Analyse du site de la RTBF

8082-1600

***Résumé*—Ce document reprend une analyse du site web de la rtbf - rtbf.be - sous quatre différents points : le Domaine Name System (DNS), les requêtes HTTP, le Transport Layer Security (TLS) et le Transmission Control Protocol (TCP).**

I. Introduction

Ce document aborde l'analyse de protocoles connus sur internet, à savoir, DNS, HTTP, TLS et TCP. Il est divisé en quatre sections analysant chacun des protocoles pour le site web rtbf.be, qui traite des informations d'actualité. Il aborde de manière complète les résultats de l'étude du site, tout en pointant les différentes anomalies ou spécificités rencontrées. Une analyse rapide du site web de la RTBF permet de mettre en avant le fait que rtbf.be redirige directement vers www.rtbf.be. En effet, le statut de la requête HTTP vers rtbf.be est *301 Moved Permanently*. Cette réponse signifie que les ressources demandées ont été assignées à un nouvel URI de manière permanente [1].

II. Domain Name System

Le DNS est un protocole faisant partie de l'ensemble des normes relatives à la façon dont les ordinateurs échangent des données sur Internet et sur de nombreux réseaux privés. Son travail de base est de traduire un nom de domaine en une adresse Internet Protocol (IP) que les ordinateurs utilisent pour s'identifier mutuellement sur le réseau [2].

A. Adresses IP

Seul le domaine www.rtbf.be est accessible via des adresses IPv6 (AAAA records). Les adresses IP ainsi que leur version sont reprises dans le Tableau I. Nous pouvons remarquer que le nom de domaine rtbf.be utilise donc le load balancing qui permet de distribuer une charge de travail entre différents serveurs. Cette technique permet la stabilité du service proposé par la RTBF. Le choix de faire du load balancing sur le domaine rtbf.be et non sur www.rtbf.be est expliqué dans la Section II-F.

B. Canonical Name record

Un Canonical Name record (CNAME) est un type d'enregistrement qui lie un nom de domaine à un autre, désigné sous le Canonical Name. Le domaine www.rtbf.be est un CNAME pour prdfast.rtbf.be qui en est lui-même un pour http2.infomaniak.map.fastly.net. Ce dernier redirige vers les deux adresses IPv4 et IPv6 mentionnées dans le Tableau I.

C. Hébergeur

La consultation du serveur whois.arin.net à l'aide de la commande whois nous permet de conclure que l'hébergeur est Fastly qui se trouve à San Fransico. Cet hébergeur utilise un service de Content Delivery Network (CDN). Un CDN a des points de présence (PoPs) ou des centres de données qui sont situés dans le monde entier. Les données d'un PoP sont copiées dans un autre PoP situé loin du premier pour que les données demandées soient accessibles via plusieurs PoPs répartis dans le monde. L'information passe donc par moins de routeurs [3].

D. Authoritative name server

L'authoritative name server (NS) spécifie un nom d'hôte où l'on pourra trouver les informations DNS à propos du domaine auquel l'enregistrement NS est rattaché. Ceux relatifs aux domaines analysés sont repris dans le Tableau I.

E. Time To Live

Le Time To Live (TTL) est un mécanisme qui limite la durée de vie des données sur un ordinateur ou dans un réseau. Une fois le nombre d'événements atteints, les données sont supprimées. Dans les réseaux informatiques, le TTL empêche un paquet de circuler indéfiniment. Le Tableau I reprend les TTL des domaines rencontrés.

F. Hébergement mutualisé

L'analyse des adresses IP avec l'outil IPinfo.io a pu mettre en évidence que le nom de domaine rtbf.be fonctionnait sur un hébergement mutualisé. Ce service permet que les hôtes virtuels servent plusieurs noms d'hôtes sur une seule machine avec une seule adresse IP. Cette technique est réalisable parce que lorsqu'un navigateur demande une ressource à un serveur en utilisant HTTP/1.1, il inclut le nom d'hôte demandé dans la requête. Le serveur sait donc chez quel hôte virtuel il doit prendre la ressource demandée. L'hébergeur a donc mis en place un load balancing au niveau du domaine rtbf.be parce que plusieurs autres noms de domaine - de sites ayant un flux de visiteurs moins important - utilisent les mêmes adresses IP.

III. Requêtes HTTP

Lors du chargement de la page www.rtbf.be, le navigateur [1] envoie environ entre 100 et 250 requêtes à 41 domaines différents dont les cinq plus sollicités sont ds1.static.rtbf.be, sgc.static.rtbf.be, js.static.bda.rtbf.be, www.static.rtbf.be et connect.facebook.net. Les quatre premiers domaines sont des domaines contenant des liens statiques vers des ressources.

