datacard Modem AT Command Interface Specification » dans un moteur de recherche.

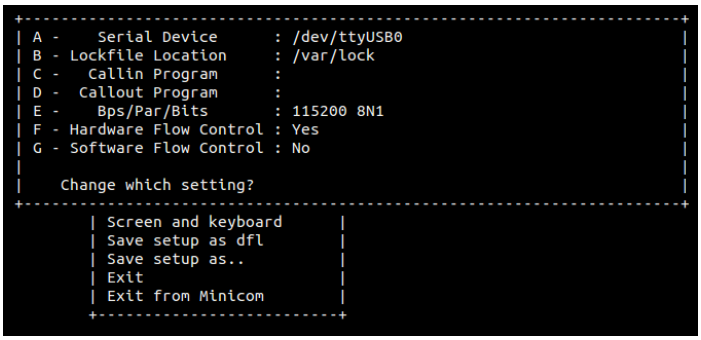
Une fois « gammu-smsd » lancé, il vérifie l'état de la carte SIM en envoyant la commande : « AT+CPIN? ». Le modem-GSM répond en demandant le code PIN de la carte SIM. Gammu-smsd utilise alors le code PIN écrit dans le fichier de configuration « */etc/gammu_smsd_conf* ».

Si ce code PIN est erroné, la carte SIM sera bloquée. Il faudra alors exploiter le code PUK pour la débloquer. La manipulation suivante permet de débloquer la carte SIM à l'aide d'un terminal et de commandes AT.

Arrêtez le service gammu via l'ACC ou via la commande « *alcasar-sms.sh --stop* »

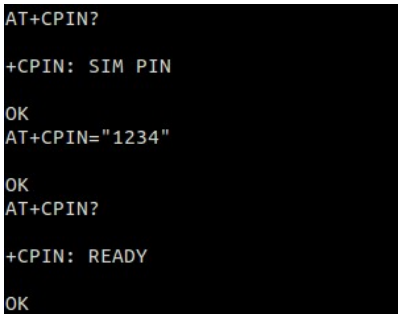
Dans un premier temps, installez « *Minicom* » sur votre système Linux : « urpmi minicom ». Modifiez la configuration de *Minicom* en lançant la commande « minicom -s ». Sélectionnez la 3e entrée (Serial port setup // Configuration du port série). Configurez le port série avec les paramètres suivants (cf. copie d'écran ci-dessous). Une fois les configurations effectuées, appuyer sur « échap » puis déplacer vous dans le menu pour enregistrer les modifications (Save setup as dfl // Enregistrer config. sous dfl). Vous pouvez alors quitter le menu (Exit from Minicom // Sortir de Minicom).





Pour se connecter au modem-gsm, lancez la commande : « minicom -c on ». Vérifier la connexion au modem avec la commande suivante : « AT ». Si la configuration est correcte, le modem devrait renvoyer « OK ».

La commande « AT+CPIN ? » permet de connaitre l'état de la carte SIM. La copie d'écran ci-contre, montre la demande de code PIN de la carte SIM afin de pouvoir être exploitée : « +CPIN : SIM PIN ». La commande « AT+CPIN="xxxx" » (où xxxx correspond à votre code PIN), permet de renseigner le code PIN de la carte. Testez alors à nouveau l'état de la carte SIM.



Dans le cas où la carte SIM est bloquée, le modem retourne « +CPIN : SIM PUK ». Récupérez le code PUK (disponible généralement en ligne sur le compte d'abonnement, ou fourni avec la carte SIM). Exécutez alors la commande « AT+CPIN="yyyyyyyy","zzzz"» (où yyyyyyyy correspond à votre code PUK et zzzz correspond à votre nouveau code PIN).

Remarque :

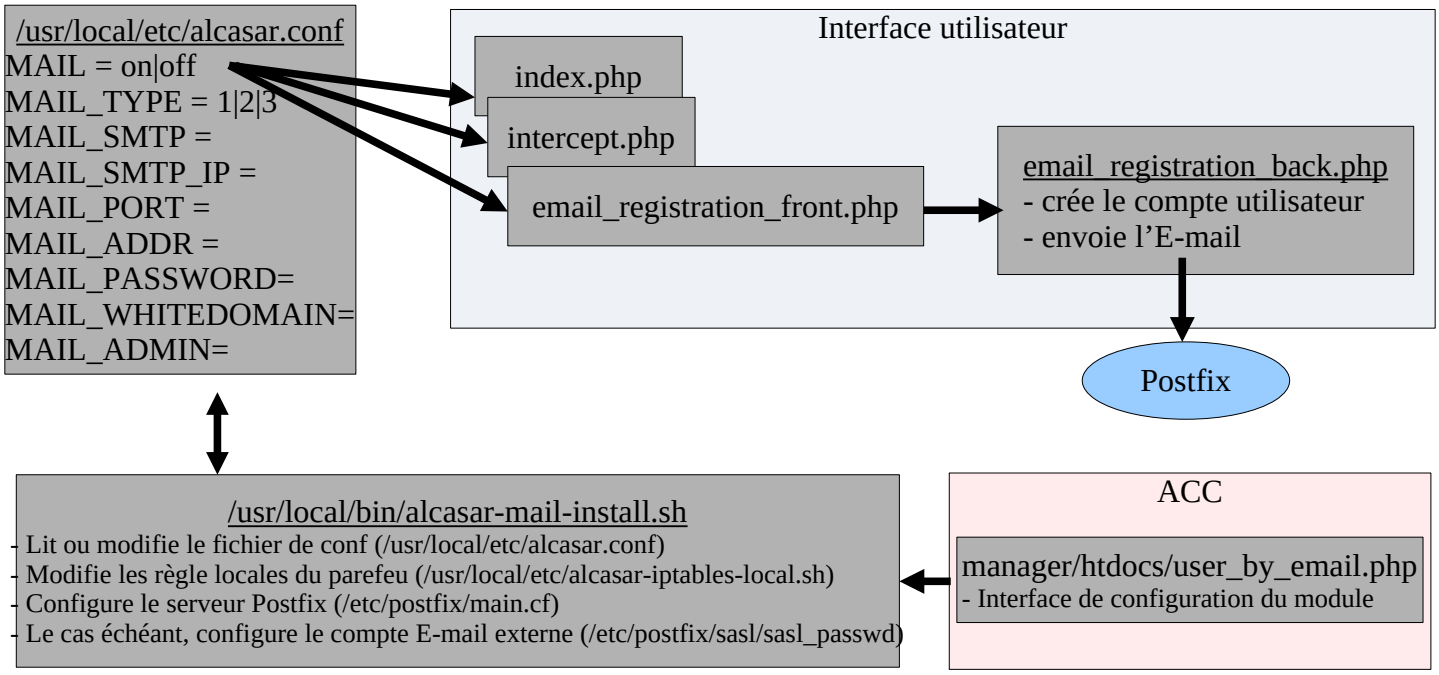
- Vous pouvez accéder au menu de Minicom via la combinaison de touche : « Ctrl+a » puis « z »
- Le modem vous retourne « OK » si la commande envoyée est correcte (syntaxiquement) et reconnue.

8.3 - Inscription par adresse électronique

L'objectif de ce module est de permettre aux utilisateurs de s'auto-inscrire en renseignant une adresse e-mail. ALCASAR crée alors un nouvel utilisateur donc le login est l'adresse e-mail. Un mot de passe aléatoire est envoyé à cette adresse. Comme ce type d'inscription ne permet pas systématiquement d'imputer les traces de connexion (cas des adresses e-mail à usage unique par exemple), l'administrateur doit configurer les noms de domaine qui seront les seuls autorisés (ex : airbus.com, sncf.fr, etc.). Ce module a été initialement imaginé et développé par K@M3L & T3RRY (laplateforme.io).

Bien qu'ALCASAR puisse envoyer des e-mails selon les 3 méthodes présentées ci-après, seule la 3è a été implémentée :

1. il est serveur de messagerie ;
2. Il relaie vers un serveur de messagerie externe (serveur d'entreprise par exemple) ;
3. Il utilise un compte de messagerie géré par un serveur externe (free, sfr, orange, etc.).



Débug : Les journaux de Postfix sont visibles via « journalctl -f -u postfix ». Il est possible d'augmenter le niveau de log par serveur de mail en ajoutant l'entrée « debug_peer_list = free.fr, gmail.com » dans « */etc/postfix/main.cf* »
La configuration de postfix est visualisable via la commande « *postconf -a* ».
Pour gérer la queue de messages de postfix : https://mailmum.io/posts/manage-postfix-mail-server-queues/

- Lire la queue de messages : « *postqueue -p » ou « mailq* »
- Lire le message Id de la queue : « *postcat -q Id* »
- Supprimer le message Id de la queue : « *postsuper -d Id* » ou « *postsuper -d ALL* »

Pour la 3e méthode (postfix comme « client de messagerie »), Postfix exploite les bibliothèques « cyrus-sasl » pour s'authentifier sur un serveur externe (vérifiable via la commande « *postconf -A* »).

La qualité des Email peut-être testée via le site « mail-tester.com ».

7 serveurs de messagerie sont pré-enregistrés dans « *manager/htdocs/user_by_email.php* » : *smtp.orange.fr port 465, smtp.office365.com port 587, smtp.sfr.fr port 465, smtp.free.fr port 465, smtp.gmail.com port 587, smtp.laposte.net port 465, smtp.bbox.fr port 465*

8.4 - Watchdog

Ce script (*alcasar-watchdog.sh*) est lancé toutes les 10 minutes par le Daemon « crond ». Il est ainsi possible de le désactiver en commentant la ligne ci-dessous du fichier *« /etc/cron.d/alcasar-watchdog »*, puis, d'avertir « crond » de la modification : *« systemctl reload crond.service » :*

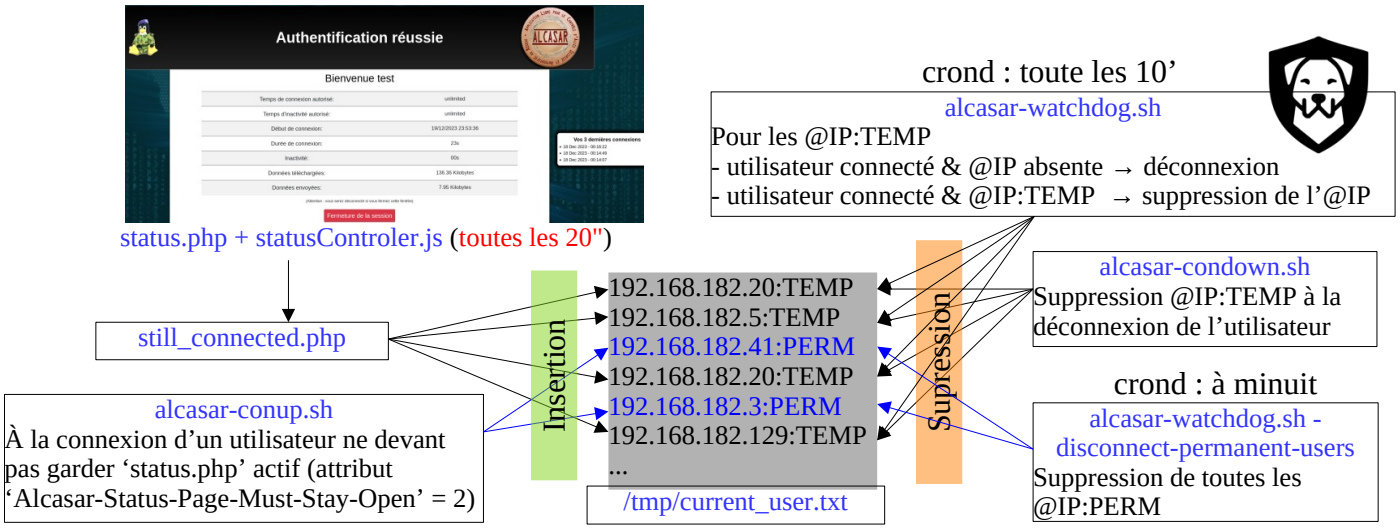
```
#*/10 * * * * root /usr/local/bin/alcasar-watchdog.sh > /dev/null 2>&1
```

Les différentes fonctions du « watchdog » sont les suivantes :

- Limiter le risque d'usurpation d'adresse IP et d'adresse MAC (cas d'un pirate qui serait connecté sur le réseau de consultation). Pour cela, le script lance une requête ARP vers toutes les @IP d'équipements ayant un utilisateur connecté. Pour chaque @IP, si plus d'une réponse est reçue (usurpation), le compte est déconnecté, une alerte est levée via syslog et une trace est écrite dans le fichier « /var/Save/security/watchdog.log » et via « syslog ».

