Title: GDPR and personal data in web server logs Description: Log anonymizer, logs and data fields masking needed for GDPR data protection regulation with Logagent, light-weight log shipper, filebeat, fluentd or rsyslog alternative with out of the box and extensible log parsing, on-disk buffering, secure transport, bulk indexing to Elasticsearch and Sematext logs management platform

Masking data in logs got really important due to meet the requirements of GDPR a European data protection regulation. In the GDPR role "data controller" for your logs, you should minimize the risk of exposing sensitive data to 3rd parties. In some cases, even IP-Addresses are considered as personal data, but your logs might more sensitive data like username, phone numbers etc.

The GDPR suggests to anonymize, mask or remove personal data before you hand over data to any 3rd party.

There are several methods to anonymize or mask data fields:

1. **Truncate field values in logs.** In case of IP addresses, it could be sufficient to replace the last digits of an IP address. This breaks the potential correlation to a specific user, so it protects the personal data. A part of the network information is still available for analytics (e.g. Country of a user based on the partial IP address).
2. **Hash field values in logs.** Strong hash algorithms don't allow to calculate the original value out of the hash code. You can use the resulting hash code for analytics, e.g. count unique entries.
3. **Encrypt field values in logs.** Encryption might be even more interesting than hashing because you don't lose the information, while you protect the data. For 3rd parties, the encrypted content is not readable. But you could decrypt the data field in case you really need to access the content. E.g. for forensics after a security breach. Depending on the encryption method you might be able to use the encrypted value for analytics, like hash codes.
4. **Remove fields from logs.** The removal of data fields provides has the lowest risk of data leaking, but you might lose the information for troubleshooting or statistics.

You can combine all mentioned methods depending on nature of data and your needs to analyze or restore the information in specific cases.

So let's see how you can mask data in web server logs by and an example using Sematext Logagent.

Sematext Logagent is an open-source, light-weight log shipper parsing many log formats out of the box. With its rich set of input and output plugins, it becomes a general ETL tool for time series data like logs or IoT sensor data. You can read data from various sources like files, databases, Elasticsearch or IoT devices (via MQTT), process the data and store the data in files, databases Apache Kafka, or Elasticsearch, InfluxDB or Sematext Cloud.

In the following examples, we will tail log files and ship the anonymized logs

to Elasticsearch. We will mask the field client_ip with the methods mentioned above.

Truncate IP address fields in logs

The ip-truncate-fields output filter replaces IP addresses with an anonymized string, replacing the last block of an IP address with "0". Example (client_ip field):

IPv4: 192.168.1.22 results in 192.168.1.0 IPv6: 2001:db8:0:0:0:ff00:42:8329 results in 2001:db8:0:0:0:ff00:42:0

All occurrences of the IP address are replaced in the log "message" fields with the new value. Example (message field): "Client connect 192.168.1.22" would result in "Client connect 192.168.1.0".

Configuration:

```
# tail web server logs
input:
  files:
    - '/var/log/*/access.log'

# log agent parses web server logs out of the box ...
outputFilter:
  iptruncate:
    module: ip-truncate-fields
    # JS regular expression to match log source name
    matchSource: !!js/regexp access.log
    fields:
      - client_ip

output:
  module: elasticsearch
  url: https://logsene-receiver.sematext.com
  index: YOUR_LOG_TOKEN
```

Run Logagent with your config: `logagent --config ip-truncate.yml --yaml -n httpd`

The output shows then the last IP block with the value "0" in the field client_ip.

Hash fields in logs

The hash-fields output filter replaces field values with its hash code. All occurrences of the original field value are replaced in the log "message" field with the hash code.

Configuration:

```
# tail web server logs
input:
  files:
    - '/var/log/*/access.log'

# log agent parses web server logs out of the box ...
# output filter to encrypt client_ip and user field in web server logs
outputFilter:
  hashfields:
    module: hash-fields
    # JS regular expression to match log source name
    matchSource: !!js/regexp access.log
    # algorithms supported by nodejs crypto module, e.g. sha1, sha256, sha512, md5, ...
    algorithm: sha256
    fields:
      - client_ip
      - user
output:
  module: elasticsearch
  url: https://logsene-receiver.sematext.com
  index: YOUR_LOG_TOKEN
```

Run Logagent with your config: `logagent --config ip-truncate.yml --yaml -n httpd`

The output shows has codes in the fields user and client_ip.

Encrypt fields in logs

The aes-encrypt-fields output filter encrypts data fields with AES. The original field value gets replaced with its AES encrypted HEX string. All occurrences of the original field value are replaced in the log "message" field with the encrypted value.

Encrypting the same text with the same password results in the same encrypted text. This could be helpful when you need to search for specific encrypted field value. You could encrypt your search term and search the encrypted term in Elasticsearch to find relevant log entries.

Configuration:

Add the following section 'outputFilter' to the Logagent configuration file. Please note you could use the plugin with multiple configurations for different event sources.

```
# tail webs erver logs
input:
  files:
    - '/var/log/*/access_log'
```

```
# log agent parses web server logs out of the box ...
# output filter to encrypt client_ip and user field in web server logs
outputFilter:
  aes:
    module: aes-encrypt-fields
    # JS regular expression to match log source name
    matchSource: !!js/regexp access_log|nginx|httpd
    fields:
      - client_ip
      - user
    password: "Top secret!"
    # algorithms supported by nodejs crypto module, e.g. aes-128-cbc, aes-128-ecb, aes-192-c
    # aes-192-ecb, aes-256-cbc, aes-256-ecb
    # short names might work as well e.g. "aes256"
    # default value is aes256
    algorithm: aes-256-ecb

output:
  module: elasticsearch
  url: https://logsene-receiver.sematext.com
  index: YOUR_LOG_TOKEN
```

Run Logagent with your config:

```
logagent --config laes-encrypt-config.yml -n httpd --yaml
```

The output shows the encrypted values in the fields client_ip and user.

Remove fields from logs

The remove-fields output filter removes fields before the output happens. All occurrences of the original field value are replaced in the log "message" field with the string "!REMOVED!".

Configuration:

Add the following section 'outputFilter' to the Logagent configuration file. Please note you could use the plugin with multiple configurations for different event sources.

```
# tail web server logs
input:
  files:
    - '/var/log/*/access_log'

# log agent parses web server logs out of the box ...
# output filter to encrypt client_ip and user field in web server logs
```

```
outputFilter:
  ip-truncate-fields:
    module: remove-fields
    # JS regular expression to match log source name
    matchSource: !!js/regexp access_log
    fields:
      - client_ip
      - user
```

Run Logagent with your config:

```
logagent --config logagent-example-config.yml -n httpd --yaml
```

The output does not contain client_ip and user field:

```
logSource:    httpd
_type:        access_common
remote_id:    -
method:       GET
path:         /
http_version: HTTP/1.1
status_code:  304
size:         0
@timestamp:   Thu Apr 26 2018 22:02:26 GMT+0200 (CEST)
```

Summary

The examples above show that Logagent provides all required methods to truncate IP addresses, hash, encrypt or remove sensitive data fields. Don't hesitate to contact us for any further question.