Security in iRODS

Reagan W. Moore

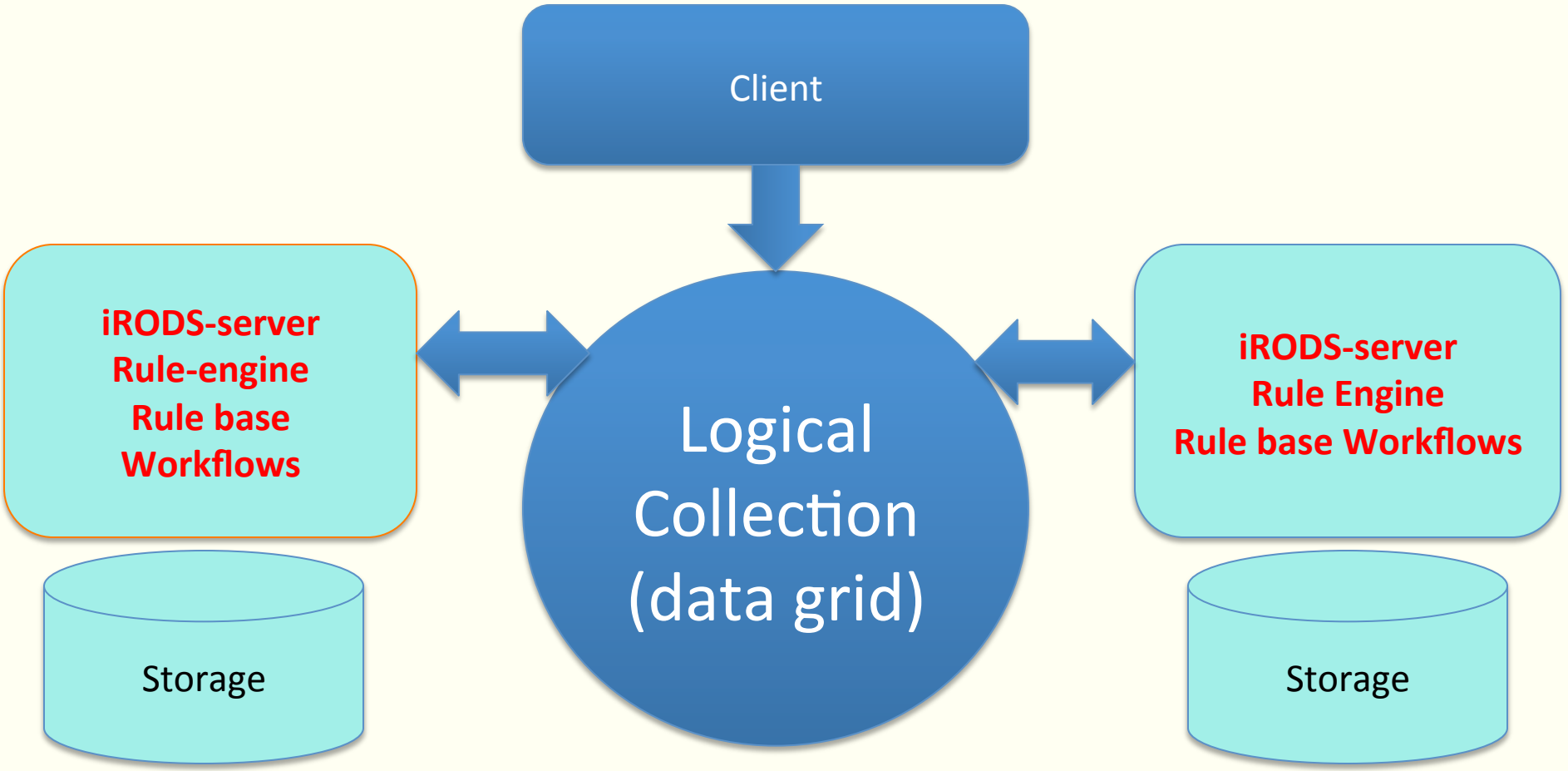
University of North Carolina at Chapel Hill

