Un système anti-censure qui évolue en autonomie : Geneva

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 janvier 2020

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

La lutte technique de la liberté d'expression contre la censure sur l'Internet n'est pas près de s'arrêter. Chaque fois que les censeurs conçoivent de nouvelles techniques, les défenseurs de la liberté mettent au point de meilleurs méthodes pour leur échapper, les censeurs améliorent alors leurs techniques, et ainsi de suite. La partie est donc difficile pour ceux et celles qui réalisent des dispositifs anti-censure, car il faut en permanence suivre et s'adapter. D'où l'idée d'un système qui évolue « tout seul ». Le génial Geneva <https://geneva.cs.umd.edu/> utilise le concept des algorithmes génétiques, pour s'adapter à la censure et évoluer automatiquement en même temps qu'elle.

Geneva <https://geneva.cs.umd.edu/> traite le cas des systèmes de censure Internet de type Homme du Côté. Dans ces systèmes (il en existe beaucoup d'autres ; les censeurs ont de l'imagination, et beaucoup plus d'argent et d'hommes que les défenseurs de la liberté), le censeur peut observer les paquets IP qui passent, et générer ses propres paquets, mais il ne peut pas modifier les paquets envoyés. (Dans les systèmes d'Homme du Milieu, par exemple le pare-feu d'une entreprise, l'attaquant peut modifier ou jeter les paquets, ces systèmes sont donc plus efficaces mais aussi plus coûteux, et parfois plus difficiles à intégrer dans l'infrastructure. Ils sont donc moins employés à l'échelle d'un pays, où le trafic est important. Mais rappelez-vous toujours que les censeurs utilisent plusieurs techniques, et souvent une combinaison de techniques, ajoutant par exemple les résolveurs DNS menteurs <https://www.bortzmeyer.org/dns-menteur.html>.) Un exemple d'attaque de l'Homme du Côté est un système qui observe le SNI dans les requêtes TLS (par exemple HTTPS) et, s'il voit un nom de domaine interdit (ou même simplement un mot censuré, par exemple Falun Gong en Chine), il génère un paquet TCP RST ("*ReSeT*", RFC 793 [1], section 3.1) vers l'émetteur ou le destinataire, coupant ainsi la connexion. TLS ne protège pas contre ces attaques, puisque, contrairement à QUIC <https://www.bortzmeyer.org/quic.html>, TLS ne chiffre pas la couche Transport. (Rappelez-vous également que l'Homme du Côté voit tous les paquets, il a donc la partie bien plus facile que l'attaquant aveugle des RFC 5927 ou RFC 5961.)

Contre de telles techniques de censure, la solution habituelle (et de nombreuses ont été développées) est de semer la confusion chez l'Homme du Côté, en envoyant des paquets qui seront interprétés

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc793.txt

différemment par le censeur et par le vrai destinataire. Par exemple, on envoie un paquet TCP RST avec un TTL qui lui permet d'atteindre la machine du censeur mais pas celle du destinataire. Le censeur croira alors que la connexion est finie, alors qu'en fait elle continuera. (Le censeur, dans le cas d'un État, a des moyens importants mais pas illimités. Il a donc intérêt à libérer de la mémoire en « oubliant » les connexions terminées, ce qui fait que le reste de la communication est ignoré.) L'inconvénient de ces stratégies est qu'il faut recommencer souvent, puisque les techniques de censure évoluent. Chaque fois que la censure se perfectionne, des contournements existants deviennent inefficaces et on est à nouveau censuré jusqu'au développement (lent et laborieux) d'un nouveau contournement. La lutte contre la censure est donc un combat éternel entre l'épée et la cuirasse.

Que peuvent apporter les algorithmes génétiques ici ? Un algorithme génétique est un algorithme qui reprend les grands concepts de la sélection naturelle. On choisit un ensemble de fonctions, on leur donne des règles de mutation, une fonction d'évaluation du résultat (qui joue le rôle de la survie du plus apte) et on laisse ensuite le système évoluer. On peut ainsi explorer un espace de solutions très large en bien moins de temps.

Portrait de Charles Darwin par George Richmond. (Source : Wikimedia Commons <https://commons.wikimedia.org/wiki/File:Charles_Darwin_by_G._Richmond.jpg>.)

Tout l'art du concepteur d'algorithmes génétiques est dans la liberté plus ou moins grande qu'il ou elle laisse aux mutations. Si on laisse les mutations se faire au hasard, l'écrasante majorité des stratégies produites seront non viables, et l'exploration de l'espace des solutions prendra des siècles. Mais si on contraint fortement les mutations possibles, l'algorithme n'ira jamais bien loin, il restera sur les sentiers battus et ne fera pas de découverte disruptive.

