Security Auditing Logging

improving auditing and traceability capabilities in the short term

version 0.6+

David Groep

NIKHEF

Outline

- Rationale
- Message content
  - identification
  - time stamping
  - length and parsing
- What to log
  - server startup and termination
  - connection establishment
  - request logging
  - message linking and interrelation
  - aggregation
  - log completeness and accuracy

Rationale

Primary aims

- facilitating *post-factum* security auditing
- at the single-site level
- integrated in the existing site audit infrastructure
- be parsed automatically by software

Secondary aims

- prepare ground for future distributed aggregation systems
- help in trouble shooting is a nice side-effect

Which M/W should log

The guidelines apply to

- all services
- all components that run
  under dedicated or privileged accounts,
- all components that are started and act
  on behalf of a user with enhanced privileges

Syslog

Security logging is to be integrated in site logging systems, so

- use of the syslog(3) facility MUST be supported
- and MUST be enabled by default

- middleware MAY also log to other destinations
- and MAY use indirect systems such as *log4cpp* &c

(guidelines are written in normative, RFC2119 language)

Syslog semantics reminder

Facility

- classification of events used to discriminate sources
- typically different *facilities* end up if different log files
- a few well known ones (DAEMON, FTP, USER, LOCAL1-7)

Severity ('priority')

- is the 'log level' and is strictly incremental
- from DEBUG, INFO, NOTICE, WARN, … up to EMERG
- is orthogonal to the set of facilities
- highest levels are really reserved for system emergencies

Message content: facilities

- the syslog(3) facility MUST be configurable
- messages that *could* contain re-usable private data
  MUST be logged to AUTHPRIV by default
- all other messages SHOULD be
  logged to DAEMON by default

Re-usable private data can appear in many unlikely places

For example,

it is well known that users tend to type a password when asked for the user name – that's why login(8) and ssh(8) use AUTHPRIV also for logging the username

Message content: identification

Messages from a single request, session MUST be relatable

- A unique identifier MUST be prepended to every message

- the identifier
  - MUST contain the (shortened) process name ‘in.ftpd’ (“*TAG*”)
  - MUST contain the process ID between square brackets ‘[4213]’
  - a single colon plus space (“: ”) character MUST follow process id

- if a single process handles more than one request
  - a unique *session identifier* MUST be generated and appended after the colon-cum-space

Message content: time stamp

- normal time stamping is taken care of by syslog

- *if independent (or sub-second resolution) TS's are required*
  - the timestamp MUST appear as the first item after the *msgid*
  - the timestamp MUST be in the ISO8601 '*combination of date and time of day representation in the extended format*'
    like: `2006-11-15T10:13:02Z`
  - MAY contain fractional second information as per ISO8601
  - MUST include the time zone, and zome either
    - MUST be in UTC, indicated by "Z"
    - or MUST be expressed numerically "+01:00" since abbreviations like "PST" are not unique globally
  - date and time specification MUST NOT contain whitespace
  - timestamp SHOULD be logged in UTC ("Zulu time")

Message content: length, parsing

- Note that length of a syslog message on many systems is
  - 1024 octets max (some implementations can eat only 1024!)
  - including the system-prepended time and host information *(typically up to 64-80 characters)*

- Messages MUST be machine readable
  - so use of white space MUST be consistent within each service

- use of "*name=value*" pairs is RECOMMENDED
  - if used, *name* and *value* MUST be separated by a single equal ("=") sign
  - a *value* containing white space MUST be quoted in double-quotes (“”)
  - the *name* SHOULD consist solely of the characters "a"-"z", "A"-"Z", "0"-"9" and "_"
  - if a *value* contains a double-quote (“”) character, it MUST be escaped with a single backslash ("\")

Message characters

Allowable characters:
MUST be seven-bit ASCII in an eight-bit field. In this code set, the only allowable characters are the
visible characters (%d33-126) and space (SP value %d32).

However, no indication of the code set used within the MSG is required, nor is it expected. Other code sets MAY be used as long as the characters used in the MSG are exclusively visible characters and spaces similar to those described above. The selection of a code set used in the MSG part SHOULD be made with thoughts of the intended receiver. A message containing characters in a code set that cannot be viewed or understood by a recipient will yield no information of value to an operator or administrator looking at it. BUT we want to be understood widely

from RFC3164

Examples

- Examples in the document (non-normative) appendices

```
daemon:notice
   jss-serv[5241]: event=NewConnection
        ts=2006-09-28T10:09:23.021Z
        remoteHost=192.16.199.115:28773
        DN="/DC=org/DC=example/CN=Pietje Puk"
daemon:debug  (can be debug severity since the pid remains the same)
   jss-serv[5241]: event=info
        msg="delegating authorization to mapper in-line"
        pid=5241
daemon:notice
   jss-serv[5241]: event=AuthenticationRequest
   msg="access allowed" DN="/DC=org/DC=example/CN=Pietje Puk"
        uid=43004 uname=dzero004 pgid=2008
        gname=dzero sgids=512,100
...
```

What to log?

