RFC 8952 : Captive Portal Architecture

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 décembre 2020

Date de publication du RFC : Novembre 2020

https://www.bortzmeyer.org/8952.html

Les portails captifs sont ces insupportables pages Web vers lesquels certains réseaux d'accès en WiFi vous détournent pour vous faire laisser des données personnelles avant d'avoir accès à l'Internet. Très répandus dans les hôtels et aéroports, ils utilisent pour ce détournement des techniques qui sont souvent celles des attaquants, et ils posent donc de graves problèmes de sécurité, sans parler de l'utilisabilité. Le groupe de travail CAPPORT <https://datatracker.ietf.org/wg/capport/> de l'IETF travaille à définir des solutions techniques qui rendent ces portails captifs un peu moins pénibles. Ce RFC décrit l'architecture générale du système, d'autres RFC décrivant les solutions partielles.

Vu de l'utilisateurice, la connexion à l'Internet sur pas mal de "*hotspots*" se fait ainsi : on accroche le réseau WiFi, on se rend à une page Web quelconque et, au lieu de la page demandée, on est détourné vers une page des gérants du "*hotspot*", qui vous montre de la publicité, exige des données personnelles (et parfois une authentification) et l'acceptation de conditions d'utilisation léonines, avant de continuer. Si la page que vous vouliez visiter était en HTTPS, ce qui est le cas le plus courant aujourd'hui, vous aurez probablement en outre un avertissement de sécurité. Certains systèmes d'exploitation, comme Android, ou certains navigateurs, comme Firefox, détectent automatiquement la captivité, suspendent leurs autres activités, et vous proposent d'aller directement à une page du portail. Mais ça reste pénible, par exemple parce que c'est difficile à automatiser. (Un exemple d'automatisation en shell avec curl que j'ai récupéré en ligne <https://gist.github.com/Lekensteyn/fe10f5df53b37604125e510ee2a423d5> est . Un autre en Go et utilisant Chrome est captive-browser <https://blog.filippo.io/captive-browser/>.)

Techniquement, la mise en œuvre du concept de portail captif se fait en général actuellement en détournant le trafic HTTP (via un "*transparent proxy*") ou avec un résolveur <https://www.bortzmeyer.org/resolveur-dns.html> DNS menteur, qui donne l'adresse IP du portail en réponse à toute requête. L'annexe A du RFC résume les mises en œuvre actuelles de portails captifs, et les méthodes utilisées pour pouvoir se connecter le moins mal possible. La machine qui veut savoir si elle est derrière un portail captif va typiquement faire une requête HTTP vers un URL connu et contrôlé par le fournisseur de l'application ou du système d'explication. Cet URL est appelé le canari. (Au lieu ou en plus de

HTTP, on peut faire un test DNS sur un nom de domaine où on connait le résultat, au cas où le portail captif fonctionne avec un résolveur menteur. Attention, DNS et HTTP peuvent donner des résultats différents, détournement DNS mais pas HTTP, ou bien le contraire.) Notez que toutes les techniques avec un canari peuvent mener à des faux négatifs si le canari est connu et que le gardien derrière le portail captif le laisse passer, via une règle spécifique.

Notre RFC choisit d'être positif et donne une liste des exigences auxquelles essaie de répondre les solutions développées dans le groupe de travail. Ces exigences peuvent aussi être lues comme une liste de tous les inconvénients des portails captifs actuels. Un échantillon de ces exigences :

