Centralisation sécurisée de syslog avec TLS

Matthieu Herrb

Capitoul - 16 février 2023

https://homepages.laas.fr/matthieu/talks/capitoul-tcp-tls-logs.pdf

Licence

3.3.5 Sécurisation du transfert des journaux

Il est nécessaire de mettre en place des mécanismes de protection garantissant la confidentialité et l'intégrité des flux de transfert des journaux, en particulier lorsque les données transitent sur des réseaux non maîtrisés. Le besoin en confidentialité est aussi fonction de la sensibilité des informations journalisées. L'idéal est de mettre en place un canal de transmission dédié réalisé à l'aide de mécanismes cryptographiques robustes [12]. Idéalement, ce canal doit être établi après authentification mutuelle de la machine émettrice et du serveur de collecte en utilisant des certificats issus d'une autorité de certification de confiance. Il existe pour certains protocoles de transfert de journaux une version sécurisée utilisant TLS [10] qui permet de répondre à ces exigences. Si le protocole de transfert de journaux ne dispose pas nativement de mécanismes de chiffrement ou de signature, il est possible de s'appuyer sur des protocoles tels que SSH [4] ou IPsec [7].

Recommandations de sécurité pour l'architecture d'un système de journalisation, ANSSI, Janvier 2022, page 18.

Syslog historique

Pour le transfert de messages de journal (événements) il faut que la source (émetteur) et la destination (collecteur) soient d'accord sur le format.

Historiquement 2 formats :

- Format BSD RFC 3164
- Format IETF RFC 5424

Transport : UDP port 514, un message par datagramme → RFC 5426

Simple, efficace, mais pas sécurisé...

Sécurisation

RFC 5425 : transport sur TCP & TLS, port 6514.

Assure:

- l'identification de la source (optionnel) et du collecteur
- la confidentialité des messages pendant le transport
- l'intégrité des messages transportés

Mais :

- Plusieurs implémentations... $\rightarrow$ soucis d'interopérabilité.
- L'envoi d'un message devient plus coûteux (pbs de performances et de disponibilité)
- Déploiement plus complexe (certificats & autorités de certfication)

La longueur, c'est important (*TCP_framing*)

Avec TCP+TLS, plusieurs messages dans une connexion :
Comment séparer les messages ?
Le RFC 5425 définit deux options :

- caractère séparateur (en général *line-feed*)
- champ longueur en tête de message (aka TCP_framing)

En pratique **privilégier la 2e**, plus largement supportée.
(mais pas la valeur par défaut dans rsyslog)

Certificats et identités

Rappel de la recommandation ANSSI :
« authentification mutuelle de la machine émettrice et du serveur de collecte »

Donc :

- un certificat sur le serveur de collecte
- un certificat sur **chaque** émetteur de journal
- les certficats de l'AC pour vérifier ces certicats partout
- importance du format RFC 5424: avoir dans le message de journal le nom de l'émetteur cohérent avec son certificat

Il est possible de n'authentifier que le serveur pour se simplifier la vie si on accepte de perdre l'authentification forte des clients.

Autorité de certification (AC) dédiée ?

Si on veut déployer des certificats clients sur toutes les sources de traces : multiplication des certificats... Quasi impossible avec Terena !

Pour une AC dédiée :

- permet de générer des cerficats pour des machines pas connectées à l'internet
- permet de générer des certificats avec duree de vie $>$ un an
- coût faible
- simplifie la validation des sources (clients)

Contre :

- un service de plus à gérer
- sécurité bornée par la sécurité de l'AC

Collecteur syslog-ng

Sans auth clients / RFC5424 + TCP_Framing actifs par défaut.

```
source s_network {
    syslog (
        ip-protocol(6) # TCP
        port(6514)
        transport("tls")
        tls (
            cert-file("/etc/ssl/certs/server.crt") # avec la chaîne de l'AC
            key-file("/etc/ssl/private/server.key")
            peer-verify(optional-untrusted)
        )
    );
};
```

documentation

Collecteur rsyslog

Sans auth clients / RFC5424 + TCP_Framing actif par défaut. **/etc/rsyslog.conf**

```
# provides TLS syslog reception
module(load="imtcp"
    StreamDriver.Name="gtls"
    StreamDriver.Mode="1"
    StreamDriver.Authmode="anon"
)
$DefaultNetstreamDriver gtls
$InputTCPServerRun 6514

$DefaultNetstreamDriverCAFile /etc/ssl/certs/ca-certificates.crt
$DefaultNetstreamDriverCertFile /etc/ssl/certs/server.crt
$DefaultNetstreamDriverKeyFile /etc/ssl/private/server.key
```

Attention **/etc/ssl/private/server.key** lisible par groupe *syslog*

Emetteur syslog-ng

```
destination d_syslog_tls {
    syslog("10.100.20.40"
        transport("tls")
        port(6514)
        tls(peer-verify(required-trusted)
            ca-dir('/etc/ssl/ca') # AC du serveur
        )
    );
};
```

Emetteur rsyslog

Options explicites pour RFC 5424 et TCP_framing

```
global(
    DefaultNetstreamDriverCAFile="/etc/ssl/certs/ca-certificates.crt"
    DefaultNetstreamDriver="gtls"
)

*.* action(type="omfwd" protocol="tcp" port="6514" target="logs.laas.fr"
       StreamDriver="gtls"
       StreamDriverMode="1" # TLS
       StreamDriverAuthMode="x509/certvalid" # vérifie le cert serveur
       TCP_Framing="octet-counted"
)
```

OpenBSD

Collecteur:

```
syslogd_flags = "-S logs.laas.fr:6514"
```

Émetteur:

```
*.* @tls://logs.laas.fr:6514
```

- mode RFC5424 + TCP_framing par défaut
- vérifie le certificat du serveur avec le magasin système
- syslogd(8) syslog.conf(5)

systemd-journald

- Pas compatible syslog
- Paquet : `systemd-journal-remote`
- Source : `systemd-journal-upload.service`
- Collecteur : `systemd-journal-remote.service`
- Certificats clients nécessaires jusqu'à version récente de systemd
- Pas d'outils utiles pour exploiter les journaux centralisés

→ Pas très utilisable...

Conclusion

- Centraliser les logs de manière sécurisée :
- Avec uniquement authentification du serveur
- Multi-plateformes avec RFC 5424 et longueur des messages TCP

Questions ?