Start-up, termination

- Log on
  - start-up
  - termination(!)
  - reconfiguration
- at severity NOTICE
- with information
  - configuration used or (re)loaded – if service can be configured
  - on termination: reason of termination or status (success) code
- if termination is due to a signal (except KILL)
  - log severity MUST be raised to WARNING

Service access

1. On each access the source address and port – or the process id and owner of the process connecting to the socket – MUST be logged at severity **LOG_NOTICE**.

2. If the connection is TLS-authenticated with client credentials, the certificate subject DN of the client MUST be logged at severity **LOG_NOTICE**. If the TLS handshake fails, a message including the reason, and if possible the client subject name, MUST be logged at severity **LOG_WARNING**, unless the handshake failed because no data at all was received from the remote peer, when the severity MAY be set lower but not lower than **LOG_NOTICE**. This message MUST re-iterate the connection source information reported in (1).

3. The result of the (subsequent) authorization decision that results in granting access (i.e. the allow decision) MUST be logged at severity **LOG_NOTICE**. If a local identity is assigned to this session or request, the local mapping, including the numeric *uid*, the numeric *gid*, and the list of numeric supplementary *gid*s, MUST be logged at severity **LOG_NOTICE**.

4. The result of a denied authorization MUST be logged at severity **LOG_WARNING**. This message MUST re-iterate the connection source already reported in (1)&(2).

5. In either case, the list of attributes (e.g. obtained from VOMS attribute certificates), or the fact recognising the lack of any attributes, MUST be logged at **LOG_INFO**.

6. Any (session) identifiers that are required to link the authorization state to subsequent specific requests MUST be logged at severity **LOG_NOTICE**.

Service access

Other information at this stage MUST NOT be logged with a severity higher than LOG_DEBUG

- thus, if you want to add more diversity for debugging problems, you MUST do this in a debug-verbosity system that's orthogonal to the syslog severities
- if you want to boost performance, suppress calls to syslog for DEBUG messages unless a service specific debugging flag is on

---

At the end of a request

- If the service supports multiple requests with a single access, and if the termination of such an established session can be identified, a message at severity LOG_NOTICE SHOULD be logged at completion of each session
- Such a message MUST contain the unique identifier that ensures this message can be linked to the initial access messages

Request logging

- For external requests that successfully modify persistent state for which the service is responsible, both the action and the operands MUST be logged at severity LOG_INFO.
  If such an external request fails, it SHOULD be logged at any severity level that is considered appropriate, but not at a lower level than LOG_INFO.

- External requests that do not modify the persistent state and complete successfully MUST NOT be logged at a severity higher than LOG_DEBUG.
  External requests that do not modify the persistent state that failed MAY be logged at any severity level up to but not higher than LOG_INFO.

- Other (internal) actions MAY be logged at any severity up to but not exceeding LOG_INFO.

Linking messages

- A ‘linking’ identifier between a service request and any and all access log messages of a single service MUST be provided in all messages (e.g. using a session id).

- If a service is able (within reasonably bounds) to obtain an external request or session identifier, this identifier SHOULD be logged as part of at least one message that also contains the single service request identifier.

- If a service delegates (part of) the processing of a request or session to another service or process or thread, or if it contacts another service to satisfy parts of a request, a unique identifier referencing this delegation process or thread or service MUST be logged by the parent or invoking process as part of at least one message that also contains the single service request identifier.
  The completion of such a delegated request, or the termination of the process or thread, MUST be logged by the parent or invoking process.
  The message severity for all such messages MUST be set to LOG_NOTICE

Aggregation

- Multiple pieces of information MAY be combined into a single log message, as long as all information is to be logged at the same severity level. If the service only ever sends a single message to syslog, all information MAY be combined into a single log message, regardless of the severity level, and the severity level of this single message MUST be set to the highest severity of any of its constituent parts.

- If a service instance only ever serves a single request, it is RECOMMENDED that the service start up message and the connection establishment message are combined into a single message

How and when to log

In case a service contacts other services to fulfil a request, or initiates processes to which processing is delegated, it is

- strongly RECOMMENDED that log messages are generated and sent out even if the delegation or invocation fails, even if such a failed attempt will cause the service itself to terminate.

- Note that this may imply wrapping any such invocations in code that traps and recovers from exceptions to ensure the log message is generated and sent.

Examples

- Examples in the document (non-normative) appendices

daemon:notice
jss-serv[5241]: event=NewConnection
ts=2006-09-28T10:09:23.021Z
remoteHost=192.16.199.115:28773
DN="/DC=org/DC=example/CN=Pietje Puk"

daemon:debug *(can be debug severity since the pid remains the same)*
jss-serv[5241]: event=info
msg="delegating authorization to mapper in-line"
pid=5241

daemon:notice
jss-serv[5241]: event=AuthenticationRequest
msg="access allowed" DN="/DC=org/DC=example/CN=Pietje Puk"
uid=43004 uname=dzero004 pgid=2008
gname=dzero sgids=512,100

...

Guidelines Document

- Discussions and comments to me or to the security audit mailing list

  security-audit-log-discuss@george.lbl.gov

  - *mailing list will also discuss future directions in context of the new SciDAC project on distributed grid auditing*

- The document in the SCG area of the EGEE-II EDMS tree

  https://edms.cern.ch/document/793208

- current version: 0.6, and still incorporating comments