RFC 6872 : The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Information Model

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 février 2013

Date de publication du RFC : Février 2013

https://www.bortzmeyer.org/6872.html

Un logiciel connecté à l'Internet a tout intérêt à enregistrer ses activités dans un « livre de bord » dont la lecture permettra notamment d'analyser ce qui s'est produit. Une des histoires à succès de la journalisation est celle des serveurs Web qui disposent, depuis longtemps, d'un format de journalisation standard, le "*Common Log Format*" (CLF). Ainsi, des logiciels comme Apache et Squid produisent des journaux compatibles, ce qui a permis de développer d'innombrables outils d'analyse et de statistiques pour ce format. Mais le protocole SIP, contrairement à HTTP, n'avait pas un tel format, chaque logiciel SIP journalisait comme cela lui plaisait. Ce fut le travail du groupe SIPCLF <http://tools.ietf.org/wg/sipclf> de l'IETF que de développer 1) un cadre standard de journalisation, dans ce RFC 6872 [1] 2) un format standard utilisant ce cadre, dans le RFC 6873.

Un exemple de journal au format CLF produit par Apache est :

```
192.0.2.219 - - [03/Feb/2013:07:33:07 +0000] "GET /samknows.html HTTP/1.0" 200 3923 "http://www.bortzmeyer.org/"
```

Mais HTTP a un modèle d'interaction très simple, chaque requête (ici, `GET /samknows.html` qui veut dire « passe moi la page `samknows.html` x) étant complète et autonome. SIP est plus complexe et justifie donc une approche plus élaborée (la section 7 détaille cette différence entre les deux protocoles).

La section 3 du RFC synthétise la définition du problème : la variété des formats de journaux utilisés par les logiciels SIP rendent les analyses pénibles (particulièrement lorsqu'on veut analyser un appel

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6872.txt

qui est passé par plusieurs logiciels). Il faut donc un format commun. Il doit être utilisable par des outils simples en ligne de commande (un format fondé sur XML ne conviendrait donc pas). Il doit être cherchable de manière efficace (un journal SIP peut grossir très vite, c'est un protocole bavard), ce qui n'est pas le cas du CLF des serveurs HTTP, où il faut faire une analyse syntaxique de chaque ligne. Et, enfin, il doit pouvoir être généré facilement par tous les logiciels, sans dépendre de bibliothèques qui ne sont pas forcément disponibles partout (rien que pour cela, la famille ASN.1 ne convient donc pas).

La section 4, elle, est l'anti-cahier des charges, elle expose tout ce que le format commun de journaux SIP n'est **pas** : ce n'est pas un mécanisme de facturation (il se focalise sur la signalisation et ne voit donc pas l'échange de contenu), ce n'est pas un outil de mesure de la qualité de service, et ce n'est pas un outil d'espionnage, il n'est pas prévu pour aider Big Brother.

Le groupe de travail `<http://tools.ietf.org/wg/sipclf>` avait envisagé quatre possibilités, toutes abandonnées, avant de s'attaquer à la définition d'un format commun (section 5) :

— Utiliser les "*Call Detail Records*" traditionnels de la téléphonie. Plutôt conçus pour la facturation, ces CDR ne contiennent pas toute l'information dont on a besoin pour déboguer des problèmes SIP. Par exemple, le CDR contiendra plutôt les coordonnées de l'abonné et pas les champs `From:` et `To:` de SIP (qui peuvent être différents de l'abonné). En outre, les CDR sont typiquement générés par des serveurs intermédiaires, alors que le format commun de SIP pourra être utilisé partout (par exemple par les "*softphones*"). Enfin, le monde SIP est bien plus large que celui des telcos : des universités, par exemple, utilisent SIP, sans pour autant générer de CDR.
— Utiliser de la capture de paquets, genre pcap. Ces outils comme Wireshark savent parfaitement décoder le SIP. Mais ce mécanisme ne répond pas à l'exigence « pouvoir utiliser grep » du cahier des charges. Il ne correspond pas forcément à ce que le logiciel SIP peut faire (sur un téléphone Android, le client SIP n'a pas le droit de faire une capture de paquets directement sur l'interface, alors qu'il a analysé tous les champs du paquet et peut donc facilement les journaliser). Et enfin, si on utilise TLS, cette approche peut être impossible, les paquets étant chiffrés.
— Et syslog (RFC 5424) ? Sa principale limite est qu'il ne spécifie pas le format du contenu des messages. Et toute entité SIP n'a pas forcément un serveur syslog facilement accessible.
— Il reste IPFIX (RFC 7011). Mais ce n'est pas un outil de journalisation, il ne produit pas ce que l'on cherche ici.

La section 6 revient sur les motivations de ce travail, en faisant une liste complète des scénarios d'usage du futur format commun :

— Une référence commune entre différents outils SIP. Utilisant le même modèle de données, leurs journaux seront comparables.
— La possibilité d'écrire des logiciels d'analyse génériques. Pensez à la variété et à la qualité des outils d'analyse de journaux HTTP (comme analog), rendues possibles par le format commun.
— Corrélation entre événements se situant sur des serveurs différents. Un appel SIP peut passer par pas mal de machines qui, dans le plus compliqué des cas, appartiennent à des domaines administratifs différents.
— Analyse de tendances, comme la détermination des heures de pointe selon le domaine,
— Entraîner des IDS comme dans l'article « "*A Self-learning System for Detection of Anomalous SIP Messages*" `<http://eprints.pascal-network.org/archive/00004172/01/2008-iptcomm.pdf>` »,
— Test de matériels et de logiciels SIP, en utilisant des appels réels, journalisés, comme scénarios à rejouer,
— Analyse d'incident (« l'appel a été coupé au bout de trois minutes, que s'est-il passé exactement ? »),
— Et bien d'autres encore.