- Afficher une page WEB aux utilisateurs en cas de problèmes de connectivité Internet. Le script diagnostique la connectivité Internet (1-carte réseau « extif » active et connectée, 2-le routeur par défaut répond, 3-la résolution DNS fonctionne). En cas d'échec, toutes les requêtes DNS du réseau de consultation sont dirigées vers le daemon « unbound-blackhole » qui les résout avec l'@IP d'ALCASAR ce qui permet de présenter une page d'erreur.

- Éviter les « oublis » de déconnexion (utilisateur ayant éteint son système sans s'être déconnecté, utilisateur ayant fermé la fenêtre de statut, équipement utilisateur ayant quitté le réseau ou ne répondant plus, etc.). Fonctionnement : un fichier temporaire (/tmp/current_users.txt) est alimenté dynamiquement avec les @IP des utilisateurs connectés. Si le watchdog constate que l'@IP d'un utilisateur censé être connecté n'est pas dans ce fichier temporaire, il le déconnecte (sauf si cet utilisateur à l'attribut « Alcasar-Status-Page-Must-Stay-Open » égal à 2). L'alimentation de ce fichier est réalisée de la manière suivante :



Évolution possible :

- *Remplacer le fichier temporaire par un « ipset » (ressource moins gourmande) ;*
- *Remplacer ce système par le suivant :*
  - *L'attribut radius 'Alcasar-Status-Page-Must-Stay-Open' étant un entier, on gère sa valeur comme un nombre d'heures où l'utilisateur peut rester connecter (sans obligation d'avoir sa fenêtre de statut ouverte).*
  - *Lors de la connexion d'un utilisateur possédant l'attribut 'Alcasar-Status-Page-Must-Stay-Open', « alcasar-conup.sh » crée un nouvel utilisateur de type « utilisateur de confiance » avec les attributs suivants :*
    - *« login » = @MAC récupérée ;*
    - *« date d'expiration » = date d'expiration de l'utilisateur (ou date actuelle + « Alcasar-Status-Page-Must-Stay-Open » si inférieure à la date d'expiration de l'utilisateur) ;*
    - *nom&prénom = login_de_l'utilisateur (pour la traçabilité).*

8.5 - Statistiques réseau

Afin de protéger la vie privée des utilisateurs conformément aux préconisations de la CNIL, les statistiques de navigation ne comportent pas d'éléments permettant de lier les flux aux comptes des utilisateurs.

Il est possible via l'interface d'administration d'obtenir une représentation graphique de la charge réseau d'ALCASAR. Une sonde Netflow a été compilée à cet effet. Deux règles de pare-feu permettent de traiter tous les flux sortants par cette sonde

- *Flux transitant dans les proxy HTTP internes :* $IPTABLES -A OUTPUT -o $EXTIF -p tcp --dport http -j NETFLOW
- Flux sortant directement : $IPTABLES -A FORWARD -i $TUNIF -s $PRIVATE_NETWORK_MASK -m state --state NEW -j NETFLOW

Le module ipt_NETFLOW d'ALCASAR exporte ses informations sur la socket 127.0.0.1 :2055.

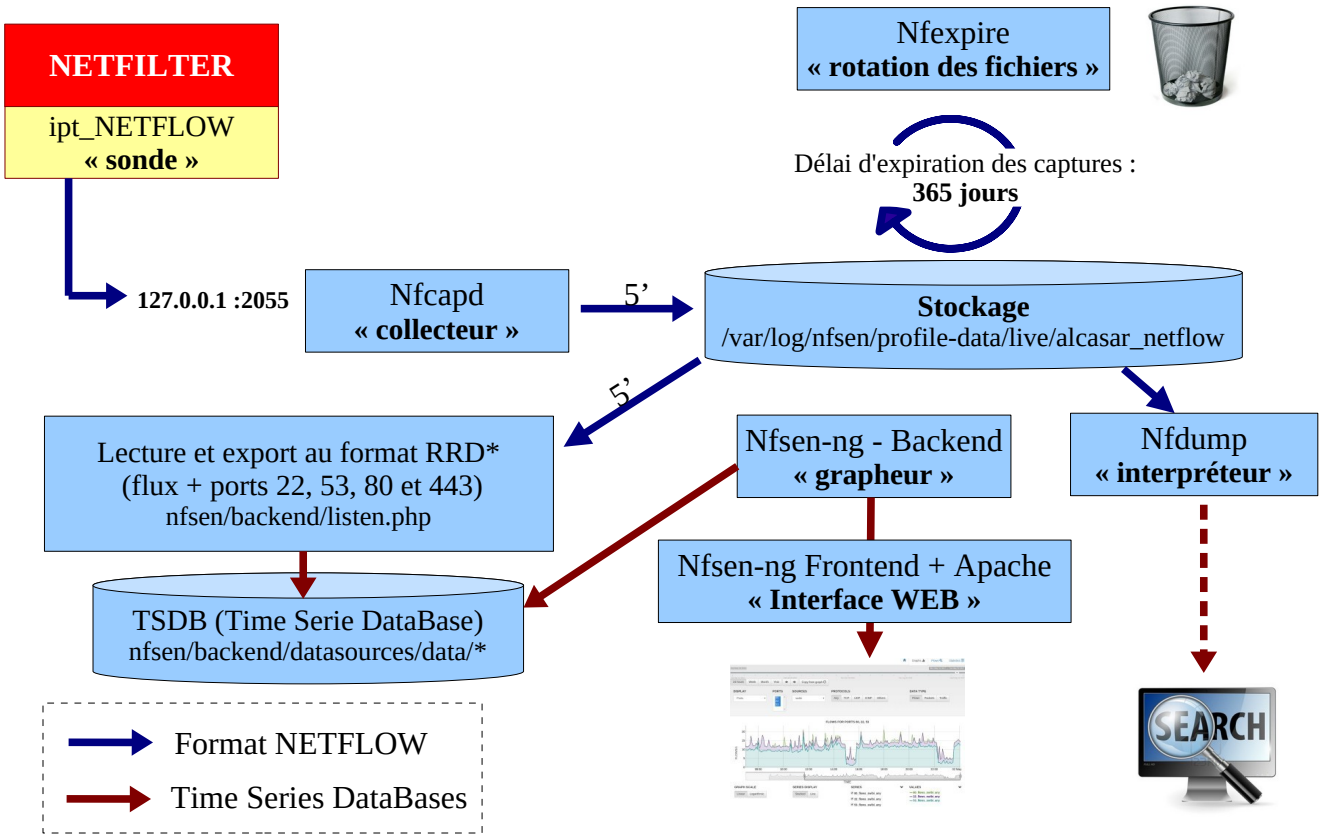
La fonction de collecteur est prise en compte par le démon « nfcapd » en écoute sur le port 2055. Il collecte les flux NetFlow et crée un fichier toutes les 5 minutes dans le répertoire *« /var/log/nfsen/profile-data/live/alcasar_netflow/ ».*

Le module **Nfexpire** (installé avec le RPM « nfdump ») permet de réaliser une rotation sur les fichiers capturés par « nfcapd ». Une règle ajoutée à *« /etc/cron.d/alcasar-nfcapd-expire »* permet d'actualiser tous les jours le délai d'expiration sur le répertoire contenant les fichiers de capture Netflow : *nfexpire -e /var/log/nfsen/profile-date/live/alcasar_netflow/ -t 365d*

Tel quel, le format Netflow n'est pas lisible, en revanche il est possible à tout moment d'afficher le contenu des fichiers de capture de manière lisible. Il faut pour cela utiliser un **interpréteur Netflow** (ex : «nfdump ») comme suit : nfdump -R <fichier_au_format_netflow> -o extended -a

L'ACC (menu « statistiques » + « trafic détaillé ») s'appuie sur le projet « nfsen-NG » pour visualiser graphiquement les flux Netflow après les avoir transformés et stockés en séries chronologiques via les RDDtools.

Récapitulatif de l'architecture NETFLOW d'ALCASAR



*RRD (Round Robin Database) est une base de données de séries chronologiques (Time Series DataBase – TSDB). Ces TSDB sont conçues pour organiser des informations mesurées dans le temps.

8.6 - Contournement (by-pass)

En cas de problème technique concernant une des briques logicielles du portail (principalement « coova-chilli »), il est possible de court-circuiter le module d'authentification tout en maintenant le traçage des logs réseau (pare-feu).

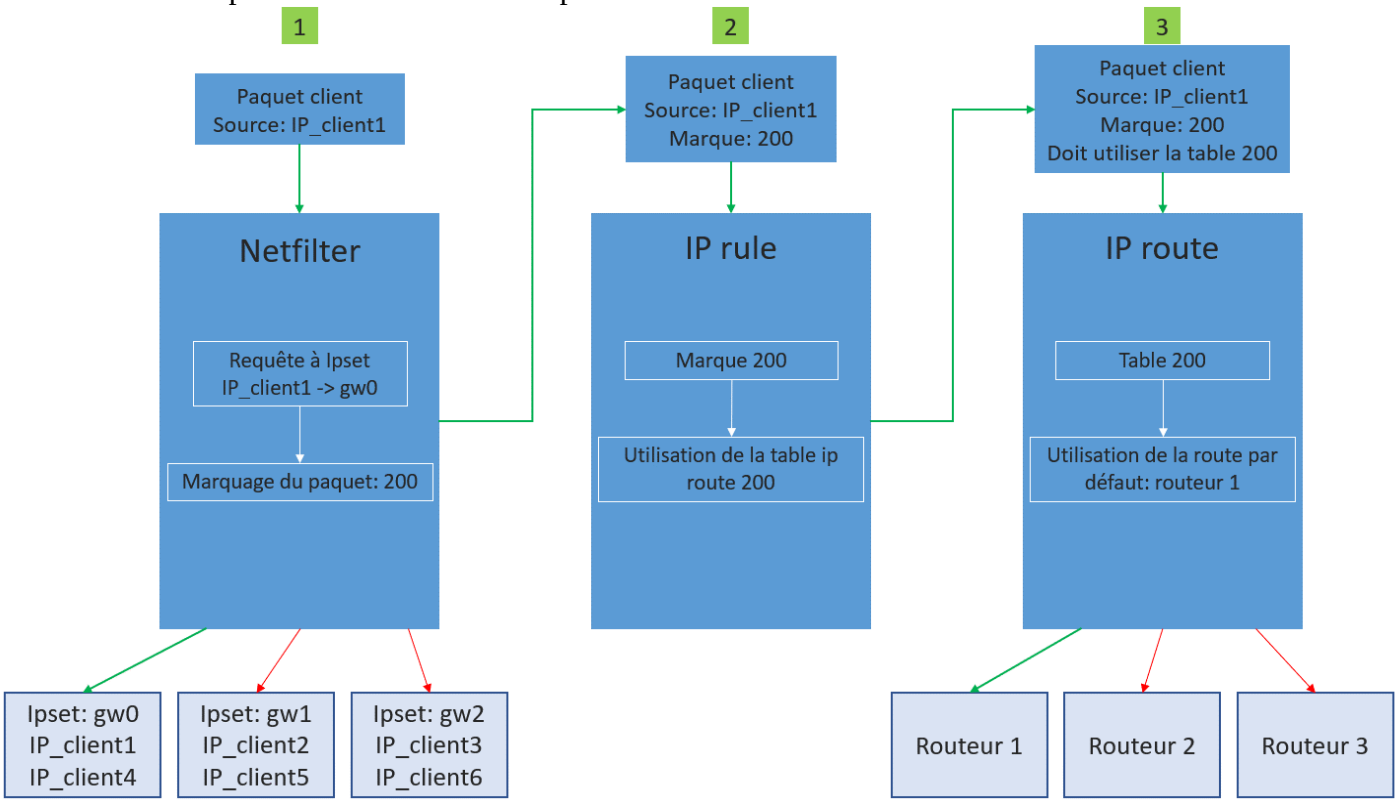
Un script lancé localement en root *alcasar-bypass.sh –on* | *--off* permet au choix de mettre :

- en mode « On » le bypass → le portail désactive les services coova-chilli et E²Guardian
- en mode « Off » : le portail est en mode normal. Tous les services nécessaires sont activés.

