NAME

ausearch-expression − audit search expression format

OVERVIEW

This man page describes the format of "ausearch expressions". Parsing and evaluation of these expressions is provided by libauparse and is common to applications that use this library.

LEXICAL STRUCTURE

White space (ASCII space, tab and new-line characters) between tokens is ignored. The following tokens are recognized:

Punctuation

**( ) **

Logical operators

! && ||

Comparison operators

< <= == > >= !== i= i!= r= r!=

Unquoted strings

Any non-empty sequence of ASCII letters, digits, and the **_** symbol.

Quoted strings

A sequence of characters surrounded by the **"** quotes. The \ character starts an escape sequence. The only defined escape sequences are \\ and \". The semantics of other escape sequences is undefined.

Regexps

A sequence of characters surrounded by the **/** characters. The \ character starts an escape sequence. The only defined escape sequences are \\ and \/. The semantics of other escape sequences is undefined.

Anywhere an unquoted string is valid, a quoted string is valid as well, and vice versa. In particular, field names may be specified using quoted strings, and field values may be specified using unquoted strings.

EXPRESSION SYNTAX

The primary expression has one of the following forms:

field comparison-operator value

\regexp *string-or-regexp*

field is either a string, which specifies the first field with that name within the current audit record, or the \ escape character followed by a string, which specifies a virtual field with the specified name (virtual fields are defined in a later section).

field is a string. *operator* specifies the comparison to perform

r= r!= Get the "raw" string of *field*, and compare it to *value*. For fields in audit records, the "raw" string is the exact string stored in the audit record (with all escaping and unprintable character encoding left alone); applications can read the "raw" string using **auparse_get_field_str**(3). Each virtual field may define a "raw" string. If *field* is not present or does not define a "raw" string, the result of the comparison is **false** (regardless of the operator).

i= i!= Get the "interpreted" string of *field*, and compare it to *value*. For fields in audit records, the "interpreted" string is an "user-readable" interpretation of the field value; applications can read the "interpreted" string using **auparse_interpret_field**(3). Each virtual field may define an "interpreted" string. If *field* is not present or does not define an "interpreted" string, the result of the comparison is **false** (regardless of the operator).

< <= == > >= !==
Evaluate the "value" of *field*, and compare it to *value*. A "value" may be defined for any field or virtual field, but no "value" is currently defined for any audit record field. The rules of parsing *value* for comparing it with the "value" of *field* are specific for each *field*. If *field* is not present, the result of the comparison is **false** (regardless of the operator). If *field* does not define a "value", an error is reported when parsing the expression.

In the special case of **\regexp** *regexp-or-string*, the current audit record is taken as a string (without interpreting field values), and matched against *regexp-or-string*. *regexp-or-string* is an extended regular expression, using a string or regexp token (in other words, delimited by **"** or /).

If *E1* and *E2* are valid expressions, then **!** *E1*, *E1* **&&** *E2*, and *E1* **||** *E2* are valid expressions as well, with the usual C semantics and evaluation priorities. Note that **!** *field op value* is interpreted as **!(***field op value***)**, not as **(!***field***)** *op value*.

VIRTUAL FIELDS

The following virtual fields are defined:

\timestamp
The value is the timestamp of the current event. *value* must have the **ts:***seconds*.*milli* format, where *seconds* and *milli* are decimal numbers specifying the seconds and milliseconds part of the timestamp, respectively.

\record_type
The value is the type of the current record. *value* is either the record type name, or a decimal number specifying the type.

SEMANTICS

The expression as a whole applies to a single record. The expression is **true** for a specified event if it is **true** for any record associated with the event.

EXAMPLES

As a demonstration of the semantics of handling missing fields, the following expression is **true** if *field* is present:

(*field* **r= "") || (***field* **r!= "")**

and the same expression surrounded by **!(** and **)** is **true** if *field* is not present.

FUTURE DIRECTIONS

New escape sequences for quoted strings may be defined.

For currently defined virtual fields that do not define a "raw" or "interpreted" string, the definition may be added. Therefore, don’t rely on the fact that comparing the "raw" or "interpreted" string of the field with any value is **false**.

New formats of value constants for the **\timestamp** virtual field may be added.

AUTHOR

Miloslav Trmac