1. Tous les tests concernant la Section III ont été réalisés conjointement avec les navigateurs Chrome et Safari.

Name	TTL	Type	IP/CNAME
www.rtbf.be	900	CNAME	prdfast.rtbf.be
prdfast.rtbf.be	300	CNAME	http2.infomaniak.map.fastly.net
http2.infomaniak.map.fastly.net	30	AAAA	2a04 :4e42 :9 : :319
http2.infomaniak.map.fastly.net	30	A	151.101.37.63
rtbf.be	300	A	151.101.1.63
	300	A	151.101.65.63
	300	A	151.101.129.63
	300	A	151.101.193.63

Authoritative name server of www.rtbf.be			
Name	**TTL**	**Type**	**IP**
ns1.fastly.net	3600	A	23.235.32.32
ns2.fastly.net	3600	A	104.156.80.32
ns3.fastly.net	3600	A	23.235.36.32
ns4.fastly.net	3600	A	104.156.84.32

Authoritative name server of rtbf.be			
Name	**TTL**	**Type**	**IP**
ns.rtbf.be	300	A	185.153.40.30
ns1.rtbf.be	300	A	185.153.40.30
ns2.rtbf.be	300	A	185.153.42.30
ns3.rtbf.be	3600	A	185.153.41.252
ns1.belnet.be	3600	A	193.190.198.14
	3600	AAAA	2001 :6a8 :3c80 : :14
ns2.belnet.be	3600	A	193.190.182.40
	3600	AAAA	2001 :6a8 :3c80 :c000 : :40
ns3.belnet.be	3600	A	145.0.7.163
	3600	AAAA	2001 :610 :188 :441 :145 : :7 :163

Tableau I
DIG ANALYSIS

On y stocke notamment des images, des scripts en Javascript, etc. Le nombre de requêtes chargées dépend de la taille de la fenêtre du navigateur. En effet, le site de la RTBF charge seulement les ressources qu'il doit afficher à l'écran et les ressources qui permettent d'analyser le comportement de l'utilisateur. Toutes les autres ressources sont chargées au fur et à mesure que l'utilisateur navigue sur la page. Le navigateur charge environ 1.4 Mb (∼120 requêtes) sur un smartphone et 3 Mb (∼200 requêtes) sur un ordinateur. Cette technique permet un chargement plus rapide des pages. Sur la Image 1, nous pouvons observer la répartition des requêtes HTTP. La majorité d'entre elles sont effectuées avec le protocole HTTP/1.1 contre seulement 35% avec le protocole HTTP/2.

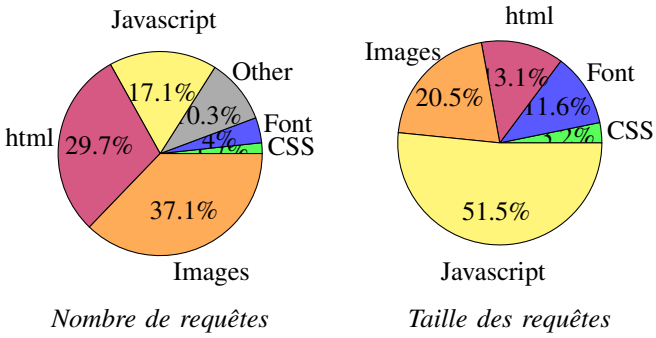

Image 1. Répartition des requêtes

A. Type de requêtes

Toutes les requêtes sont protégées par un certificat SSL/TLS. Si nous faisons une requête vers le port 80 (HTTP) de www.rtbf.be, nous sommes automatiquement redirigés vers le port 443 (HTTPS) car la requête nous renvoie un status *301 Moved Permanently*. Toutes les requêtes faites par le navigateur à la connexion du site web sont des requêtes `GET`, seules trois requêtes font exception. En effet, le navigateur fait une requête `POST` vers un sous-domaine de demdex.net [2] appartenant à la RMB [3] contenant un timestamp et un identifiant propre à chaque navigateur et utilisateur. Ce dernier fait également deux requêtes `HEAD` identiques.

B. En-têtes des requêtes

Au vu du nombre de requêtes, nous nous intéresserons seulement à la requête principale ainsi qu'aux en-têtes non-standards de quelques requêtes. Nous pouvons analyser les en-têtes non-standards de la requête `GET www.rtbf.be`.