rwmoore@renci.org


Topics

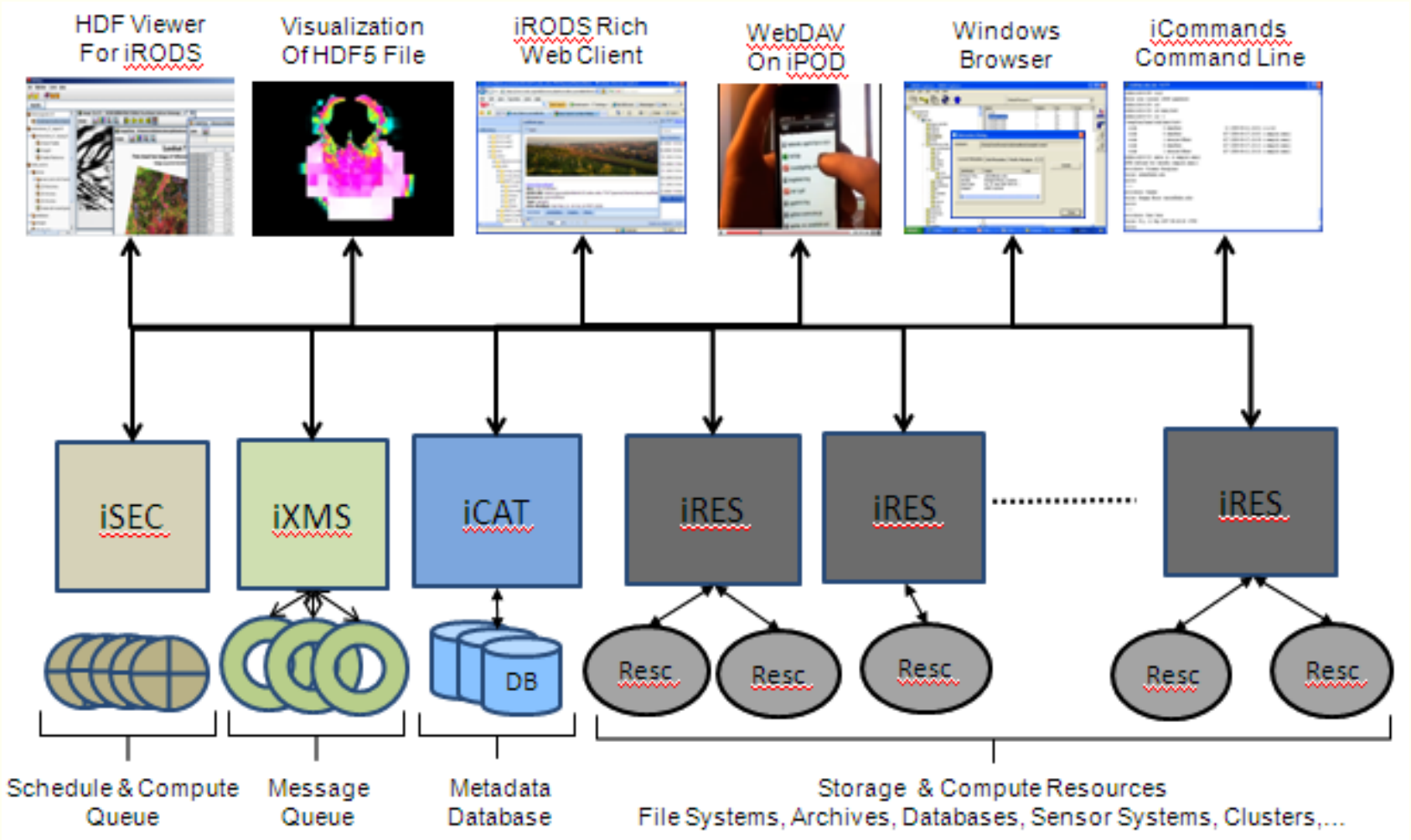
- **Identity management**
- **Authentication**
  - Generic Security Service API - GSSAPI
  - Pluggable Authentication Modules
- **Authorization**
  - Access controls
  - Policy constraints
- **Audit**
- **Vulnerability assessments**