8.7 - Équilibrage de charge en sortie (load balancing)

ALCASAR intègre un dispositif spécial permettant d'équilibrer la charge sur plusieurs routeurs de sortie. Ce dispositif est décrit dans un document externe disponible sur le site WEB. Nous avons décidé de créer notre propre dispositif d'équilibrage de charge après avoir constaté que les systèmes existants ne donnaient plus satisfaction (équilibrage de charge par adresses de destination ou par protocoles réseau). Le système mis en oeuvre dans ALCASAR effectue un équilibrage de charge par systèmes/utilisateurs authentifiés.

Le fichier de configuration d'ALCASAR (/usr/local/etc/alcasar.conf) intègre ainsi une entrée d'activation (MULTIWAN=On/Off) et autant d'entrées que de routeurs complémentaires (WAN1=w.x.y.z,1 WAN2=a.b.c.d,1 WAN3=...). Le routeur par défaut reste défini par l'entrée "GW=". Un "ipset" est créé pour chaque routeur (gw0, gw1, etc.). Les @ip des systèmes authentifiés (utilisateurs ou @MAC) sont distribuées (algorithme "Round Robin" ou "tourniquet") dans chacun de ces ipsets. Des règles de routage (ip rule) intégrant ces ipsets distribuent les utilisateurs sur leur routeur respectif. L'administrateur peut affecter un "poids" à chaque routeur afin d'ajuster la charge qu'il recevra en termes de nombre d'utilisateurs. C'est le script "alcasar-network.sh" (qui est appelé par l'ACC) qui applique les différentes règles de routage en fonction du nombre de routeurs et de leur poids. Le schéma suivant présente le cheminement des flux d'un utilisateur :



Étape 1 : À chaque authentification réussie, « coova » lance le script « alcasar-conup.sh » qui détermine le routeur de sortie en fonction de la répartition de charge et qui affecte l'@IP de l'utilisateur dans l'ipset correspondant (gwX). Dans sa table de « prerouting », Nefilter marque les paquets en fonction de cet ipset (200, 201, 202, etc.). Lors d'une déconnexion, « coova » lance le script « alcasar-condown.sh » qui supprime l'@IP de l'utilisateur de son ipset.

Étape 2 : Des règles de routage sont mises en place par le script « alcasar-network.sh » (lancé au démarrage et/ou appelé par l'ACC). Ces règles sont visibles via la commande « ip rule ». Exemple avec 2 routeurs :

```
ip rule
0:      from all lookup local
32764:  from 172.16.0.0/16 fwmark 0xc9 lookup 201
32765:  from 172.16.0.0/16 fwmark 0xc8 lookup 200
32766:  from all lookup main
32767:  from all lookup default
```

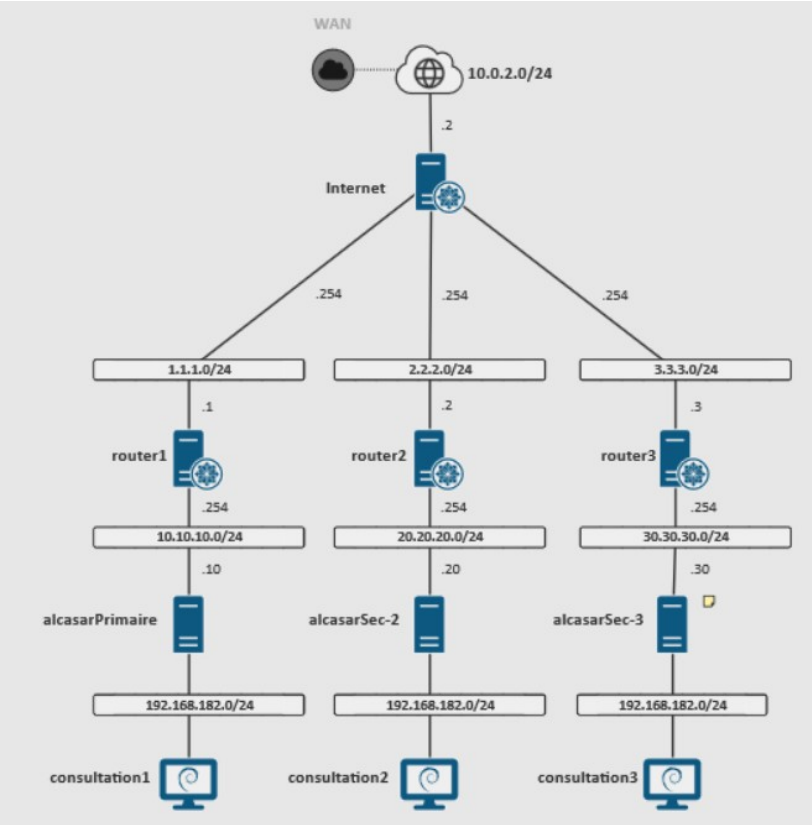
Étape 3 : Les paquets de l'utilisateur sont traités par la table de routage associée à la marque. La commande « ip r » permet d'afficher une synthèse des tables de routage. La commande « ip r show table xxx » permet d'afficher la table de routage xxx. Exemple avec 2 tables :

```
ip r  (ou ip r show table main)
default
      nexthop via 192.168.0.254 dev eno1 weight 1
      nexthop via 192.168.0.253 dev eno1 weight 1
172.16.0.0/16 dev tun0 proto kernel scope link src 172.16.0.1
192.168.0.0/24 dev eno1 scope link src 192.168.0.201

ip r show table 200
default via 192.168.0.254 dev eno1
192.168.0.0/24 dev eno1 scope link src 192.168.0.201


ip r show table 201
default via 192.168.0.253 dev eno1
192.168.0.0/24 dev eno1 scope link src 192.168.0.201
```

Pour tester ce dispositif, une architecture virtuelle a été créée sur une « CyberRange ». La topologie est la suivante :



8.8 - Sauvegardes

Les sauvegardes d'ALCASAR sont disponibles sous 3 formes : l'archive des journaux de traçabilité, la base de données des utilisateurs et le rapport d'activité hebdomadaire.

8.8.1 - Sauvegarde des journaux de traçabilité

Les journaux du firewall (*/var/log/firewall/**) et de la sonde Netflow (*/var/log/nfsen/profiles-data/live/alcasar-netflow*) sont « rotatés » chaque semaine. Chaque semaine (lundi à 5h35), une tâche planifiée appelle le script « *alcasar-archive.sh -now* » qui crée une archive compressée, constituée de ces journaux et de la base de données des utilisateurs. Cette archive est copiée dans le répertoire « */var/Save/archive* » afin d'être disponible dans l'ACC pour téléchargement (fichier « *traceability-<date>.tar.gz* »). Afin de limiter la conservation des traces à 1 an, ce script efface toutes les archives dont la date de création est supérieure à 365 jours.

8.8.2 - Sauvegarde de la base de données

Chaque semaine (lundi à 04h45), le script « *alcasar-mariadb.sh –dump »* vérifie, sauvegarde et compresse la base de données des utilisateurs dans le répertoire « */var/Save/base »* sous la forme : « *alcasar-users-database-<date>.sql.gz* ».

Chaque nuit à 4h40, le script « *alcasar-mariadb.sh --expire_user* » supprime les utilisateurs dont la date d'expiration est dépassée de plus de 7 jours.

8.8.3 - Le rapport d'activité hebdomadaire

Ce rapport au format « PDF » est généré par le script « *alcasar-activity_report.sh* » tous les dimanches matin à partir de 5h35. Les rapports sont disponibles via l'ACC. Ils sont stockés dans le répertoire « /*var/Save/activity_report/* ».

8.9 - WIFI4EU

La procédure technique est disponible ici :
https://ec.europa.eu/inea/sites/inea/files/wifi4eu/cnect-2017-00250-00-11-fr-tra-00_0.pdf

Deux attributs du fichier de conf d'alcasar sont utilisés (WIFI4EU=on/off et WIFI4EU_CODE=). Le code de test est « 123e4567-e89b-12d3-a456-426655440000 ».

Dans l'ACC, la page de gestion des services permet de modifier l'identifiant de réseau et d'activer/désactiver ce service en appelant le script « alcasar-wifi4eu.sh » qui effectue les modifications suivantes :

