

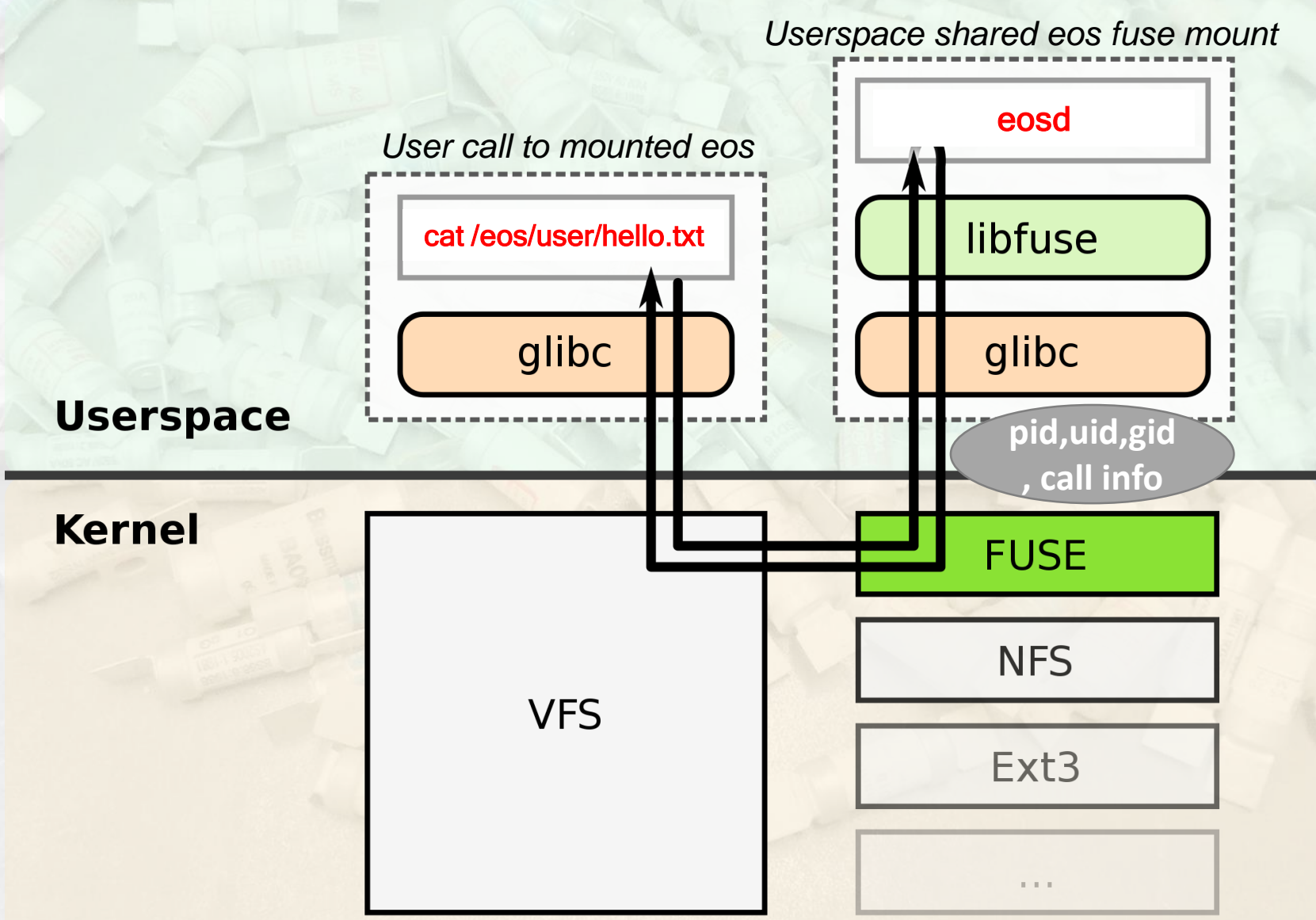
Strong Authentication in EOS Shared Fuse Mount

*EOS Workshop, 02/02/2017
Geoffray Adde, IT-ST-AD*

- Introduction
 - A bit of context
 - Fuse in a sketch
- How it works
 - The new XRootD 4 authentication features
 - Integration into the eos fuse client
 - Getting the (right) users credentials
 - Implementation
- How to use it
 - Configuration
 - eofsusebind and its avatars
- Wrap-up

- Fuse rather than kernel module because much lighter development.
- EOS used to offer two ways to run the fuse mount (it used to be 2 programs):
 - Personal mount : use the krb5 CC or the gsi proxy of the user
 - Shared mount : was using UNIX authentication and had to be granted root access on the mgm
- Important drawbacks
 - Personal mounts are a waste of resource on shared machines
 - Shared mounts are a security threat to the eos instances.
 - Shared mounts have to be manually given special rights on the instance
 - Unix authentication is restrictive in a Grid environment
- One solution : strong authentication in shared fuse mount
 - Prior to XRootd4, the implementation would have been hazardous
 - XRootd4 allows to explicitly pass credentials to be used in urls
 - Let's do it then!
- Ideally, it should feel like using AFS with kerberos

Fuse in a sketch



- Required protocol and credentials can be passed as cgi arguments in the url
root://eosuser.cern.ch/eos/myfile.txt?**xrd.wantprot**=krb5&**xrd.k5ccname**=somefile

...Looks straightforward, easy and great but...