Geneva <https://geneva.cs.umd.edu/> tourne sur la machine du client, celui ou celle qui veut accéder à des contenus censurés. Geneva définit les solutions contre les attaques de l'Homme du Côté en trois parties : un déclencheur ("*trigger*") qui dit quand lancer la solution, des actions qui disent ce qu'on va faire (par exemple créer un nouveau paquet) et des arbres ("*action trees*") qui coordonnent la succession des actions. Par exemple, la stratégie mentionnée plus haut (un paquet TCP RST avec un TTL conçu pour atteindre le censeur mais pas le destinataire) s'exprimerait, dans le DSL de Geneva :

```
[TCP:flags:A]-

duplicate(send,
          tamper(TCP:flags:R)(
              tamper(IP:TTL:replace:10)
                  (send)))

-| \/
```

Le terme `[TCP:flags:A]` est le déclencheur : quand on envoie le TCP ACK (accusé de réception), on duplique ce paquet et on envoie à la fois le vrai et une copie ayant le bit R (RST) et un TTL diminué, pour n'atteindre que le censeur et lui faire croire que la connexion est finie. Un autre exemple corrompt délibérément la somme de contrôle TCP : certains censeurs ignorent cette somme de contrôle, ce qui permet d'envoyer un paquet (par exemple RST) qui ne sera lu que par l'Homme du Côté.

Geneva commence avec un ensemble de solutions faites à la main, les individus sur lesquels va s'exercer l'évolution. Ensuite, les mutations se produisent, portant sur les déclencheurs, sur les actions, et sur les arbres (la succession des actions). Geneva teste ensuite les résultats des mutations en cherchant à se connecter à un site Web censuré. Le critère de sélection est donc la réussite dans le contournement de

la censure. (Les règles exactes sont plus complexes, car il faut éviter de se laisser piéger dans des minima locaux ; ce n'est pas parce qu'une solution marche qu'elle est la meilleure. Et puis certains systèmes de censure ne sont pas déterministes ; soit suite à une bogue, soit délibérement pour compliquer l'analyse, ils ne bloquent pas toujours.)

Geneva a été mis en œuvre en Python, utilisant évidemment Scapy, et le NFqueue de Netfilter sur Linux pour accéder au réseau. Notons que certaines opérations nécessitent d'être root (modifier les en-têtes IP) mais pas toutes (changer le découpage des données en segments TCP), ce qui permet dans certains cas de faire tourner Geneva, ou du moins le script (la stratégie) sélectionné, sans être root. Le code pour exécuter les scripts (mais pas encore celui avec l'algorithme génétique) est disponible en ligne <https://github.com/Kkevsterrr/geneva>.

Quels sont les résultats ? Eh bien, Geneva fonctionne, et même très bien d'après ses auteurs. En seulement quelques heures, il trouve une stratégie de contournement de la censure. Geneva a été testé dans le laboratoire, face à des solutions de censure connues, mais aussi en vrai sur le terrain, en Chine face au GFW, en Inde et au Kazakhstan. Disons-le tout de suite, en matière de censure, le GFW est le seul système intéressant, les autres sont vraiment primitifs. Geneva n'a eu aucun problème à trouver une stratégie pour échapper aux censures indienne et kazakhstanaise (simplement mettre le SNI dans deux segments TCP différents suffit, au Kazakhstan) mais le GFW lui a donné plus de fil à retordre. Le script :

```
[TCP:flags:PA]-fragment{tcp:8:True}(send,send)-|
[TCP:flags:A]-tamper{TCP:seq:corrupt}(send)-| \/
```

a réussi à passer. J'ai également bien aimé la stratégie FRAPUN (ainsi nommée en raison des bits de l'en-tête TCP qu'elle met à 1.)

Un des avantages des algorithmes génétiques est qu'ils peuvent trouver des solutions auxquelles leurs auteurs n'ont pas pensé (mais qu'ils peuvent expliquer a posteriori). Plus drôle, Geneva a aussi trouvé une solution que les auteurs du programme n'ont pas pu expliquer, mais qui marche face au GFW :

```
[TCP:flags:PA]-
fragment{tcp:8:True}(send,
fragment{tcp:4:True}(send, send))-| \/
```

Ce très bon article se termine par une discussion sur l'avenir de la lutte anti-censure. Geneva ne va évidemment pas mettre un point final à la question, il est certain que les censeurs s'y adapteront (c'est un problème classique en sécurité informatique : les logiciels distribués publiquement peuvent être utilisés par les bons comme par les méchants.)

J'ai aussi apprécié que l'article n'utilise pas de "*buzzwords*" comme IA ou "*machine learning*". Aujourd'hui, où les chercheurs doivent passer davantage de temps à chercher des subventions et de l'impact médiatique (l'un n'allant pas sans l'autre) plutôt qu'à chercher, c'est appréciable. Mais, hélas, le dépôt git du projet <https://github.com/Kkevsterrr/geneva>, lui, le fait.

Notez que l'article mentionne des considérations éthiques. Lancer Geneva sur une machine en Chine peut vous attirer des ennuis. (L'Inde est moins dangereuse, de ce point de vue.) Certes, Geneva n'est pas trop bavard en octets mais son comportement si particulier pourrait être remarqué par les systèmes de surveillance que déploient la plupart des pays. C'est un problème fréquent quand on étudie la censure <https://labs.ripe.net/Members/kistel/ethics-of-ripe-atlas-measurements>. Il faut donc bien prévenir les utilisateurs (ce que rappelle bien la section "*Disclaimer*" dans la documentation d'installation du logiciel.)