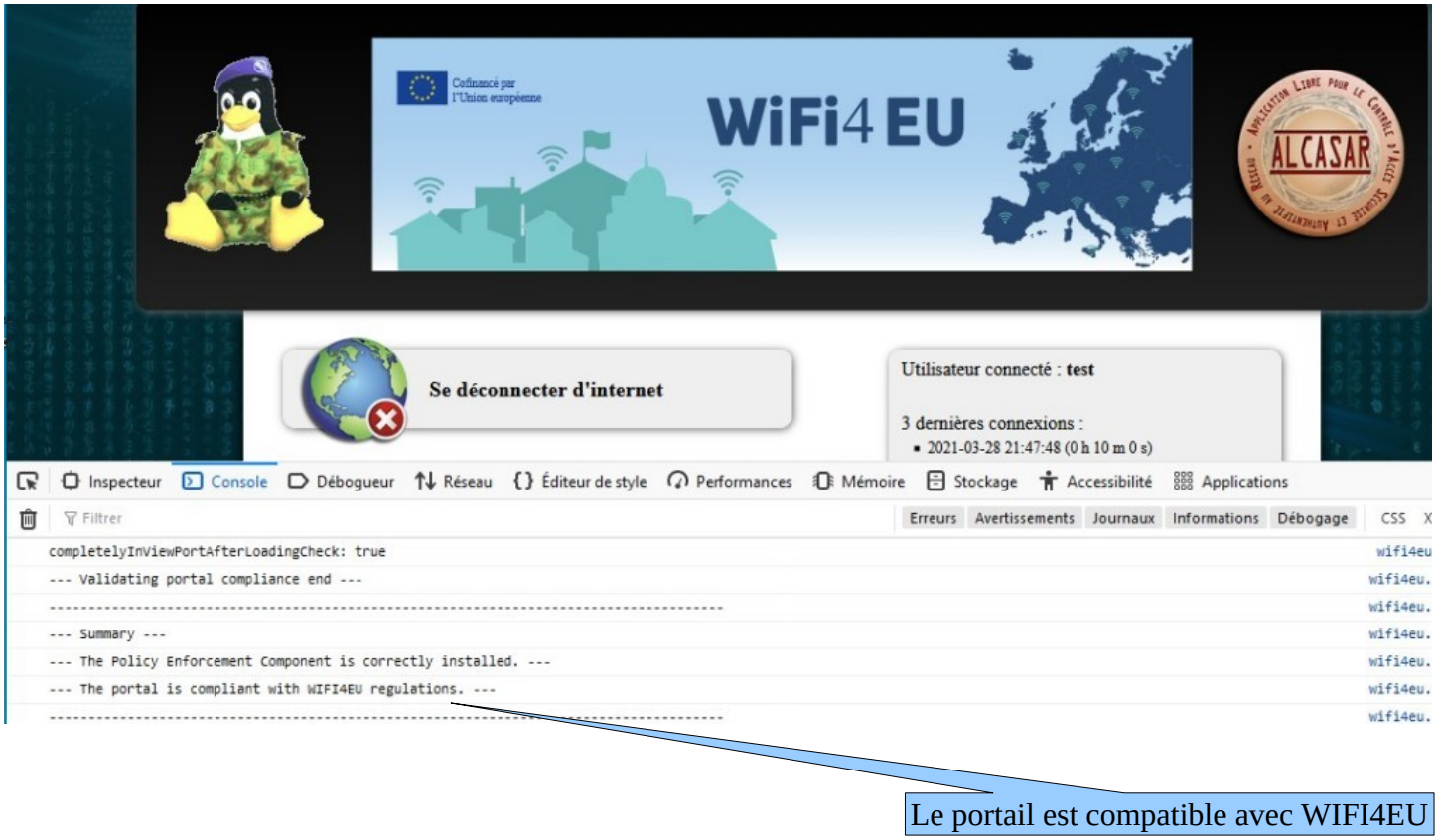
- ajout du site « collection.wifi4eu.ec.europa.eu » comme site de confiance (inscription dans le fichier : « /usr/local/etc/alcasar-uamdomain »).
- Les pages « /*var*/www/html/index.php » activent le code suivant entre leurs balises <head>

```
<script type="text/javascript">
        var wifi4euTimerStart = Date.now();
        var wifi4euNetworkIdentifier = '123e4567-e89b-12d3-a456-426655440000';
        var wifi4euLanguage = 'fr';
        //var selftestModus = true;
</script>
<script type="text/javascript" src="https://collection.wifi4eu.ec.europa.eu/wifi4eu.min.js"></script>
```

- Dans ces pages, on remplace le titre de la page par la bannière « WIFI4EU » (<img id="wifi4eubanner">)
  - index.php (ligne 563)
  - status.php (ligne)
  - intercept.php (ligne)

Test de bon fonctionnement :

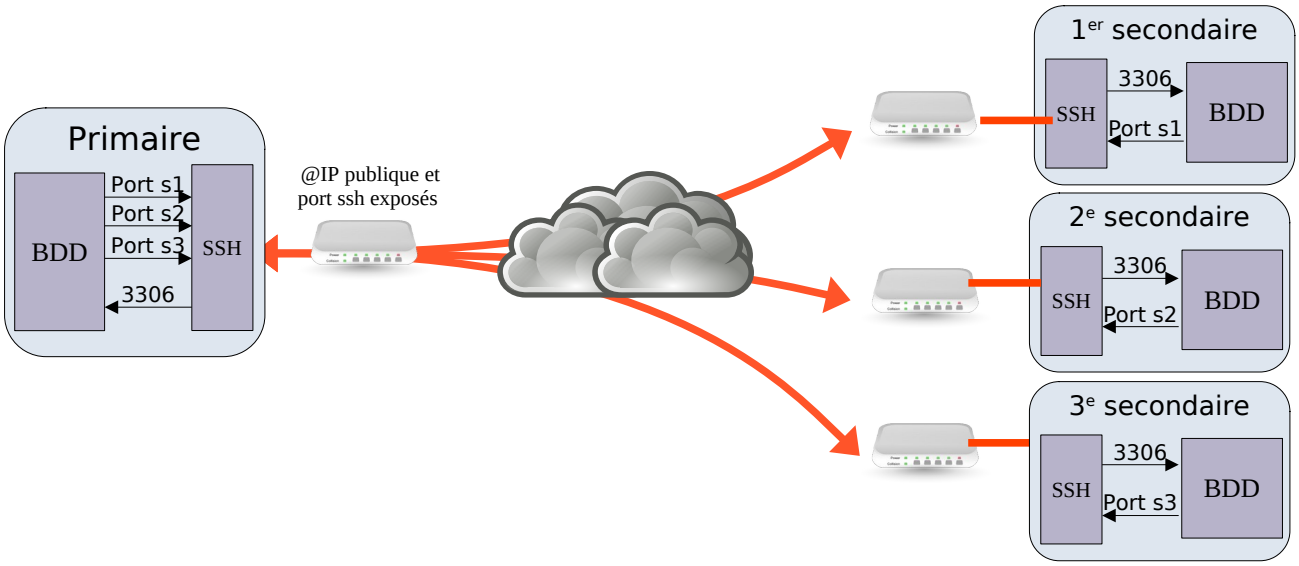
Pour évaluer la validité de cette intégration, décommentez la ligne « var selftestModus = true; ». Ouvrez la « console web » de débug du navigateur pour afficher le journal de validation :



Le portail est compatible avec WIFI4EU

8.10 - Fédération

Le principe de la fédération est de créer des redirections de ports bidirectionnelles dans des tunnels SSH afin de permettre aux serveurs « mariadb » de gérer des répliqua ensemble de manière transparente en exposant leur port d'écoute 3306 uniquement sur l'interface loopback (127.0.0.1). Afin de limiter l'exposition sur Internet, seul l'ALCASAR primaire expose un port SSH.



Un exemple de commande permettant d'activer ce type de tunnelling à partir d'un secondaire : /usr/bin/ssh -NT -4 -o ServerAliveInterval=60 -o ExitOnForwardFailure=yes -p port_ssh_exposé -L port_s1:localhost:3306 -R port_s1:localhost:3306 replication@IP secondaire

En prenant l'exemple du schéma ci-dessus :

1. Lorsque la réplication est configurée, le secondaire se connecte via SSH de manière persistante sur le primaire via le port s1. Le primaire reçoit cette connexion sur le port 3306.
2. Lors de l'activation de la réplication via le tunnel SSH (START SLAVE), le secondaire reçoit le flux des évènements (transactions) depuis les logs binaires du primaire, au fur et à mesure qu'ils sont écrits. Dès qu'ils sont reçus, le secondaire les écrit dans ses « relay logs » locaux.
3. Le secondaire lit ses « relay logs » et applique les modifications à ses propres données.

8.10.1 - Installation/désinstallation de la fédération sur primaire ou secondaire (sur chaque ALCASAR)

- **alcasar-replication-install.sh**
  - Crée l'utilisateur BDD « db_replication ». ajoute « db_replication » + mdp dans « /root/ALCASAR-passwords.txt » ;
  - Crée le fichier de configuration Mariadb : « /etc/my.cnf.d/replication.cnf » ;
  - Autorise Mariadb à écouter en réseau sur 127.0.0.1 (modif « /etc/my.cnf.d/server.cnf ») ;
  - Crée le fichier de configuration du logging binaire et du REPLICA (/etc/my.cnf.d/replication.cnf) ;
  - Relance Mariadb
  - Crée l'utilisateur Linux « replication » Ajoute « replication » + mdp dans « /root/ALCASAR-passwords.txt » ;
  - Crée le fichier « .ssh/authorized_keys » dans « /home/replication » ;
  - Crée une biclé RSA dans « /root/.ssh » : id_rsa et id_rsa.pub ;
  - Crée le fichier /home/replication/local-db_replication-pwd.txt contenant le pwd de « db_replication »
  - Fixe la clé « REPLICATION=on » dans « alcasar.conf ».

- **alcasar-replication-uninstall.sh**
  - Stop les réplications en cours (/usr/local/bin/alcasar-replication-stop.sh –all) ;
  - Supprime toutes les conf de réplication (/usr/local/bin/alcasar-replication-delete.sh –all) ;
  - Supprime les entrées du fichier « /root/ALCASAR-passwords.txt » (replication et db_replication) ;
  - Supprime l'utilisateur BDD « db_replication » ;
  - Supprime le fichier de configuration du REPLICA ;
  - Supprime l'écoute de Mariadb sur 127.0.0.1 ;
  - Relance Mariadb
  - Supprime l'utilisateur Linux « replication » et son homedirectory. Tue ses processus actifs ;
  - Fixe la clé « REPLICATION=off » et la clé « REPLICATION_TO= » dans « alcasar.conf »
  - Relance le parefeu (fermeture des sorties SSH)

8.10.2 - Copie de la clé publique SSH d'un secondaire vers le primaire.

- **alcasar-replication-ssh-keys-management.sh --add -file …**
  - importe la clé publique à partir d'un fichier dans « /home/replication/.ssh/authorized_keys »
- **alcasar-replication-ssh-keys-management.sh --delete -regex …**
  - supprime la clé publique de l'hôte distant (regex) de « /home/replication/.ssh/authorized_keys »
- **alcasar-replication-ssh-keys-management.sh --list|-l**
  - montre les clés publiques importées des hôtes distants dans « /home/replication/.ssh/authorized_keys »
- **alcasar-replication-ssh-keys-management.sh –show-pubkey**
  - montre la clé publique locale stockée dans « /root/.ssh/id_rsa.pub »

8.10.3 - Ajout/suppression d'une réplication (à partir d'un secondaire)

- **alcasar-replication-add.sh --to-primary --name=nom_primaire --address=192.168.182.10 --port=22 --user=replication --db-user=db_replication --db-password=(à récupérer sur primaire dans '/root/ALCASAR-passwords.txt', attribut 'db_replication_pwd')**
  - Récupère les infos des réplications actives (alcasar-replication-list.sh –all) et vérifie qu'il n'y a pas doublon ;
  - Ouvre temporairement le port SSH en sortie du parefeu vers le primaire (pour test) ;
  - Teste la connexion SSH vers le primaire avec le compte « replication » ;
  - Copie le fichier « local-db_replication_pwd.txt » à partir du primaire. Extraction du mdp et suppression de la copie ;
  - Teste la connexion sur la BDD du primaire avec le compte « db_replication » + mdp récupéré ;
  - Récupère la liste des ports réseau déjà en écoute sur le primaire, puis localement
  - Récupère la plage des ports éphémères locaux (inutile sur le primaire, car même distrib)
  - Détermine un port éphémère libre commun
  - Crée une unité systemd (/etc/systemd/system/replication-nom_primaire.service) permettant de maintenir un tunnel SSH vers le primaire avec une redirection de port (port_libre local vers port 3386 distant et vice versa)
  - Active et lance cette unité
  - Crée le REPLICA (SQL) : CHANGE MASTER 'primary_name' TO MASTER_HOST='127.0.0.1', MASTER_PORT=$bind_port, MASTER_USER='$remote_db_user', MASTER_PASSWORD='$remote_db_pwd', MASTER_USE_GTID=replica_pos
  - Fixe la clé « REPLICATION_TO=@IP:port » dans « alcasar.conf »
  - Relance le parefeu (prise en compte de cette sortie SSH)
  - Se connecte au primaire et prépare la réplication inverse
  - TODO : finaliser la replication inverse automatique (alcasar-replication-add.sh --to_secondary)

CHECK :