— Ne pas détourner les protocoles standards de l'Internet comme HTTP, et, d'une manière générale, ne pas se comporter comme un attaquant (alors que les portails actuels violent systématiquement la sécurité en faisant des attaques de l'intermédiaire).
— Ne pas détourner non plus le DNS, puisque la solution doit être compatible avec DNSSEC, outil indispensable à la sécurité du DNS.
— Travailler au niveau IP (couche 3) et ne pas être dépendant d'une technologie d'accès particulière (cela ne doit pas marcher uniquement pour le WiFi).
— Les machines terminales doivent avoir un moyen simple et sûr d'interroger leur réseau d'accès pour savoir si elles sont captives, sans compter sur le détournement de protocoles Internet, sans avoir besoin de « canaris », d'URI spéciaux qu'on tester, comme le `detectportal.firefox.com` de Firefox.
— Les solutions déployées ne doivent pas exiger un navigateur Web, afin d'être utilisable, par exemple, par des machines sans utilisateur humain (les portails actuels plantent toutes les applications non-Web, comme la mise à jour automatique des logiciels). Ceci dit, le RFC se concentre surtout sur le cas où il y a un utilisateur humain.

L'architecture décrite ici repose sur :

— L'annonce aux machines clients d'un URI qui sera l'adresse de l'API à laquelle parler pour en savoir plus sur le portail captif. Cette annonce peut se faire en DHCP (RFC 8910[1]) ou en RA (même RFC). Mais elle pourra aussi se faire avec RADIUS ou par une configuration statique.
— L'information donnée par le portail captif à la machine cliente, lui disant notamment si elle est en captivité ou pas.

Plus concrètement, le RFC liste les composants du problème et donc des solutions. D'abord, la machine qui se connecte ("*user equipment*", dans le RFC). Pour l'instant, la description se limite aux machines qui ont un navigateur Web, même si elles sont loin d'être les seules, ou même les plus nombreuses. Pour que tout marche comme il faut, cette machine doit pouvoir récupérer l'URI de l'API du portail captif (par exemple, elle doit avoir un client DHCP), doit pouvoir gérer le fait que certaines de ses interfaces réseaux sont derrière un portail captif et pas les autres (RFC 7556), doit pouvoir notifier l'utilisateur qu'il y a un tel portail (et attention au hameçonnage !), et ce serait bien qu'il y a ait un moyen de dire aux applications « on est encore en captivité, n'essayez pas de vous connecter tout de suite ». Un exemple de cette dernière demande est fournie par Android où l'utilisation du réseau par défaut ne passe de la 4G au WiFi qu'une fois qu'Android a pu tester que l'accès WiFi n'était pas en captivité. Enfin, si la machine gère l'API des portails captifs, elle doit évidemment suivre toutes les règles de sécurité, notamment la validation du certificat.

Ensuite, deuxième composant, le système de distribution d'informations dans le réseau ("*provisioning service*"), DHCP ou RA. C'est lui qui va devoir envoyer l'URI de l'API à la machine qui se connecte, comme normalisé dans le RFC 8910.

Troisième composant, le serveur derrière l'API. Il permet de se passer des « canaris », des tests du genre « est-ce que j'arrive à me connecter à `https://detectportal.example?` », il y a juste à interroger l'API. Cette API (RFC 8908) doit permettre au minimum de connaitre l'état de la connexion

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8910.txt`

(en captivité ou pas), et d'apprendre l'URI que l'humain devra visiter pour sortir de captivité (« j'ai lu les 200 pages de textes juridiques des conditions d'utilisation, les 100 pages du code de conduite et je les accepte inconditionnellement »). Elle doit utiliser HTTPS.

Enfin, il y a le composant « gardien » ("*enforcement*") qui s'assure qu'une machine en captivité ne puisse effectivement pas sortir ou en tout cas, ne puisse pas aller en dehors des services autorisés (qui doivent évidemment inclure le portail où on accepte les conditions). Il est typiquement placé dans le premier routeur. L'architecture présentée dans ce RFC ne change pas ce composant (contrairement aux trois premiers, qui devront être créés ou modifiés).

