CybOX Regular Expression Support – Version 1.0

The CybOX Language supports the definition of observable patterns that leverage regular expressions for criteria to be evaluated against instance data. To support interoperability, the CybOX Language defines an intersection of regular expression character classes, operations, expressions, and other lexical tokens defined by common regular expression syntaxes (Boost, Java, JavaScript, .NET, PCRE, Perl 5, and Python) to be used when crafting regular expressions. This set of capabilities was identified through a survey of several regular expression libraries in an effort to increase interoperability between pattern authors and evaluators. Pattern authors are encouraged to leverage the capabilities defined below when creating regular expressions, but are not required to do so. Deviation from these regular expression capabilities is discussed later in this section.

Supported Regular Expression Syntax

Regular expression modifiers, like the “m”, “i”, “s”, and “x” modifiers used by Perl are not supported. Character matching assumes a Unicode character set. Note that no syntax is supplied for specifying code points in hex; actual Unicode characters must be used instead.

The following regular expression capabilities are identified as recommended in the CybOX Language. Regular expression capabilities that are not listed below should be avoided if possible as they are likely to be incompatible or have different meanings between commonly used regular expression libraries.

Metacharacters

```
\   Quote the next metacharacter
^   Match the beginning of the line
.   Match any character (except newline)
$   Match the end of the line (or before newline at the end)
|   Alternation
()  Grouping
[]  Character class
```

Greedy Quantifiers

```
*      Match 0 or more times
+      Match 1 or more times
?      Match 1 or 0 times
{n}    Match exactly n times
{n,}   Match at least n times
{n,m}  Match at least n but not more than m times
```

Reluctant Quantifiers

```
*?     Match 0 or more times
+?     Match 1 or more times
??     Match 0 or 1 time
{n}?   Match exactly n times
{n,}?  Match at least n times
{n,m}? Match at least n but not more than m times
```

Escape Sequences

```
\t     tab                   (HT, TAB)
```

```
\n      newline                 (LF, NL)
\r      return                  (CR)
\f      form feed               (FF)
\033    octal char (think of a PDP-11)
\x1B    hex char
\c[     control char
```

Character Classes

```
\w  Match a "word" character (alphanumeric plus "_")
\W  Match a non-word character
\s  Match a whitespace character
\S  Match a non-whitespace character
\d  Match a digit character
\D  Match a non-digit character
```

Zero Width Assertions

```
\b  Match a word boundary
\B  Match a non-(word boundary)
```

Extensions

```
(?:regexp)  - Group without capture
(?=regexp)  - Zero-width positive lookahead assertion
(?!regexp)  - Zero-width negative lookahead assertion
```

Other

```
[chars]  - Match any of the specified characters
[^chars] - Match anything that is not one of the specified characters
[a-b]    - Match any character in the range between "a" and "b", inclusive
a|b      - Alternation; match either the left side of the "|" or the right
           side
\n       - When 'n' is a single digit: the nth capturing group matched.
```

The @regex_syntax Attribute

The capabilities outlined above are recommended, but are not required for the creation of observable patterns which leverage regular expressions. If a pattern author requires capabilities that are outside of the set defined above, the `@regex_syntax` attribute can be leveraged to identify the regular expression syntax, library, or software required to properly evaluate it (e.g., “PCRE v8.32” or “Java 1.6”). By setting the `@regex_syntax` attribute, the author accepts any adverse effects caused by the deviation with respect to interoperability and compatibility. Omitting this attribute or setting it with an empty value declares alignment with CybOX Language Specification.