...XRootd client is a smart piece of software:

- It will reuse open connections as much as possible
 - Authentication is done when connecting
 - It will **not** reconnect to conform to the required authentication. If a connection can be reused, it will **ignore** required authentication.
-
- When a new connection is needed, there is no way to force a reconnection, a new one has to be made.
 - Use xrootd login as connection id (root://**a23FIPk0@**eosuser.cern.ch/eos.....)
 - 1 connection per (xroot login,server)
 - xrootd login is 8 characters, if using base64 encoding: 2^{48} .
Enough to avoid reusing connections.
 - Connections left behind will timeout and get closed.

- Using user's environment to specify the credentials to use seems intuitive
 - Implemented! Almost working.
 - **Deadlock** with the kernel..... find something else.....
- Using symlinks created by the user to point to the credentials to use
 - Security to be explicitly taken care of.
 - Implemented!
 - Working .. after some bug fixing
- We call this credentials binding
- Two types of credentials binding
 - User binding : bind a user to default credentials
 - Session binding : allow a user to use specific credentials in a given (linux) session
- Credentials resolution in the daemon
 - Try to find a session binding, if success, done.
 - Try to find a user's binding , if success, done.
 - Optionally : fallback as being nobody , if success, done.
 - Fail

- Coarsely, that is it. A few points had to be watched carefully
- Speed and concurrency:
 - Credentials have to be resolved and checked for any call from any user from any process accessing the fuse mount
 - Several levels of caching
 - Finely crafted locking and mappings
- Security:
 - Credentials safety check is deferred to the underlying library (GSI, Kerberos)
 - Security of symlinks to credentials have to be checked
- OS interaction:
 - Heavy use of proc files
 - All the pid, ppid, sid stuff
- Integration:
 - Tool for the users to create the bindings **eosfusebind**
 - Integration with sshd via pamd, jobs systems and a few servers

- If GSI or KRB5 is enabled, Unix authentication is **disabled**
- GSI and KRB5 **can** be used together
- Using GSI authentication:
 - Grid security should be properly configured on the box i.e. a user should be able to authenticate to eos using the client.
XrdSecPROTOCOL=gsi eos whoami should authenticate the user with gsi
 - *Export EOS_FUSE_USER_GSI_PROXY=1* should be added to */etc/sysconfig/eos*
 - Users should create a proxy with **grid-proxy-init**
- Using KRB5 authentication:
 - Kerberos should be properly configured on the box i.e. a user should be able to authenticate to eos using the client.
XrdSecPROTOCOL=krb5 eos whoami should authenticate the user with krb5.
 - *Export EOS_FUSE_USER_KRB5CC=1* should be added to */etc/sysconfig/eos*
 - *If in-memory krb5 tickets are to be used, Export EOS_FUSE_USER_UNSAFEKRB5= 1* should be added to */etc/sysconfig/eos*
 - Users should create a ticket with **kinit**

- The script manages symlinks of the form:
`/var/run/eosd/credentials/u<uid>.<protocol>`
`/var/run/eosd/credentials/u<uid>_sid<session_id>_sst<session_startuptime>.<protocol>`
- For user bindings, eosfusebind copies (and cleans-up) the credentials to:
`/var/run/eosd/credentials/store/`
- For session bindings, eosfusebind just symlinks the credentials in-place.
- Every time it's called, the script cleans-up the unused symlinks for the user.
- There is locking mechanism for concurrent calls
- The script allows to display the binding for the current session, for the user and for all the other sessions of the same user.
- The script is integrated with pamd for ssh and co.
- The script is also integrated with the wrapping script of kinit
- For very specific symlinks can be created by hand (or should we evolve eosfusebind?)

eosfusebind -h

- Features
 - Shared fuse mount with krb5 and gsi authentication
 - No measurable impact on performance
 - No deployment overhead on eos administration
 - No security risk
 - Easy to use and to integrate **eosfusebind** tool to bind credentials to users/sessions
- Operational experience
 - Running for a **few months** on lxplus (interactive use)
 - Running for a few weeks on lxbatch (batch systems)
 - After many iterations of bug fixing/evolution, the authentication is **now very stable**.
 - Today, most major eos instances are automatically available on lxplus on login.
- Perspectives
 - Long run and expiring credentials
 - Upcoming new fuse implementation does not require high performance. Rewriting without all the caching?