- *Unité systemd de maintien du canal ssh vers le primaire active et enabled (« systemctl status replication-remote_host »)*
- *Tunnel ssh actif : ss -ntaup |grep ssh*
- *« iptable -nvL OUTPUT » avec une entrée ssh vers remote_host*
- *Entrée « REPLICATION_TO » avec le remote_hote:port dans « /usr/local/etc/alcasar.conf »*
- *Replication configurée, mais non active (alcasar-replication-list.sh --all (Slave_IO_running=No et Slave_SQL_Running=No)*

- **alcasar-replication-delete.sh** --name=nom_primaire|all
  - Arrête et supprime le repliqua
  - Arrête, désactive et supprime l'unité systemd de maintien du tunnel SSH
  - retire « @IP:port, » de la clé « REPLICATION_TO= dans « alcasar.conf »
  - relance le parefeu (*prise en compte de ce retrait)*

- **alcasar-replication-start.sh –name=nom_hôte_distant|all**
  - *SQL : START REPLICA $connexion_name (remote_host)*

CHECK : Replication configurée, et active (alcasar-replication-list.sh --all (Slave_IO_running=Yes et Slave_SQL_Running=Yes)

- **alcasar-replication-stop.sh --name=nom_hôte_distant|all**
  - *SQL : STOP REPLICA $connexion_name (remote host)*

8.10.4 - Ajout /suppression d'une réplication à partir du primaire

- **alcasar-replication-add.sh --to-secondary --name=nom_secondaire --bind-port=port_négocié_phase1 –db-user=db_replication --db-password=(à récupérer sur secondaire dans '/root/ALCASAR-passwords.txt', attribut 'db_replication_pwd')**
  - *Crée le REPLICA (SQL) : CHANGE MASTER 'primary_name' TO MASTER_HOST='127.0.0.1', MASTER_PORT=$bind_port, MASTER_USER='$remote_db_user', MASTER_PASSWORD='$remote_db_pwd', MASTER_USE_GTID=replica_pos*

CHECK : Replication configurée, mais non active (alcasar-replication-list.sh --all (Slave_IO_running=No et Slave_SQL_Running=No)

- **alcasar-replication-delete.sh** --name=nom_secondaire|all
  - Arrête et supprime le repliqua

9 - Annexes

Ce chapitre reprend les fichiers de configuration des briques exploitées par ALCASAR.

9.1 - CoovaChilli

Le service « chilli » est lancé par systemd *(/etc/systemd/system/chilli.service)*

- Fichier principal : */etc/chilli.conf*
- Exceptions Domaines : */usr/local/etc/alcasar-uamdomain*
- Exceptions URLs : */usr/local/etc/alcasar-uamallowed*
- L'affectation d'adresses IP statiques s'effectue dans le fichier : */usr/local/etc/alcasar-ethers*

9.2 - Freeradius

Le service « radiusd » est lancé par systemd *(/etc/systemd/system/radiusd.service)*

- Fichier principal : */etc/raddb/radiusd.conf*
- Connexion au SGBD : *sql.conf*
- Clients autorisés à requêter le service (chilli via 127.0.0.1) : */etc/raddb/clients.conf*
- Configuration du client (*/etc/raddb/sites-enabled/alcasar* = lien symbolique vers *sites-enable/alcasar*)
- Liste des attributs (source : https://raw.githubusercontent.com/coova/coova-chilli/master/doc/attributes)

# Name	Type	Comment
User-name	String	Full username as entered by the user.
User-Password	String	Used for UAM as alternative to CHAP-Password and CHAP-Challenge.
CHAP-Password	String	Used for UAM CHAP Authentication
CHAP-Challenge	String	Used for UAM CHAP Authentication
EAP-Message	String	Used for WPA Authentication
NAS-IP-Address	IPaddr	IP address of Chilli (set by the "nasip" or "radiuslisten" option, and otherwise "0.0.0.0")
Service-Type	Integer	Set to Login (1) for normal authentication requests. The Access-Accept message from the radius server for configuration management messages must also be set to Administrative-User.
Framed-IP-Address	IPaddr	IP address of the user, which is configurable during MAC authentication in the Access-Accept.
Framed-IP-Netmask	IPaddr	IP netmask of the user, which is configurable during MAC authentication in the Access-Accept.
Filter-ID	String	Filter ID pass on to scripts possibly.
Reply-Message	String	Reason of reject if present.
State	String	Sent to chilli in Access-Accept or Access-Challenge. Used transparently in subsequent Access-Request.
Class	String	Copied transparently by chilli from Access-Accept to Accounting-Request.
Session-Timeout	Integer	Logout once session timeout is reached (seconds)
Idle-Timeout	Integer	Logout once idle timeout is reached (seconds)
Called-Station-ID	String	Set to the "nasmac" option or the MAC address of chilli.
Calling-Station-ID	String	MAC address of client
NAS-Identifier	String	Set to radiusnasid option if present.
Acct-Status-Type	Integer	1=Start, 2=Stop, 3=Interim-Update
Acct-Input-Octets	Integer	Number of octets received from client.
Acct-Output-Octets	Integer	Number of octets transmitted to client.
Acct-Session-ID	String	Unique ID to link Access-Request and Accounting-Request messages.
Acct-Session-Time	Integer	Session duration in seconds.
Acct-Input-Packets	Integer	Number of packets received from client.
Acct-Output-Packets	Integer	Number of packets transmitted to client.
Acct-Terminate-Cause	Integer	1=User-Request, 2=Lost-Carrier, 4=Idle-Timeout, 5=Session-Timeout, 11=NAS-Reboot
Acct-Input-Gigawords	Integer	Number of times the Acct-Input-Octets counter has wrapped around.
Acct-Output-Gigawords	Integer	Number of times the Acct-Output-Octets counter has wrapped around.
NAS-Port-Type	Integer	19=Wireless-IEEE-802.11
Message-Authenticator	String	Is always included in Access-Request. If present in Access-Accept, Access-Challenge or Access-reject chilli will validate that the Message-Authenticator is correct.
Acct-Interim-Interval	Integer	If present in Access-Accept chilli will generate interim accounting records with the specified interval (seconds).
WISPr-Location-ID	String	Location ID is set to the radiuslocationid option if present. Should be in the format
WISPr-Location-Name	String	Location Name is set to the radiuslocationname option if present. Should be in the format
WISPr-Logoff-URL	String	Included in Access-Request to notify the operator of the log off URL. Defaults to "http
WISPr-Redirection-URL	String	If present the client will be redirected to this URL once authenticated. This URL should include a link to WISPr-Logoff-URL in order to enable the client to log off.
WISPr-Bandwidth-Max-Up	Integer	Maximum transmit rate (b/s). Limits the bandwidth of the connection. Note that this attribute is specified in bits per second.
WISPr-Bandwidth-Max-Down	Integer	Maximum receive rate (b/s). Limits the bandwidth of the connection. Note that this attribute is specified in bits per second.
WISPr-Session-Terminate-Time	String	The time when the user should be disconnected in ISO 8601 format (YYYY-MM-DDThh
CoovaChilli-Max-Input-Octets	Integer	Maximum number of octets the user is allowed to transmit. After this limit has been reached the user will be disconnected.
CoovaChilli-Max-Output-Octets	Integer	Maximum number of octets the user is allowed to receive. After this limit has been reached the user will be disconnected.
CoovaChilli-Max-Total-Octets	Integer	Maximum total octets the user is allowed to send or receive. After this limit has been reached the user will be disconnected.
CoovaChilli-Bandwidth-Max-Up	Integer	Maximum bandwidth up
CoovaChilli-Bandwidth-Max-Down	Integer	Maximum bandwidth down
CoovaChilli-Config	String	Configurations passed between chilli and back-end as name value pairs
CoovaChilli-Lang	String	Language selected in user interface
CoovaChilli-Version	String	Contains the version of the running CoovaChilli
CoovaChilli-DHCP-Netmask	IPaddr	DHCP IP netmask of the user, which is configurable during MAC authentication in the Access-Accept.
CoovaChilli-DHCP-DNS1	IPaddr	DHCP DNS1 of the user, which is configurable during MAC authentication in the Access-Accept.
CoovaChilli-DHCP-DNS2	IPaddr	DHCP DNS2 of the user, which is configurable during MAC authentication in the Access-Accept.
CoovaChilli-DHCP-Gateway	IPaddr	DHCP Gateway of the user, which is configurable during MAC authentication in the Access-Accept.
CoovaChilli-DHCP-Domain	IPaddr	DHCP Domain of the user, which is configurable during MAC authentication in the Access-Accept.
MS-MPPE-Send-Key	String	Used for WPA
MS-MPPE-Recv-Key	String	Used for WPA

9.3 - Unbound

En fonctionnement normal, 4 instances de unbound sont lancées (une instance en mode « forward » sur le port 53, une instance en mode « blacklist » sur port 54, une instance en mode whitelist sur le port 55 et une instance en mode « blackhole » sur le port 56).

- Fichiers principaux : */etc/unbound/unbound.conf* et */etc/unbound/conf.d/**
- les utilisateurs sont redirigés sur une instance de Unbound en fonction de leur attribut de filtrage.

9.4 - Parefeu

- Fichier principal du pare-feu d'ALCASAR : *alcasar-iptables.sh* (sous */usr/local/bin*)
- Règles personnalisées du pare-feu : *alcasar-iptables-local.sh* (sous */usr/local/etc)*
- Activer/désactiver le filtrage web : *alcasar-bl.sh* (sous */usr/local/bin*)
- Fichier listant les catégories de filtrage : *alcasar-bl-categories-enabled* ; utilisée par le fichier *alcasar-bl.sh* pour le filtrage Unbound et E²Guardian.
- Fichier contenant la liste complète des domaines par catégories issues de la liste noire de Toulouse : *alcasar-dnsfilter-available* (sous */usr/local/etc*)
- Fichier du pare-feu d'ALCASAR utilisé en mode ByPass : *alcasar-iptables-bypass.sh* (sous */usr/local/bin*)

9.5 - E²Guardian

Les fichiers d'E²Guardian se situent sous « /etc/e2guardian ».

- Fichier principal de configuration : *e2guardian.conf*
- Fichier concernant le group1 exploité pour les utilisateurs blacklistés : *e2guardianf1.conf*
- Le répertoire « *lists* » contient les fichiers de filtrage et d'exception au filtrage :
  - « *group1/bannedsitelist* » : liste des domain blacklistés. Non exploités, car gérés par les DNS (cf. §6.2)
  - « *group1/exceptionsitelist* » : liste des noms de domaine réhabilités (partagé avec unbound)
  - « *group1/bannediplist* » : liste des IP blacklistés. Non exploité, car gérés par le parefeu (cf. §6.2)
  - « *common/exceptioniplist* » : liste des IP en exception de filtrage
  - « *group1/exceptionurllist* » : liste les URLs réhabilitées (non exploité)
  - « *group1/bannedurllist* » : liste les catégories d'URL à filtrer (issue de la BL de Toulouse)
  - « *blacklists »* BL de Toulouse. Les répertoires « urls » de chaque catégorie alimente le fichier « *group1/bannedurllist* ».

9.6 - Ulogd

Le daemon ulogd centralise les logs du pare-feu (dissociés des logs 'messages') ; tous les journaux d'évènements sont gérés en mode texte.

- Fichier de configuration : *ulogd.conf*
- Fichier concernant les flux SSH entrants : *ulogd-ssh.conf*
- Fichier concernant les flux bloqués en provenance du réseau extérieur : *ulogd-ext-access.conf*

La rotation des logs s'effectue de manière hebdomadaire pour httpd et tracability

9.7 - Distribution Mageia et ses dépôts

La distribution Mageia est utilisée comme système d'exploitation support pour ALCASAR. Les mises à jour et l'installation des paquets s'effectuent à l'aide des outils natifs : « urpmi ».