Response headers	
status	200
content-security-policy	upgrade-insecure-requests
x-ua-compatible	IE=edge
x-fastly-ttl	120.000
x-fastly-clientip	2a02 :a03f :3a1d :8900 :58c0 :84ff :c4cc :bac9
x-fastly-cacheable	YES
x-fastly-cache	HIT
x-fastly-cache-hits	7
x-served-by	cache-ams21033-AMS
x-cache	HIT
x-cache-hits	7
x-timer	S1544701421.445524,VS0,VE0

Nous pouvons remarquer plusieurs champs x-fastly. Ces champs correspondent à la mise en cache d'informations dans un PoP dont le principe est expliqué à la section II-C. Ils sont spécifiques à Fastly et son service 'Shielding' [4]. Ce dernier nous permet de désigner un point de présence spécifique (PoP) comme noeud de bouclier (appelé PoP shield ci-après) vers le serveur d'origine. Toutes les demandes au serveur passent par le PoP qui a été désigné. Cela augmente la chance d'avoir une ressource en cache. Si un PoP différent n'a pas la ressource, il interrogera le bouclier au lieu des serveurs. Par exemple, prenons un PoP A, un PoP B et une nouvelle ressource demandée via le PoP A. Ce dernier va interroger le PoP shield qui transmettra directement au serveur. La ressource est mise en cache dans le PoP A et dans le PoP shield. Si le PoP A est interrogé avec une même requête, la mise en cache permet à ce dernier de directement répondre. Mais dans le cas où la ressource est demandée via le PoP B, il transmet au PoP shield qui la renvoie directement depuis son cache sans solliciter le serveur. Définissons plus précisément chaque en-tête.

x-fastly-ttl Il exprime le temps pendant lequel la ressource va rester en cache dans un PoP.

2. Cette société propose un logiciel d'optimisation d'audience.
3. La régie média belge (RMB) commercialise l'espace publicitaire des médias télévision, radio et des nouveaux médias.

x-fastly-clientip L'adresse IPv4 ou IPv6 (si disponible) du client demandant la ressource.

x-fastly-cacheable YES/NO si l'entreprise a configuré ou non le 'Shielding'.

x-fastly-cache HIT si la ressource est présente en cache, MISS sinon [5]. Idem pour x-cache.

x-fastly-cache-hits Il fournit le nombre de fois que la ressource a été récupérée dans le PoP par un même client. Idem pour x-cache-hits.

x-served-by Il donne le PoP dans lequel la ressource a été récupérée.

x-time Il fournit des informations temporelles sur le parcours d'une demande de bout en bout [6].

status Il exprime le statut de la requête.

x-ua-compatible Il recommande le client préféré pour afficher le contenu. Cette en-tête est beaucoup utilisée pour Internet Explorer IE=Edge.

content-security-policy Le Content Security Policy (CSP) est une norme de sécurité informatique introduite pour prévenir les scripts inter-sites (XSS), les détournements de clics et autres attaques par injection de code résultant de l'exécution de contenu malveillant [7].

C. Alternative protocol

1) QUIC: Dans toutes les requêtes vers Google, nous retrouvons l'en-tête *alt-svc* qui a comme valeur *quic=":443"; ma=2592000; v="44,43,39,35"*. QUIC (Quick UDP Internet Connections) est un protocole développé par Google depuis 2013. Il est une alternative aux solutions TCP, HTTP/2 et TLS/SSL mais il permet un délai de connexion et de transport réduit, et des connexions de multiplexage [8]. Depuis 2016, un groupe de travail officiel de l'IEFT travaille sur l'optimisation du protocole.

2) Websocket: On peut remarquer qu'une des requêtes renvoie un statut *101 Switching Protocols*. Ce dernier signifie que le protocole a changé. Dans notre cas, nous passons de HTTPS à Websocket. Ce changement a été demandé via l'en-tête de demande *upgrade* qui a comme valeur *websocket*. WebSocket est un protocole de communication informatique qui fournit des canaux de communication full-duplex sur une seule connexion TCP [9]. Il a été normalisé par l'IETF en 2011. Ce protocole permet l'envoi de données de façon bidirectionnelle entre le client et le serveur. Il permet notamment de mettre en place des systèmes de notifications via des méthodes Push.

D. Compression

Une grande partie du contenu transféré est compressée. En comparant la taille du téléchargement avec la taille du fichier, nous pouvons mettre en évidence que compresser les ressources permet un gain de 83% en passant de 13.6 Mb à 2.3 Mb. La compression peut se faire parce que le navigateur l'autorise dans l'en-tête *accept-encoding*. Il existe plusieurs types de compression mais les plus utilisées sont *gzip* et *deflate*. Nous verrons dans la Section IV-B ce que cela implique au niveau de la sécurité.