Parmi ces différents composants qui interagissent, l'un d'eux mérite une section entière, la section 3, dédiée à la machine qui se connecte ("*user equipment*") et notamment à la question de son identité. Car il faut pouvoir identifier cette machine, et que les autres composants soient d'accord sur cette identification. Par exemple, une fois que le portail captif aura accepté la connexion, il faudra que le système « gardien » laisse désormais passer les paquets. Pour cette identification, plusieurs identificateurs sont possibles, soit présents explicitement dans les paquets, soit déduits d'autres informations. On peut imaginer, par exemple, utiliser l'adresse MAC ou l'adresse IP. Ou bien, pour les identificateurs qui ne sont pas dans le paquet, l'interface physique de connexion. Le RFC liste leurs propriétés souhaitables :

- Unicité,
- Difficulté à être usurpé par un attaquant (sur ce point, les adresses MAC ou IP ne sont pas satisfaisantes, alors que l'interface physique est parfaite ; diverses techniques existent pour limiter la triche sur l'adresse IP comme le test de réversibilité <https://www.bortzmeyer.org/returnability.html>),
- Visibilité pour le serveur derrière l'API (selon la technique de développement utilisée, l'adresse MAC peut être difficile ou impossible à récupérer par ce serveur ; idem pour l'interface physique),
- Visibilité pour le gardien.

Notons que l'adresse IP a l'avantage d'être un identificateur qui n'est pas local au premier segment de réseau, ce qui peut permettre, par exemple, d'avoir un gardien qui soit situé un peu plus loin.

Armé de tous ces éléments, il est possible de présenter le traitement complet (section 4). Donc, dans l'ordre des évènements, dans le monde futur où ces différents RFC auront été déployés :

- La machine de l'utilisateur se connecte au réseau, et obtient des informations via DHCP ou RA,
- Parmi ces informations, l'URI de l'API du portail captif (options du RFC 8910),
- La machine parle à cette API en suivant le RFC 8908, ce qui lui permet d'apprendre l'URI de la page du portail captif prévue pour les humains,
- L'utilisateur humain lit les CGU (ah, ah) et les accepte,
- Le gardien est prévenu de cette acceptation et laisse désormais passer les paquets,
- L'utilisateur peut enfin regarder Joséphine, ange gardien sur Salto.

Si tout le monde joue le jeu, et que les techniques conformes à l'architecture décrite dans ce RFC sont déployées, les portails captifs, quoique toujours pénibles, devraient devenir plus gérables. Hélas, il est probable que beaucoup de déploiements ne soient jamais changé, notamment dans les environnements où le client est. . .captif (hôtels, aéroports) et où il n'y a donc aucune motivation commerciale pour améliorer les choses. Les logiciels vont donc devoir continuer à utiliser des heuristiques de type canari pendant une très longue période, car ils ne pourront pas compter sur des portails captifs propres.

Tout cela laisse ouvert quelques problèmes de sécurité, que la section 6 étudie. D'abord, fondamentalement, il faut faire confiance au réseau d'accès. Via DHCP et RA, il peut vous conduire n'importe où. (Le RFC 7556 discute de cette difficulté à authentifier le réseau d'accès.) Des protections comme l'authentification du serveur en TLS limitent un peu les dégâts mais ne résolvent pas tout. Ceci dit, c'est pour cela que notre RFC impose l'utilisation de HTTPS pour accéder à l'API. Mais, d'une manière générale, le réseau d'accès, et le portail captif qu'il utilise, a tous les pouvoirs. Il peut par exemple vous bloquer l'accès à tout ou partie de l'Internet, même après que vous ayez accepté les conditions d'utilisation.

Il y a aussi des questions de vie privée. L'identificateur de la machine qui se connecte peut être détourné de son usage pour servir à surveiller un utilisateur, par exemple. Les bases de données utilisées par les différents composants de cette architecture de portail captif doivent donc être considérées comme contenant des données personnelles. L'utilisateur peut changer ces identificateurs (pour l'adresse MAC, avec un logiciel comme macchanger <https://github.com/alobbs/macchanger>) mais cela peut casser son accès Internet, si le gardien utilise justement cet identificateur.