Armé de ces considérations, le RFC définit ensuite (section 8) le **modèle informationnel** du journal SIP, c'est-à-dire la liste de ce qu'il faut journaliser (rappelez-vous que la syntaxe du journal n'apparaîtra que dans le RFC 6873). Ce modèle comprend une liste de champs obligatoires et la possibilité de journaliser des champs facultatifs.

D'abord, les champs obligatoires (je donne une liste non exhaustive, voir la section 8.1 du RFC pour la liste complète) :

— Estampille temporelle,
— Adresses IP source et destination,
— Ports source et destination,
— Expéditeur et destinataire (dans SIP, ce sont les champs `From:` et `To:`, qui contiennent des URI),
— Résultat ("*status code*"),
— Identificateur de la transaction, pour le client et pour le serveur,
—

Commë leur nom l'indique, les champs obligatoires doivent être présents mais, dans certains cas, un champ donné n'a pas de sens pour cet évenement et le format doit alors spécifier un moyen de représenter cette absence.

La section 9 fournit quelques exemples (mais attention, ce sont des exemples pour illustrer le modèle informationnel, pas pour illustrer la syntaxe, qui n'est pas définie par ce RFC, mais dans le RFC 6873). Dans tous les exemples, Alice, en `198.51.100.1:5060`, essaie d'appeler Bob, qui a deux téléphones actifs, un en `203.0.113.1:5060` et un en `[2001:db8::9]:5060`. Alice et Bob ont chacun un fournisseur, respectivement P1 (`198.51.100.10:5060`) et P2 (`203.0.113.200:5060`).

Ainsi, lorsque le "*softphone*" d'Alice s'enregistre auprès de son fournisseur SIP, le message SIP sera journalisé :

```
Timestamp: 1275930743.699
Message Type: R
Directionality: s
Transport: udp
CSeq-Number: 1
CSeq-Method: REGISTER
R-URI: sip:example.com
Destination-address: 198.51.100.10
Destination-port: 5060
Source-address: 198.51.100.1
Source-port: 5060
To: sip:example.com
To-tag: -
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f81-d4-f6@example.com
Status: -
Server-Txn: -
Client-Txn: c-tr-1
```

On note les champs sans valeur comme `Status`, puisqu'il s'agit d'une requête, il n'y a pas encore de réponse. Lorsqu'elle arrive, le journal va enregistrer :

```
Timestamp: 1275930744.100
Message Type: r
Directionality: r
Transport: udp
CSeq-Number: 1
CSeq-Method: REGISTER
R-URI: -
Destination-address: 198.51.100.1
Destination-port: 5060
Source-address: 198.51.100.10
Source-port: 5060
To: sip:example.com
To-tag: reg-1-xtr
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f81-d4-f6@example.com
Status: 100
Server-Txn: -
Client-Txn: c-tr-1
```

SIP peut aussi fonctionner en pair-à-pair, sans fournisseur. Ici, Alice appelle Bob :

```
Timestamp: 1275930743.699
Message Type: R
Directionality: s
Transport: udp
CSeq-Number: 32
CSeq-Method: INVITE
R-URI: sip:bob@bob1.example.net
Destination-address: 203.0.113.1
Destination-port: 5060
Source-address: 198.51.100.1
Source-port: 5060
To: sip:bob@bob1.example.net
To-tag: -
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f82-d4-f7@example.com
Status: -
Server-Txn: -
Client-Txn: c-1-xt6
```

et Bob répond positivement (résultat 200) :

```
Timestamp: 1275930746.100
Message Type: r
Directionality: r
Transport: udp
CSeq-Number: 32
CSeq-Method: INVITE
R-URI: -
Destination-address: 198.51.100.1
Destination-port: 5060
Source-address: 203.0.113.1
Source-port: 5060
To: sip:bob@example.net
To-tag: b-in6-iu
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f82-d4-f7@example.com
Status: 200
Server-Txn: -
Client-Txn: c-1-xt6
```

La traditionnelle section sur la sécurité (section 10) appelle à quelques règles de prudence : les journaux contiennent des données personnelles et leur gestion doit se faire en prenant en compte les questions de protection de la vie privée. Pas question donc de mettre des journaux en accès libre. Le modèle d'information décrit dans ce RFC est suffisamment détaillé pour dispenser un attaquant qui aurait accès aux journaux d'écouter le trafic SIP. Le RFC suggère même d'étudier des méthodes de protection plus avancées comme le chiffrement du disque qui contient les journaux et la transmission des journaux uniquement par des canaux sécurisés, par exemple avec TLS.

Autre risque de sécurité, celui d'un dépassement de tampon : les entrées dans un journal SIP peuvent être très longues (surtout avec les champs optionnels) et un analyseur naïf qui utiliserait des structures de données de taille fixe pourrait se faire déborder.

La section 11 contient d'autres conseils utiles, pour les opérateurs cette fois : elle rappelle que les journaux sont facilement de grande taille, qu'un mécanisme de rotation (par exemple avec logrorate <https://fedorahosted.org/logrotate/>) est nécessaire.

Merci à Olivier Festor et Régis Montoya pour leur relecture.