E. Cache

Lorsque le cache est activé, une grande partie du contenu statique n'est pas téléchargée à nouveau. La taille des ressources transférées passe donc de 2.7 Mb à 490 kb. Les vitesses de chargement sont impactées passant de 7 sec à 2 sec en moyenne. Beaucoup de pages sont simplement extraites du cache sans que des requêtes soient faites vers le serveur. Cette opération est possible via l'en-tête *cache-control* qui spécifie si une ressource peut ou ne peut pas être mise en cache et, dans le cas où elle le peut, quelle est la durée de la mise en cache.

F. Varnish

Dans toutes les réponses venant du serveur de fastly, nous retrouvons l'en-tête *via* à la valeur *1.1 varnish-v4 1.1 varnish*. Varnish est un serveur cache qui permet de gérer les règles de fonctionnement de la mise en cache en utilisant le langage de script VCL [10].

IV. Transport Layer Security

Le site web de la RTBF utilise des certificats TLSv1.2. Il ne supporte ni les autres versions de ces protocoles, ni SSLv2 et SSLv3 qui sont obsolètes selon l'IETF.

A. Certificat

Le certificat est distribué par GlobalSign. Il utilise un certificat intermédiaire GlobalSign CloudSSL CA - SHA256 - G3. Lors du handshake, la suite Cipher choisie est ECDHE_RSA_WITH_AES_128_GCM_SHA256[4]. Cette suite utilise la fonction de hashage SHA-256 et l'algorithme de cryptage RSA. Pour des raisons de sécurité, le serveur n'accepte ni la compression des demandes, ni les extensions au niveau du protocole. Cependant, il accepte les re-négociations.

B. BREACH Attack

Le site est vulnérable à une attaque BREACH. Cette dernière permet à l'attaquant d'amener le navigateur d'un utilisateur à effectuer une action non désirée sur un site de confiance, lorsque l'utilisateur est authentifié [11]. La vulnérabilité du site web est due au fait que le serveur accepte l'en-tête de requête *accept-encoding*. En effet, après avoir ouvert une connexion TLS avec la commande *openssl s_client -connect www.rtbf.be :443*, nous pouvons faire une requête en demandant la compression de la réponse.

```
HEAD / HTTP/1.1
host: www.rtbf.be
accept-encoding: gzip, deflate, compress
```

Nous retrouvons bien l'en-tête *content-encoding* à la valeur *gzip*. Nous pouvons donc forcer le serveur à nous renvoyer des requêtes compressées. Il existe plusieurs manières d'atténuer la vulnérabilité à cette attaque, par exemple, en limitant le nombre de requêtes venant d'une même adresse IP.

4. Ce test est effectué via la commande openssl d'un terminal MacOS.

C. Suites Cipher

Le serveur accepte plusieurs types de suite Cipher différentes à savoir (de la plus à la moins sécurisée)

ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE_RSA_WITH_AES_256_CBC_SHA
RSA_WITH_AES_128_GCM_SHA256
RSA_WITH_AES_128_CBC_SHA
RSA_WITH_AES_256_CBC_SHA

Les trois dernières suites Cipher sont vulnérables aux attaques à cause de nombreuses failles de sécurité. Heureusement, la grande majorité des navigateurs utilise la suite Cipher ECDHE_RSA_WITH_AES_128_GCM_SHA256[5] qui ne présente a priori pas de failles de sécurité.

V. Transmission Control Protocol

L'analyse des transmissions TCP entre le serveur et le client a été réalisée à l'aide du logiciel Wireshark.

A. Three-way handshake

La première connexion avec le serveur se fait comme suit.

1) [SYN] On demande une window size de 65535, un Maximum Segment Size (MSS) de 1428 bytes, un Window Scale (WS) de 6 (multiplié par 64) et on autorise le Selective Acknowledgments (SACK).
2) [SYN, ACK] Le serveur nous répond avec une window size de 27200, un MSS de 1372 bytes, un WS de 9 (multiplié par 512) et il autorise le SACK.
3) [ACK] Le client envoie alors son ACK avec comme numéro de séquence 1. Ce paquet marque la fin du three-way handshake.

B. No-Operation

Lors de l'analyse des différents paquets, on peut apercevoir une option qui se nomme No-Operation. La RFC791 nous permet de mieux comprendre son fonctionnement. Elle est utilisée afin de séparer des options entre elles. Par exemple, elle sert à aligner le début d'une option suivante sur une limite 32 bits [12].