Les fichiers de configurations se trouvent sous */etc/urpmi* :

- source des miroirs : *urpmi.cfg* ;
- exceptions des mises à jour de paquets : *skip.list* ; permet d'exclure des mises à jour certains paquets pouvant éventuellement troubler le fonctionnement du portail.
- Pour effectuer une mise à jour automatique : *urpmi –auto-update –auto*
- Pour effectuer du ménage : *urpme –auto-orphans –auto*

9.8 - Étude du remplacement de DNSMasq par Unbound

Étude réalisée par Lucas ECHARD

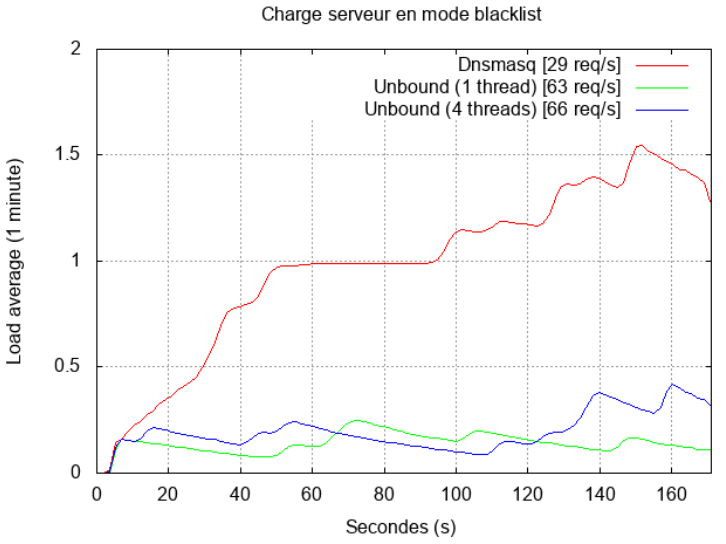
Problématique : À forte charge, le service « Dnsmasq » en mode blacklist consomme beaucoup de ressources et impacte les performances des serveurs ALCASAR.

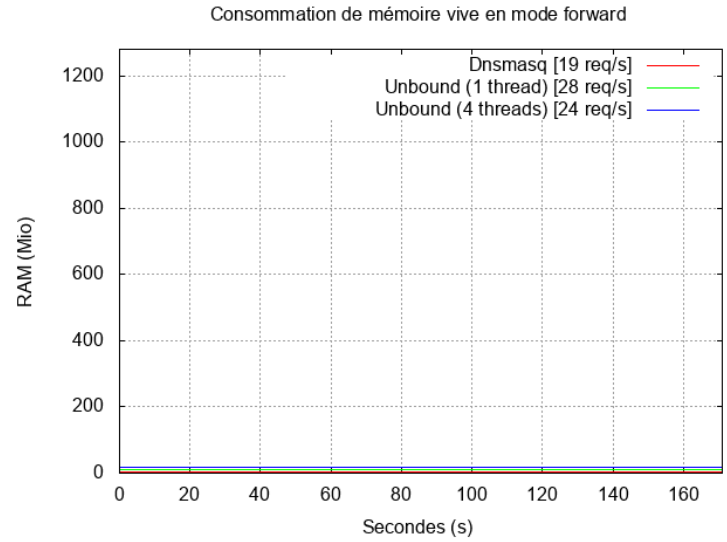
Méthodologie : Un benchmark comparatif a été réalisé entre « Unbound » et « Dnsmasq ». Voici les conditions

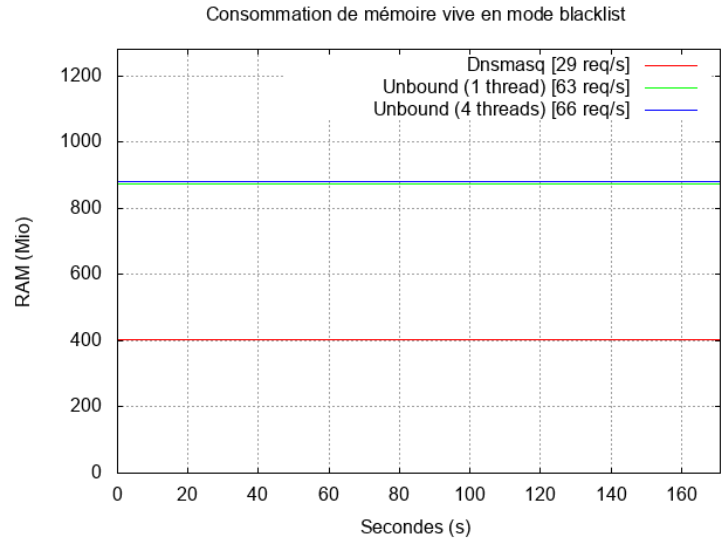
dans lesquelles les tests ont été réalisés :

- utilisation d'une machine virtuelle Mageia 6 (4 cœurs, 8 Gio de RAM) comprenant uniquement Unbound et Dnsmasq ainsi que les fichiers de blacklist/whitelist correspondant ;
- les services dnsmasq, dnsmasq-blacklist, unbound, unbound-blacklist ont été testés indépendamment ;
- une seconde VM faisant office de machine de consultation contenant un script générant des requêtes DNS (50 % whitelist, 50 % blacklist) pour stresser le serveur ;
- le « load average » et la consommation de mémoire vive liée au service testé sont sauvegardés à intervalle régulier sur le serveur.

Résultats : Voici ci-dessous les graphiques obtenus en fonction des données relevées.







Conclusion : comparé à « Dnsmasq », « Unbound » gère mieux la montée en charge au niveau du CPU mais consomme deux fois plus de RAM que « Dnsmasq » en mode blacklist.

Suite à cette étude, il a été décidé de lancer les travaux de migration vers « Unbound » à partir de la version 3.3 (fin 2018).

10 - Test plan

Installation / désinstallation / mise à jour

Function to test	Expected	3.8
Install script (alcasar.sh -i) • disconnect the wire on EXTIF • disconnect the router • Internet off (?) • Set debug mode : change the DEBUG_ALCASAR variable to « on » (line 40) • Install again (new installation)	 • The script stops • The script stops • The script stops • End of installation without error or warning • End of re-installation without error or warning	 ✔ ✔ ✔ ✔ ✔
After first reboot • All services needed are started • No error during the boot process • All ALCASAR services are started	 • “systemctl - - failed” must return 0 line • “journalctl -b |grep 'failed|error' ” • “alcasar-daemon.sh”	 ✔ ✔ ✔
Uninstall script (alcasar.sh -u) • Stop & uninstall all services • Network is working	 • All services are removed (ps fax) • ping Internet	 ✔ ✔
Minor update • Install on a running system (update mode) • verify that old parameters are take into account	 • End of re-installation without error or warning • network parameters • DNS (local & remote resolutions) • security certificates • admin accounts • users database	 ✔ ✔ ✔ ✔ ✔ ✔
Major update • Create a conf file in /var/tmp • Install Linux without formatting /var • Install ALCASAR • verify that old parameters are taken into account	 • alcasar-conf.sh -create (or ACC) • */var*/tmp/alcasar-conf.tar.gz is still present • The conf file is taken into account • (see minor update)	 ✔ ✔ ✔ ✔
Network LAN exposition • “nmap -p1-65536” from LAN • NTP service from LAN	 • TCP 22, 53, 54, 55, 56, 80, 443, 3990, 3991, 8080, 8081 should be open • ntpdate @IP_alcasar	
User management • Create 4 users (“direct”, “bl”, “wl” & “ws”) each-one with a different filtering system (‘ws’ for without “status page”). Set also other attributes to test. • bl test • ws	 The PDF voucher is ok. “ipset list -n“ to see the name of all ipset “ipset list xxx”, list the content of the ipset xxx (xxx = 'not_filtered', 'av_bl', 'av_wl' or 'av') • DNS blacklist test (“www.warez.com” & “www.rael.org”) • URL blacklist test (“wawa-mania.eu” then “wawa-mania.eu/warez”) • IP blacklist test (198.95.10.224 & 207.239.30.20) • URL with IP@ instead of domain name • Safesearch (bing, gg, youtube, ecosia) in “/*tmp*/current_users.txt” the flag “PERM” should	 ✔ ✔ ✔ ✔ X X ✔

Function to test	Expected	3.8
• change a user attribute & test	be set in the @ip line of this user	✔
• create a group “student” with several attribute (filtering, redirect_url, etc). Create a user in this group and verify that the attributes are inherited.	• student group : check this user (with “edit a user”). Two columns should appear. The first one is group attributes and second one is user attributes	✔
• Empty the database ◦ with the script “alcasar-mariadb.sh -raz” ◦ via “ACC”	• the connected users are disconnected. The database is empty.	✔
• Import a users list (simple text file with and without password)	• pdf & txt password file is generated (remove after 24h) • do nothing for users already exist (just change their password if it exists in text file) • To get a SQL database sample. Go to /var/Save/base then use 'gzip -d' to extract.	✔ ✔
• import a users database ◦ with the script “alcasar-mariadb.sh -i xxx.sql” ◦ via ACC		✔ ✔
• Connection to an AD server & test remote users		no test
• Connection to a LDAP server & test remote users		no test
Exceptions		
• Create a user with a PC MAC address as login and 'password' as password • Add domain names and IP addresses in exception form	• The PC should connect to Internet without interception • Users should connect to these IP & domain names without interception	✔ ✔
Watchdog		
• Disconnect the Ethernet cable on EXTIF	• All the DNS requests are send to the DNS blackhole (port 56). “iptables -nvL -tnat” to verify (first line redirect DNS to port 56) • for the users : Internet unavailable page • The main page alert that the access is down	✔ ✔ ✔
• Connect again the Ethernet cable on EXTIF	• Everything return to normal state	✔
Blacklist/Whitelist filtering The 3 following actions are performed when the admin wants to update BL with ACC		
• alcasar-bl.sh -download	Download the last BL from Toulouse (saved in /tmp)	✔
• alcasar-bl.sh -adapt	compute the BL in order to create three sub-BL (domains, URL and IP)	✔
• alcasar-bl.sh -reload	Reload the services that use the BL (unbound-blacklist, unbound-whitelist and netfilter)	✔
• enable all categories to know if there isn’t		

Function to test	Expected	3.8
any bad lines in blacklist files		
• add and remove a 'blacklist' file containing IP and domain names	Verify that the IPs are really in the ipset (command “ipset test bl_ip_blocked @IP”) Verify that the domain names are really blocked. Remove the file	✔ ✔
• ipset list whitelist_ip_allowed • ipset list av_wl	show ip allowed by the whitelist show current user filtered by the whitelist	✔ ✔
• cat /usr/local/share/unbound-wl/ossi.conf • cat /usr/local/share/iptables-wl-enabled/ossi	Show allowed website manually added with ACC the same for IP address	✔ ✔
Network protocols filtering • switch the 1st filter on (http + https) • switch the 2nd filter on • Add a custom port (ex : ssh)	 Check that you can’t connect to a ssh/ftp server Check that you can now connect to a ssh/ftp server Check that you can connect to that port	 ✔ ✔ ✔
Users activities • login & logout with the alcasar main page • logout with the URL : 'http://logout' • Change password • Import the CA certificate		 ✔ ✔ ✔ ✔
SMS (autoregistration) • Check gammu (it’s not a standard service) • Check with « Wavecom » Modem	 /var/log/gammu-smsd/gammu-smsd.log Receive a SMS a create an account	 ✔ ✔
After several days working • “journalctl -b |grep 'failed|error' ” • logrotate process test	 View and analyze errors • logrotate -vf /etc/logrotate.conf (test all file in /etc/logrotate.d) • logrotate -vf /etc/logrotate.d/file_to_test	
Admin • alcasar-bypass.sh -on • alcasar-bypass.sh -off • importation of an official certificate	 Users can surf without interception Users are intercepted	 ✔ ✔
Uninstall • alcasar.sh -u	All services are stopped and the modified config files are removed.	✔
Update a previous version	A conf file (alcasar-conf.tar.gz) is created in /var/tmp The parameters of the previous version are enabled (logo, admin accounts, users database, network parameters, SSL certificates, custom BL&WL).	
MultiWAN	Command : “ip route list” to see the gateways and their weights.	

