FCGlob: A New SELinux File Context Syntax

Don Miner, *University of Maryland, Baltimore County*
James Athey, *Tresys Technology*

The Problem with Regular Expressions

- Regular Expressions have counter-intuitive rules for specifying paths
  - "." means "any character", but most filenames have periods. These must be specified with "\."
- Regular Expressions are unstructured for paths
  - The file system is a tree, with "/" having special meaning, but regexes treat "/" the same as any other character
- Regular Expressions are hard to analyze
  - file_contexts is ordered using heuristics, not sorted by specificity

The Problem with Regular Expressions, cont.

- Conflicts between two specifications cannot be detected
  - Ambiguous - two specs are equally specific according to the heuristics, but cover some of the same files
- Regular Expressions can get messy
  - Clever regexes try to fit every possibility in one spec, and bite off more than they can chew

```
/usr/(.*/)?lib(64)?(/.*)?/ld-[^/]*\.so(\.[^/]*)*  system_u:object_r:ld_so_t
/opt/(.*/)?jre.*/libdeploy\.so(\.[^/]*)*      --  system_u:object_r:textrel_shlib_t
```

FCGlob Syntax

- Design goals:
  - Expressive – be able to specify groups of files as required by SELinux
  - Readable – it should be obvious what each spec covers and what the author's intention is
  - Analyzable – a computer should be able to sort by specificity and detect conflicts
  - Learnable – the syntax should be easy for policy authors to pick up

FCGlob Syntax, cont.

- FCGlob syntax borrows heavily from shell globbing
  - / - Directory marker
  - ? - Matches any character
  - * - Wildcard within directories
    - Matches any number of characters except “/”
  - [..] and [low-high] character classes
  - (..|..) - “or”
  - ** - Wildcard for directories
    - Must be specified between slashes, e.g., “../**/..”
    - Only one ** allowed per spec

Examples of FCGlob

- Wildcards cleaned up

```
/usr/(.*/)?lib/.+\.so\.[^/]*     --     system_u:object_r:shlib_t
        becomes
/usr/**/lib/*.so.*               --     system_u:object_r:shlib_t
```

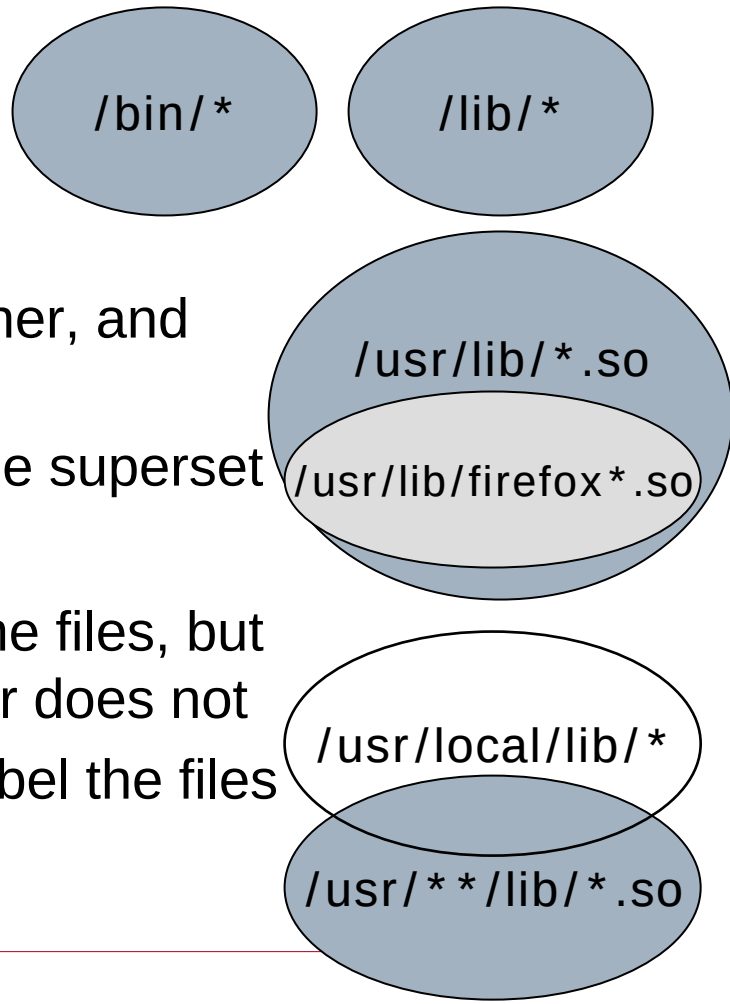
- Intentions clarified

```
/usr/sbin/in\..*d        --     system_u:object_r:inetd_child_exec_t
        becomes
/usr/sbin/in.*d          --     system_u:object_r:inetd_child_exec_t
```

