

Regex Browsing Controls

Introduction

From version 2.17.9 (back in 2001) NetPilot, and latterly CachePilot and SecurePilot have all had the ability to use Regex (Regular expressions) syntax to add flexibility for the administrator in blocking or allowing particular sites. The following notes explain the syntax further.

Overview and simple example on how to block file types such as MP3

This facility enables the full extended Regex pattern matching on entries in the Global Forbidden Sites list and user created whitelists and blacklists.

The NetPilot Administrator can add the `*/*.*.xxx$(\?|$)` syntax to a blacklist to block particular undesirable file type: e.g.

- `*/*.*.EML(\?|$)` - blocks the file type used by the W32/Nimda I-Worm
- `*/*.*.EXE(\?|$)` - blocks Executable downloads using HTTP
- `*/*.*.MP3(\?|$)` - blocks MP3 music downloads

Or you can use `*/*.*(EXE|EML|MP3)(\?|$)` to block all expressions as one. (Pattern is not case sensitive)

The NetPilot Administrator can add the `*/*.*.xxx$(\?|$)` syntax to a whitelist to only allow particular file types : e.g.

- `*/*.*.HTM(\?|$)` - allows the HTM file type.

Tip: Leave out the first asterix if you want to block file types as part of a specific URL for example: `boo.com/*.*.EXE(\?|$)`

Associated Documents :

Description of Regex www.regular-expressions.info/posix.html

Block/allow particular strings in the domain name

The hostname part of the site list (the part to the left of the first '/') for example www.boo.com from www.boo.com/page.html is a simple global pattern. You can use ``*'` for any number of any character and ``?'` for any one character. The global pattern is anchored on the left to the beginning of the name or a '.', but the right end must match the right end of either the hostname or hostname:port.

Variations on domain name

So given the input strings in the sitelist: `foo.example.com`

These patterns will match if specified a user's browser:

- | | |
|------------------------------|--|
| <code>foo.example.com</code> | will also match <code>www.foo.example.com</code> |
| <code>example.com</code> | will also match <code>wibble.example.com</code> |

NTT1 – Regex Browsing Controls

com	will also match bar.com
???.example.com	will also match baz.example.com
?????.example.com	will also match baza.example.com
foo.example.*	will also match foo.example.co.uk

But these patterns will not match:

foo.example	as Regex matches all the way to the right end
ample.com	as Regex matches on a whole word at the left
?example.com	? will only match exactly one letter

Blocking 'hostname:port' syntax:

foo.example.com:8000	block access to a particular proxy
:8???	blocks ports 8000--8999 on any host
.*	prevents using any port numbers at all

Blocking/Allowing all sites with a particular domain country identifier e.g. .de (German sites)

Because of the implicit '.' on the left and match-from-the-right rule, that could simply be:

de

In the host part, the wildcard '*' matches any number of any character, so this could also be written:

*.de

which is perhaps a little clearer in its intent.

Blocking/Allowing a particular string e.g. 'sex' or 'xxx'

To find the sequence 'xxx' anywhere in the hostname would be:

xxx

If you want to apply this rule only to certain domains it would be:

xxx.com

xxx.de

Looking for keywords in the right-hand part of the URL might be more effective. In that case, the pattern would be:

/.*xxx

More detailed information on Syntax

The path-pattern (the part after the first '/') is a case-insensitive 'extended regular expression'. The format is well-documented and is widely used for many things, like program parsers and text processors. There are lots of features, but these are perhaps the most useful:

. is a wildcard which matches any one character

NTT1 – Regex Browsing Controls

X* matches any number of optional things matching X
\X turns off the special meaning of X
\$ matches the notional end marker

So the example pattern:

```
*/*\EML(?:$)
```

breaks down like this:

/ indicates that this is a path-pattern, not a host-pattern
.* matches any number of any character
\. matches a literal `.`
EML is the literal extension to match
(?:\$) means it that that the pattern can match it when the URL ends or ends with a `.`.

For completeness, the other supported features are:

(X|Y) matches either X or Y
X? X is optional
X+ X is required and may be repeated any number of times
[S] any single character from the set S. S can be just a string of characters, or include ranges `A-B`
[^S] any single character NOT from the set S
X{n} X repeated n times
X{m,n} X repeated between m and n times

So a more compact way to express the example list would be:

```
*/*\.(EML|PIF|SCR|MP3|INI|VBS|VB|GEN)(?:$)
```

NTT1 – Regex Browsing Controls