11 - Security tests

- **« Lynis » internal audit** with the following modifications :
  - INTIF is whitelisted in « default.prf » : *if_promisc:enp0s8:INTIF with Coova (tun0):*
  - Remove « php.ini.default » in « include/test_php »
  - Result : Lynis V3.1.4 - 04/2025 - Hardening index : [68] – No warning
- **Open ports enumeration on Extif** (nmap -p 1-65535 -T4 -v @IP_Extif)
  - Not shown: 65534 filtered ports
  - 22/tcp open ssh OpenSSH 8.0 (protocol 2.0)
- **Audit sshd on Extif**
  - nmap –script-update (all scripts are in /usr/share/nmap/scripts/)
  - nmap -p 22 -- script ssh2-enum-algos,ssh-hostkey,sshv1,ssh-auth-methods,ssh-publickey-acceptance @IP_Extif
  - 22/tcp open ssh
    | ssh-publickey-acceptance:
    |_ Accepted Public Keys: No public keys accepted
    | ssh2-enum-algos:
    | kex_algorithms: (10)
    | curve25519-sha256
    | curve25519-sha256@libssh.org
    | ecdh-sha2-nistp256
    | ecdh-sha2-nistp384
    | ecdh-sha2-nistp521
    | diffie-hellman-group-exchange-sha256
    | diffie-hellman-group14-sha256
    | diffie-hellman-group16-sha512
    | diffie-hellman-group18-sha512
    | kex-strict-s-v00@openssh.com
    | server_host_key_algorithms: (4)
    | rsa-sha2-512
    | rsa-sha2-256
    | ecdsa-sha2-nistp256
    | ssh-ed25519
    | encryption_algorithms: (5)
    | aes256-gcm@openssh.com
    | chacha20-poly1305@openssh.com
    | aes256-ctr
    | aes128-gcm@openssh.com
    | aes128-ctr
    | mac_algorithms: (8)
    | hmac-sha2-256-etm@openssh.com
    | hmac-sha1-etm@openssh.com
    | umac-128-etm@openssh.com
    | hmac-sha2-512-etm@openssh.com
    | hmac-sha2-256
    | hmac-sha1
    | umac-128@openssh.com
    | hmac-sha2-512
    | compression_algorithms: (2)
    | none
    |_ zlib@openssh.com
    | ssh-hostkey:
    | 256 5a:4e:ec:41:e8:13:86:9a:ef:a1:a8:1a:89:c9:9a:06 (ECDSA)
    |_ 256 cf:67:69:3a:6a:75:8d:79:f9:20:9a:5b:5c:64:3a:e1 (ED25519)
    | ssh-auth-methods:
    | Supported authentication methods:
    | publickey
    | password
    | keyboard-interactive

- **Fail2ban**
  - Verify that the 5 ALCASAR jails are started : “fail2ban-client status”

    Status
    Number of jail: 5
    Jail list: alcasar_change-pwd, alcasar_intercept, alcasar_mod-evasive, apache-auth, sshd
  - Verify the jails (tail -f /var/log/fail2ban)
  - Jail 1 with sshd : 5 failed attempts (test with “nmap -p 22 -script ssh-brute @IP_Extif|@IP_Intif”)

    2020-10-17 18:11:57,135 fail2ban.filter [2291]: INFO [sshd] Found 192.168.182.69 - 2020-10-17 18:11:56
    2020-10-17 18:11:57,646 fail2ban.actions [2291]: NOTICE [sshd] **Ban** 192.168.182.69
    2020-10-17 18:14:59,993 fail2ban.actions [2291]: NOTICE [sshd] **Unban** 192.168.182.69
  - Jail 2 with httpd : 5 failed user connections attempts
    2020-10-17 19:12:50,134 fail2ban.filter [2291]: INFO [alcasar_intercept] Found 192.168.182.69 - 2020-10-17 19:12:49
    2020-10-17 19:12:50,646 fail2ban.actions [2291]: NOTICE [alcasar_intercept] **Ban** 192.168.182.69
    2020-10-17 19:15:51,993 fail2ban.actions [2291]: NOTICE [alcasar_intercept] **Unban** 192.168.182.69
  - Jail 3 with httpd : 5 failed user password change attempts
    2020-10-18 16:16:51,766 fail2ban.filter [2291]: INFO [alcasar_change_pwd] Found 192.168.182.69 - 2020-10-18 16:16:50
    2020-10-18 16:16:51,846 fail2ban.actions [2291]: NOTICE [alcasar_change_pwd]] **Ban** 192.168.182.69
    2020-10-18 19:19:50,393 fail2ban.actions [2291]: NOTICE [alcasar_change_pwd] **Unban** 192.168.182.69
  - Jail 4 with httpd : 5 failed ACC connection attempts
    2020-10-18 19:06:13,897 fail2ban.filter [7223]: INFO [alcasar_acc] Found 172.16.0.3 - 2020-10-18 19:06:10
    2020-10-18 19:06:14,632 fail2ban.actions [7223]: NOTICE [alcasar_acc] **Ban** 172.16.0.3
    2020-10-18 19:09:10,915 fail2ban.actions [7223]: NOTICE [alcasar_acc] **Unban** 172.16.0.3

12 - Tests de débit

Voici un exemple de test de bande passante utilisateur (sans filtrage) sur une même fibre (Freebox) à travers 2 ALCASARs 3.5.4 installés sur des PC de génération et de puissance différentes :

- ALCASAR-1 (light) : CPU Intel Atom D2550 1,8G - 4 Go-DDR3 - 2 Gigabit Ethernet BCM57788
- ALCASAR-2 : CPU Xeon E3 3.2G - 8GO-DDR3 - 1 Gigabit Ethernet Intel I212 + 1 Gigabit Ethernet Atheros AR8161

Les tests sont réalisés avec le script « speedtest.py » (https://github.com/sivel/speedtest-cli). Étant donné que les serveurs Internet réceptionnant les tests ne sont pas maîtrisés (on ne connait pas leur charge à un instant T), nous avons décidé de mesurer la perte engendrée par ALCASAR en lançant 2 tests simultanément devant et derrière l'ALCASAR (sur un PC d’utilisateur). Les résultats sont les suivants :

- 3 tests simultanés de débit descendant de chaque côté de l’ALCASAR-1 :
  - ALCASAR-1 : 482.06 Mbit/s. PC utilisateur : 411.09 Mbit/s (14%)
  - ALCASAR-1 : 430.47 Mbit/s. PC : 417.17 Mbit/s (3%)
  - ALCASAR-1 : 482.48 Mbit/s. PC : 442.03 Mbit/s (8%)
- 3 tests simultanés de débit descendant de chaque côté de l’ALCASAR-2
  - ALCASAR-2 : 485.23 Mbit/s. PC : 465.08 Mbit/s (4%)
  - ALCASAR-2 : 447.56 Mbit/s. PC : 447.38 Mbit/s (0%)
  - ALCASAR-2 : 517.78 Mbit/s. PC : 454.55 Mbit/s (12%)

En moyenne, la perte engendrée par ALCASAR sur le débit descendant est inférieure à 10 %.

13 - Arbre de dépendance JS et CSS

13.1 - JS

/var/www/html/js/

```
-rw-r--r-- 1 apache apache  62440 févr. 21 17:47 bootstrap-4.6.1.min.js
lrwxrwxrwx 1 apache apache     22 févr. 21 17:47 bootstrap.min.js -> bootstrap-4.6.1.min.js
-rw-r--r-- 1 apache apache 201572 févr. 21 17:47 Chart.bundle.min.js
-rw-r--r-- 1 apache apache  26625 févr. 21 17:47 ChilliLibrary.js
-rw-r--r-- 1 apache apache  34363 févr. 21 17:47 epoch_classes.js
-rw-r--r-- 1 apache apache   3087 févr. 21 17:47 fonctions.js
-rw-r--r-- 1 apache apache  67015 févr. 21 17:47 gstatic-loader.js
-rw-r--r-- 1 apache apache 255084 févr. 21 17:47 jquery-1.13.2-ui.min.js
-rw-r--r-- 1 apache apache  72818 févr. 21 17:47 jquery-3.6.3.slim.min.js
-rw-r--r-- 1 apache apache  89501 févr. 21 17:47 jquery-3.6.min.js
-rw-r--r-- 1 apache apache   6283 févr. 21 17:47 jquery.connections.js
-rw-r--r-- 1 apache apache 422284 févr. 21 17:47 jquery.dataTables.js
lrwxrwxrwx 1 apache apache     17 févr. 21 17:47 jquery.min.js -> jquery-3.6.min.js
lrwxrwxrwx 1 apache apache     24 févr. 21 17:47 jquery.slim.min.js -> jquery-3.6.3.slim.min.js
lrwxrwxrwx 1 apache apache     23 févr. 21 17:47 jquery-ui.min.js -> jquery-1.13.2-ui.min.js
-rw-r--r-- 1 apache apache   3139 févr. 21 17:47 login-time.js
-rw-r--r-- 1 apache apache  15405 févr. 21 17:47 pwdmeter.js
-rw-r--r-- 1 apache apache  11398 févr. 21 17:47 schedule.js
-rw-r--r-- 1 apache apache   6753 févr. 21 17:47 statusControler.js
```

Libraries	Use by
bootstrap.min.js	web/email_registration_front.php web/acc/backup/log_generation.php web/acc/manager/htdocs/security.php
Chart.bundle.min.js	scripts/alcasar-activity-report.sh
ChilliLibrary.js	web/status.php
epoch_classes.js fonction.js login-time.js schedule.js	web/acc/manager/htdocs/group_new.php web/acc/manager/htdocs/user_edit.php web/acc/manager/htdocs/user_new.php
gstatic-loader.js	
jquery-ui.min.js	web/acc/manager/htdocs/group_new.php web/acc/manager/htdocs/user_edit.php web/acc/manager/htdocs/user_new.php
jquery.min.js	scripts/alcasar-activity-report.sh web/email_registration_front.php web/sms_registration.php web/acc/menu.php web/acc/admin/network.php web/acc/admin/services.php web/acc/backup/log_generation.php web/acc/manager/htdocs/group_new.php web/acc/manager/htdocs/security.php web/acc/manager/htdocs/user_by_sms.php web/acc/manager/htdocs/user_edit.php web/acc/manager/htdocs/user_new.php
jquery.connections.js	web/acc/admin/network.php
jquery.dataTables.js	web/sms_registration.php web/acc/manager/htdocs/user_by_sms.php
pwdmeter.js	web/password.php
statusControler.js	web/status.php