  - Subtle! This regex says that /usr/sbin/in. could be a directory, followed by any directory at all, then any filename ending in d

FCGlob Set Arithmetic

- Two specifications can be
  - Disjoint
    - Each covers different files
  - Superset / Subset
    - One covers all of the files of the other, and then some
    - The subset is more specific than the superset
  - Intersect
    - Each spec covers some of the same files, but each also covers files that the other does not
    - Ambiguous – which spec should label the files covered by both?



FCGlob Set Arithmetic, cont.

- Example: which is more specific, /usr/**, or /usr/share/**/*.so?
  - Step 1 – Make specs the same length

```
/usr/   *   /**/*
/usr/share/**/*.so
```

  - Step 2 – Compare each path component

```
/usr/   *   /**/*
/usr/share/**/*.so
  =     <    =   <
```

  - Result – Each component of the latter is equal or subset of former, so the latter is more specific

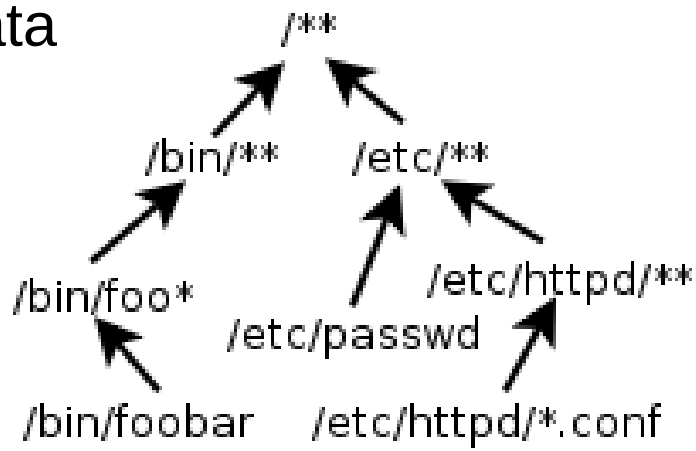
FCGlob Set Arithmetic, cont.

- Conflicts occur when one path component is a subset, but another path component is a superset

```
/etc/ *  /*.conf
/etc/dhcp/*
 =    <     >
```

Other Benefits

- Faster matchpathcon(3)
  - matchpathcon with regular expressions
    - Linear searches through file_contexts
    - Worst case – has to check every single spec
  - matchpathcon with FCGlob
    - file_contexts loaded into a tree data structure
    - Worst case – has to descend to last leaf of longest branch of tree



Other Benefits – cont.

- Visualize filesystem from file_contexts
  - file_contexts is parsed into a tree
- Raise warnings or errors when two specifications conflict
  - Identify which files are ambiguously labeled
  - Deterministic labeling behavior
- Easier module development
  - Before loading a policy module, check for file_context conflicts that might break the policy

TODO List

- FCGlob parser and tree builder
- Reference Policy
  - *.fc translated into FCGlob syntax
  - Compare live filesystems using FCGlob to those using regexes, looking for conversion mistakes
- FCGlob to regex translator
  - Make it possible to use legacy toolchain
- Toolchain updates
  - matchpathcon
  - checking for conflicts at module link time

QUESTIONS?