C. Keep-alive

La connexion avec le serveur est de type keep-alive, c'est à dire que le serveur ne ferme pas la connexion TCP une fois le contenu transféré. Après dix minutes sans échange, la connexion se termine à l'aide d'un segment [FIN]. Ce comportement, a priori bizarre, s'explique par le fait que toutes les ressources statiques sont chargées au fur et à mesure que l'utilisateur navigue sur la page (cf. Section III) et par le fait que le site propose des articles sur l'actualité. De nouveaux articles peuvent donc sortir à tout moment. Néanmoins, cela amène parfois à recevoir un segment [RST] qui signifie qu'une rupture anormale de la connexion s'est produite.

5. Tous les navigateurs que j'ai testés utilisent cette suite Cipher.

D. Connexion aux autres services

Le serveur se connecte aux autres services de la RTBF en utilisant des segments [SYN, ECN, CWR]. ECN (Explicit Congestion Notification) est une extension de protocole définie dans la RFC3168. Elle permet de signaler la congestion du réseau avant qu'il n'y ait perte de paquets. CWR (Congestion Window Reduced) sert à informer le récepteur que la fenêtre de congestion a été réduite [13]. Le serveur de la RTBF suppose donc par défaut que le réseau est saturé à l'envoi de son premier segment de connexion.

VI. Conclusion

En conclusion, à l'aide de outils simples, nous pouvons obtenir beaucoup d'informations sur la manière dont un site a été construit et optimisé. J'ai réalisé de nombreuses découvertes notamment sur l'hébergement mutualisé ou sur la manière dont la RTBF optimise le nombre de requêtes en fonction de la taille du navigateur.

Références

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee. 'Hypertext Transfer Protocol – HTTP/1.1'. Disponible : https://tools.ietf.org/html/rfc2616#section-10.3.2. [Consulté : 27 nov. 2018].

[2] B. Marshall and C. Stephanie. 'How Domain Name Servers Work'. Disponible : https://computer.howstuffworks.com/dns.htm. [Consulté : 27 nov. 2018].

[3] J. McIlwain. 'How Content Delivery Networks Work'. Disponible : https://www.cdnetworks.com/en/news/how-content-delivery-networks-work/4258. [Consulté : 28 nov. 2018].

[4] Fastly. 'Shielding'. Disponible : https://docs.fastly.com/guides/performance-tuning/shielding. [Consulté : 10 dec. 2018].

[5] Fastly. 'Understanding the X-Timer header'. Disponible : https://docs.fastly.com/guides/performance-tuning/understanding-the-xtimer-header. [Consulté : 10 dec. 2018].

[6] Fastly. 'Understanding cache HIT and MISS headers with shielded services'. Disponible : https://docs.fastly.com/guides/performance-tuning/understanding-cache-hit-and-miss-headers-with-shielded-services. [Consulté : 10 dec. 2018].

[7] Sidstamm. 'Security/CSP/Spec'. Disponible : https://wiki.mozilla.org/index.php?title=Security/CSP/Spec&oldid=133465. [Consulté : 14 dec. 2018].

[8] Digital Guide. 'QUIC : qu'est-ce qui se cache derrière le protocole expérimental de Google ?'. Disponible : https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/quic/. [Consulté : 15 dec. 2018].

[9] I. Fette, Google Inc., A. Melnikov et Isode Ltd. 'RFC 6455 - The WebSocket Protocol'. Disponible : https://tools.ietf.org/html/rfc6455#section-1.7. [Consulté : 15 dec. 2018].

[10] Pål Hermunn Johansen. 'Varnish Cache 6.0.1'. Disponible : https://varnish-cache.org/releases/rel6.0.1.html. [Consulté : 16 dec. 2018].

[11] Omar Santos. 'BREACH, CRIME and Black Hat'. Disponible : https://blogs.cisco.com/security/breach-crime-and-blackhat. [Consulté : 15 dec. 2018].

[12] Information Sciences Institute. 'INTERNET PROTOCOL SPECIFICATION'. Disponible : https://tools.ietf.org/html/rfc791#page-2. [Consulté : 16 dec. 2018].

[13] K. Ramakrishnan, S. Floyd, ACIRI, D. Black, EMC. 'The Addition of Explicit Congestion Notification (ECN) to IP'. Disponible : https://tools.ietf.org/html/rfc3168. [Consulté : 16 dec. 2018].