RFC 7194 : Default Port for IRC via TLS/SSL

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 août 2014

Date de publication du RFC : Août 2014

https://www.bortzmeyer.org/7194.html

Le protocole de messagerie instantanée IRC est toujours très utilisé, malgré la concurrence de XMPP. Il n'existait pas de port « officiel » pour les connexions IRC sécurisées par TLS, ce qui est fait avec ce RFC, qui décrit l'usage, déjà très répandu, du port 6697.

IRC est officiellement standardisé dans les RFC 1459[1], RFC 2810, RFC 2811, RFC 2812 et RFC 2813 mais il ne faut pas s'y fier : la pratique réelle est loin de ces RFC, qu'il faut plutôt utiliser comme point de départ d'une description du protocole que comme une « Bible » définitive. IRC avait depuis longtemps deux ports standards <https://www.iana.org/assignments/port-numbers>, 194 pour le trafic en clair et 994 pour le trafic chiffré avec TLS. Ces deux ports sont inférieurs à 1 024 et nécessitent donc que le serveur soit lancé par root sur une machine Unix. Comme ce n'est pas toujours possible, ou souhaitable, en pratique, les serveurs IRC écoutent en général sur 6667 en clair et 6697 en chiffré. Le port 6667 a été documenté pour cet usage mais ce n'était pas encore le cas du 6697. Les clients IRC n'avaient donc pas de moyen standard fiable de découvrir si un serveur gérait les connexions IRC sur TLS.

La section 2 du RFC décrit le fonctionnement d'IRC sur TLS : le client se connecte à un serveur IRC. Une fois la connexion établie, la négociation TLS a lieu. Le serveur est censé présenter un certificat d'une autorité reconnue, dont le nom est le nom du serveur. Le client peut s'authentifier, aussi comme décrit dans la documentation d'OFTC <http://www.oftc.net/oftc/NickServ/CertFP> . Une fois la négociation terminée avec succès, les identités vérifiées et tout, le trafic passe alors par cette connexion protégée.

Le nouveau numéro est désormais dans le registre IANA <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml> :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1459.txt

```
ircs-u            6697       tcp    Internet Relay Chat via    TLS/SSL              2014-02-11
```

À noter que ce mécanisme ne protège que la liaison client¡-¿serveur. IRC a aussi des communications serveur¡-¿serveur (coordination non standardisée entre les différentes machines qui mettent en œuvre un service IRC) qui ne sont pas traitées ici, même si le RFC recommande aux opérateurs IRC de systématiquement chiffrer le trafic entre machines d'un même service. À noter aussi qu'IRC ne fait pas de liaison directe en pair à pair. Quand Alice et Bob se parlent en IRC, cela passe toujours par un serveur et, même si TLS sécurise la connexion avec le serveur, cela ne protège pas Alice et Bob d'un espionnage par le serveur (par exemple si celui-ci participe au programme PRISM), comme le rappelle la section 3 du RFC. Le chiffrement qui n'est pas de bout en bout a des pouvoirs de protection limités ! Seul OTR fournirait une protection de bout en bout (mais nécessite un peu plus d'action de la part d'Alice et Bob alors que TLS est largement invisible).

L'annexe A du RFC rappelle qu'en 2010, une étude avait montré que dix des vingt principaux <http://irc.netsplit.de/networks/top100.php> services IRC géraient TLS.

La plupart des grands serveurs IRC aujourd'hui gèrent TLS. Parmi les exceptions, j'ai trouvé `ircnet.nerim.fr` mais aussi le serveur IRC du W3C qui n'est hélas accessible en TLS que par les membres du W3C. Avec un serveur qui gère TLS, et avec Pidgin comme client, il faut cocher la case "*Use SSL [sic]*" mais aussi changer manuellement le port, pour 6697 :

On verra alors des connexions TLS :

```
(09:00:20) gandalf.geeknode.org: (notice) *** You are connected to gandalf.geeknode.org with TLSv1-AES256-SH
```

À noter que Geeknode est signé par CAcert <https://www.bortzmeyer.org/cacert.html> comme beaucoup de serveurs IRC. Freenode, par contre, est signé par Gandi :

```
% openssl s_client -connect chat.freenode.net:6697
CONNECTED(00000003)
depth=1 C = FR, O = GANDI SAS, CN = Gandi Standard SSL CA
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/OU=Domain Control Validated/OU=Gandi Standard Wildcard SSL/CN=*.freenode.net
   i:/C=FR/O=GANDI SAS/CN=Gandi Standard SSL CA
 1 s:/C=FR/O=GANDI SAS/CN=Gandi Standard SSL CA
   i:/C=US/ST=UT/L=Salt Lake City/O=The USERTRUST Network/OU=http://www.usertrust.com/CN=UTN-USERFirst-Hardw
---
...
```