13.2 - CSS

/var/www/html/css/

```
-rw-r--r-- 1 apache apache   4272 févr. 21 17:47 acc.css
-rw-r--r-- 1 apache apache 161996 févr. 21 17:47 bootstrap-4.6.1.min.css
lrwxrwxrwx 1 apache apache     23 févr. 21 17:47 bootstrap.min.css -> bootstrap-4.6.1.min.css
-rw-r--r-- 1 apache apache   1873 févr. 21 17:47 epoch_styles.css
-rw-r--r-- 1 apache apache  12805 févr. 21 17:47 error.css
-rw-r--r-- 1 apache apache   4745 févr. 21 17:47 index.css
-rw-r--r-- 1 apache apache   3535 févr. 21 17:47 intercept.css
-rw-r--r-- 1 apache apache  32130 févr. 21 17:47 jquery-1.13.2-ui.min.css
-rw-r--r-- 1 apache apache  15111 févr. 21 17:47 jquery.dataTables.css
lrwxrwxrwx 1 apache apache     24 févr. 21 17:47 jquery-ui.min.css -> jquery-1.13.2-ui.min.css
-rw-r--r-- 1 apache apache   2151 févr. 21 17:47 ldap.css
-rw-r--r-- 1 apache apache    805 févr. 21 17:47 menu.css
-rwxr-xr-x 1 apache apache   3887 févr. 21 17:47 pass.css
-rwxr-xr-x 1 apache apache   5833 févr. 21 17:47 pwdmeter.css
-rw-r--r-- 1 apache apache    378 févr. 21 17:47 report.css
-rw-r--r-- 1 apache apache   3248 févr. 21 17:47 status.css
```

CSS files	Use by
acc.css	web/acc/admin_log.php
	web/acc/haut.php
	web/acc/index.html
	web/acc/menu.php
	web/acc/welcome.php
	web/acc/admin/bl_categories_help.php
	web/acc/admin/bl_filter.php
	web/acc/admin/ldap.php
	web/acc/admin/logo.php
	web/acc/admin/network.php
	web/acc/admin/protocols_filter.php
	web/acc/admin/services.php
	web/acc/admin/wl_filter.php
	web/acc/admin/backup/log_generaton.php
	web/acc/admin/backup/sauvegarde.php
	web/acc/manager/auth_exceptions.php
	web/acc/manager/nfsen.php
	web/acc/manager/vnstat.php
	web/acc/manager/htdocs/activity.php
	web/acc/manager/htdocs/badusers.php
	web/acc/manager/htdocs/clear_opensessions.php
	web/acc/manager/htdocs/failed_logins.php
	web/acc/manager/htdocs/find.php
	web/acc/manager/htdocs/group_admin.php
	web/acc/manager/htdocs/group_new.php
	web/acc/manager/htdocs/import_user.php
	web/acc/manager/htdocs/security.php
	web/acc/manager/htdocs/show_groups.php
	web/acc/manager/htdocs/stats.php
	web/acc/manager/htdocs/user_accounting.php
	web/acc/manager/htdocs/user_admin.php
	web/acc/manager/htdocs/user_by_email.php
	web/acc/manager/htdocs/user_by_sms.php
	web/acc/manager/htdocs/user_delete.php
	web/acc/manager/htdocs/user_edit.php
	web/acc/manager/htdocs/user_finger.php
	web/acc/manager/htdocs/user_new.php

	web/acc/manager/htdocs/user_stats.php web/acc/manager/htdocs/help/coovachilli_bandwith_max_down _help.html web/acc/manager/htdocs/help/coovachilli_bandwith_max_up_help.html web/acc/manager/htdocs/help/coovachilli_max_input_octets_help.html web/acc/manager/htdocs/help/coovachilli_max_output_octets_help.html web/acc/manager/htdocs/help/coovachilli_max_total_octets_help.html web/acc/manager/htdocs/help/expiration_help.html web/acc/manager/htdocs/help/expire_after_help.html web/acc/manager/htdocs/help/filtering_help.html web/acc/manager/htdocs/help/login_time_help.html web/acc/manager/htdocs/help/max_all_session_help.html web/acc/manager/htdocs/help/protocols_help.html web/acc/manager/htdocs/help/session_timeout_help.html web/acc/manager/htdocs/help/simultaneous_use_help.html web/acc/manager/htdocs/help/statusOpenRequired_help.html web/acc/manager/htdocs/help/wispr_redirection_url_help.html
bootstrap.min.css	conf/alcasar-e2g-en.html conf/alcasar-e2g-fr.html scripts/alcasar-activity_report.sh scripts/alcasar-generate_log.sh web/email_registration_front.php web/index.php web/intercept.php web/password.php web/sms_registration.php web/status.php web/acc/backup/log_generation.php web/acc/manager/htdocs/security.php web/acc/manager/vnstat/templates/module_header.tpl web/acc/phpsysinfo/templates/html/index_bootstrap.html
epoch_styles.css	web/acc/manager/htdocs/group_new.php web/acc/manager/htdocs/user_edit.php web/acc/manager/htdocs/user_new.php
error.css	web/error.php
index.css	conf/alcasar-e2g-en.html conf/alcasar-e2g-fr.html web/sms_registration.php
intercept.css	web/intercept.php
jquery-ui.min.css	web/acc/manager/htdocs/group_new.php web/acc/manager/htdocs/user_edit.php web/acc/manager/htdocs/user_new.php
jquery.dataTables.css	web/sms_registration.php web/acc/manager/htdocs/user_by_sms.php
ldap.css	web/acc/admin/ldap.php
menu.css	web/acc/menu.php
pass.css	web/email_registration_front.php web/password.php
pwdmeter.css	web/password.php
report.css	scripts/alcasar-activity_report.sh scripts/alcasar-generate_log.sh
status.css	web/email_registration_front.php web/status.php

14 - TODO List

blacklists/whitelists : homogénéiser la gestion des listes personnalisées	L'architecture de gestion des blacklists personnelles doit être revue et homogénéisée. Pour rappel, cette architecture permet d'importer, de supprimer, d'activer, de désactiver et de synchroniser une blacklist personnelle qui complète la blacklist de Toulouse.
ACC : simplifier le code des modules de gestion des utilisateurs et des groupes	Une partie de ce code est hérité du projet « freeradius-web » qui permet de gérer une base de données connectée à un serveur freeradius. Ce code avait été prévu modulaire afin de s'adapter à plusieurs cas de figure (pgsql, mysql, etc.). Sur ALCASAR, les modules non exploités ont été supprimés. Cependant, l'architecture du code reste basée sur cette gestion de modules, ce qui le complexifie inutilement.
Réétudier le module d'inscription par SMS	Le module d'inscription par SMS doit être mis à jour après avoir testé de nouveaux modem compatibles 3G/4G/5G (suppression de la 2G 900/1800Mhz à moyen terme). Modem actuel : Ostend (avec module wavecom Q2303A-RS232 ou Q2403A-USB). • Idée 1 : tester le modem Ostend (avec module wavecom Q24plus), adaptateur USB-4G LTE (avec AP WIFI), adaptateur USB LTE, etc. • Idée 2 : exploiter un canal de connexion USB vers un GSM ◦ « termux + termux-api + openssh » sur Android, ssh sur Linux. ◦ « termux-sms-send -n "+33612345678" "Bonjour depuis Termux !").
E²guardian : Intégrer le blocage d'URL contenant une @IP	Le filtrage d'URL pour les utilisateurs filtrés par blacklist est géré par E²guardian. Ce filtrage peut être amélioré en bloquant les URL contenant une @IP à la place d'un nom de domaine (ex: http://25.56.58.59/erp/about.htm). Il faut réactiver/revalider cette option de l'ACC.
Groupe « default » pour les utilisateurs	Par défaut, tous les utilisateurs appartiennent à un groupe nommé « default ». Ce groupe n'est pas créé par défaut. Adapter l'affichage de l'ACC pour que les attributs du groupe soient affichés à côté des attributs utilisateurs. Question : serait-il judicieux de créer ce groupe lors de l'installation ?
Remplacement de Wkhtml2pdf par TCPDF	La génération des pages pdf exploite une très vieille et lourde version de wkhtml2pdf packagé pour ALCASAR. La migration vers TCPDF permettra de s'appuyer sur des RPM natif Mageia.
E²Gardian : filtrage HTTPS avec et sans inspection SSL/TLS	Étudier les possibilités de filtrage sans inspection SSL/TLS • validité des certificats ; • CRL ; • inspection du champ SNI (si non chiffré par l'option ESNI de TLS1.3) ; • autres. Étudier les possibilités de filtrage avec inspection SSL/TLS
ACC : limitation d'accès par @IP	Ajouter dans l'ACC la possibilité pour l'administrateur de limiter l'accès à l'ACC à une liste d'@IP ou @IP de réseau.
Fédération d'ALCASAR	**Intégré dans la version 3.8.0.**
Cybersurveillance	Étudier l'ajout d'un agent de Cybersurveillance (Wazuh).
Multi-WAN : amélioration et intégration de la fonction « HA/failover »	Le module multi-WAN permet d'équilibrer la charge de navigation sur N routeurs de sorties. Le module réalise un équilibrage par utilisateur. Ce module fonctionne, mais présente une bizarrerie qu'il faut investiguer : - constat : erreurs dans la résolution DNS sur les postes utilisateurs. Piste de réflexion : vers quel routeur sont envoyées les requêtes générées par ALCASAR (certaines Box refusent les requêtes DNS vers d'autres serveurs que les leurs). Le développement de la fonction « HA/failover » (analyse de la qualité de liaison sur chaque routeur) permettrait : • une alerte dans l'ACC • le rééquilibrage dynamique de la charge
Multi-LAN	ALCASAR contrôle plusieurs réseaux de consultation (VLAN et/ou plusieurs

	cartes Ethernet et/ou une carte WIFI en A.P.)
Évolution de l'attribut 'Alcasar-Status-Page-Must-Stay-Open'	L'attribut radius 'Alcasar-Status-Page-Must-Stay-Open' étant un entier, on gère sa valeur comme un nombre d'heures où l'utilisateur peut rester connecter (sans obligation d'avoir sa fenêtre de statut ouverte). • Lors de la connexion d'un utilisateur possédant l'attribut 'Alcasar-Status-Page-Must-Stay-Open', « alcasar-conup.sh » crée un nouvel utilisateur de type « utilisateur de confiance » avec les attributs suivants : ◦ « login » = @MAC récupérée ; ◦ « date d'expiration » = date d'expiration de l'utilisateur (ou date actuelle + « Alcasar-Status-Page-Must-Stay-Open » si inférieure à la date d'expiration de l'utilisateur) ; ◦ nom&prénom = login_de_l'utilisateur (pour la traçabilité).
Email administrateur	Depuis ALCASAR 3.5, un module Email a été intégré (utilisé pour l'auto-inscription d'utilisateur par Email). L'idée est d'utiliser ce module pour créer l'Email de l'administrateur afin d'offrir de nouvelle possibilité (envoi des rapports, envoi des fichiers de log, envoi d'alertes, etc.)
Étude de la sécurité de l'UAM	ALCASAR exploite la méthode UAM (Universal Access Method) pour authentifier et autoriser les utilisateurs. Cette méthode permet de s'affranchir d'un agent radius installé dans les équipements utilisateurs, ce qui serait ingérable pour les cas d'usage d'ALCASAR. Ainsi, l'agent est remplacé par un cycle d'échange entre un serveur WEB, le navigateur de l'utilisateur et un agent radius (« coova » pour ALCASAR). L'étude de la sécurité de ce cycle permettra de vérifier si la sécurité mise en place est suffisante et/ou si elle peut être améliorée. **1 POC a été réalisé. Il faut le tester et l'intégrer le cas échéant.**
802.1x sur le LAN	Tester et créer une documentation expliquant comment configurer ALCASAR dans le cas où il serait exploité comme serveur d'authentification 802.1x sur le LAN (Coova en « proxy radius »). Voir si cette configuration peut être intégrer dans l'ACC.
Créer des profils de blacklist/whitelist	E²Guardian est capable de gérer des profils de filtrage par blacklist. Il peut être intéressant d'étudier la possibilité d'exploiter cette capacité sur ALCASAR.
Charte informatique	- Ajout d'un attribut utilisateur : charte=on/off - si 'on' : l'utilisateur doit cocher une case 'charte lue' dans intercept.php. Une fois cochée : charte=off.