Identity Management



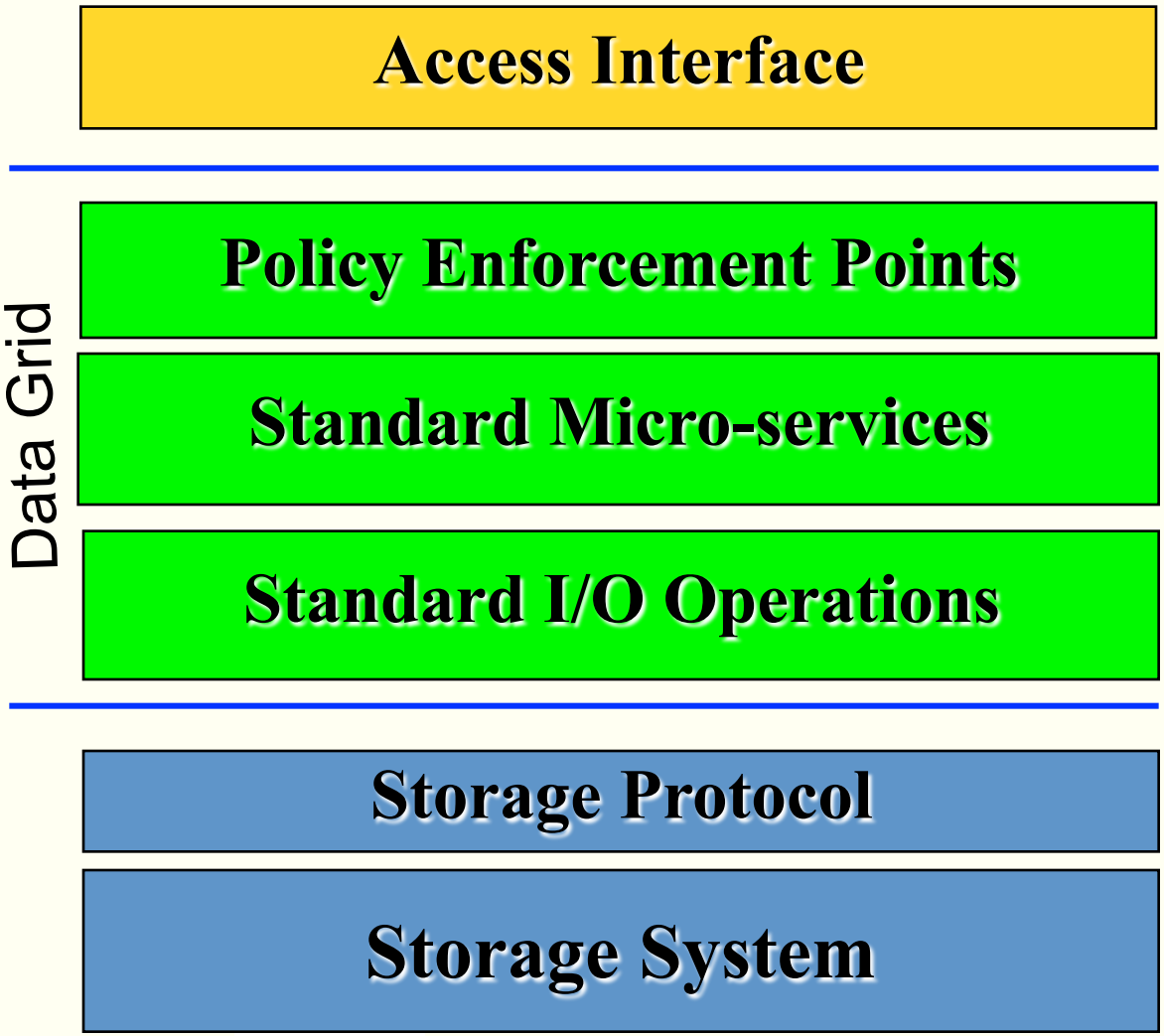
Create Logical Name for each user
Store data under iRODS data grid name

iRODS User Identity Stored in iCAT



D·I·C·E renci UCSD DFC DataNet FEDERATION CONSORTIUM THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL THE NATIONAL ARCHIVES ARCHIVES.GOV NSF

Access Virtualization



Map from the actions requested by the client to multiple policy enforcement points.

Map from policy to standard micro-services.

Map from micro-services to standard Posix I/O operations.

Map standard I/O operations to the protocol supported by the storage system

Identity Management

- **3.1 - ARCS maintained identity in an external certificate authority**
  - When iRODS received a certificate, would create an iRODS logical name to match the certificate
- **3.2 - Pluggable Authentication Modules**
  - Use LDAP to implement the identity management
  - User names are maintained in LDAP server
  - iRODS uses SSL and OpenSSL libraries to exchange information with LDAP

Authentication

- **iRODS data grid 3.1 – GSSAPI**
  - Challenge response – iRODS manages a password for each user
  - Kerberos – certificate authority manages password
  - GSI – certificate authority manages password
  - UK ASPiS system (Architecture for a Shibboleth-Protected iRODS System, UK E-Science)
    - Added attributes to the REI structure for user identification
    - Added rules and micro-services to control access
    - Installed policies at the policy enforcement points for acSetRescSchemeForCreate, acPreprocForDataObjOpen, and acDataDeletePolicy

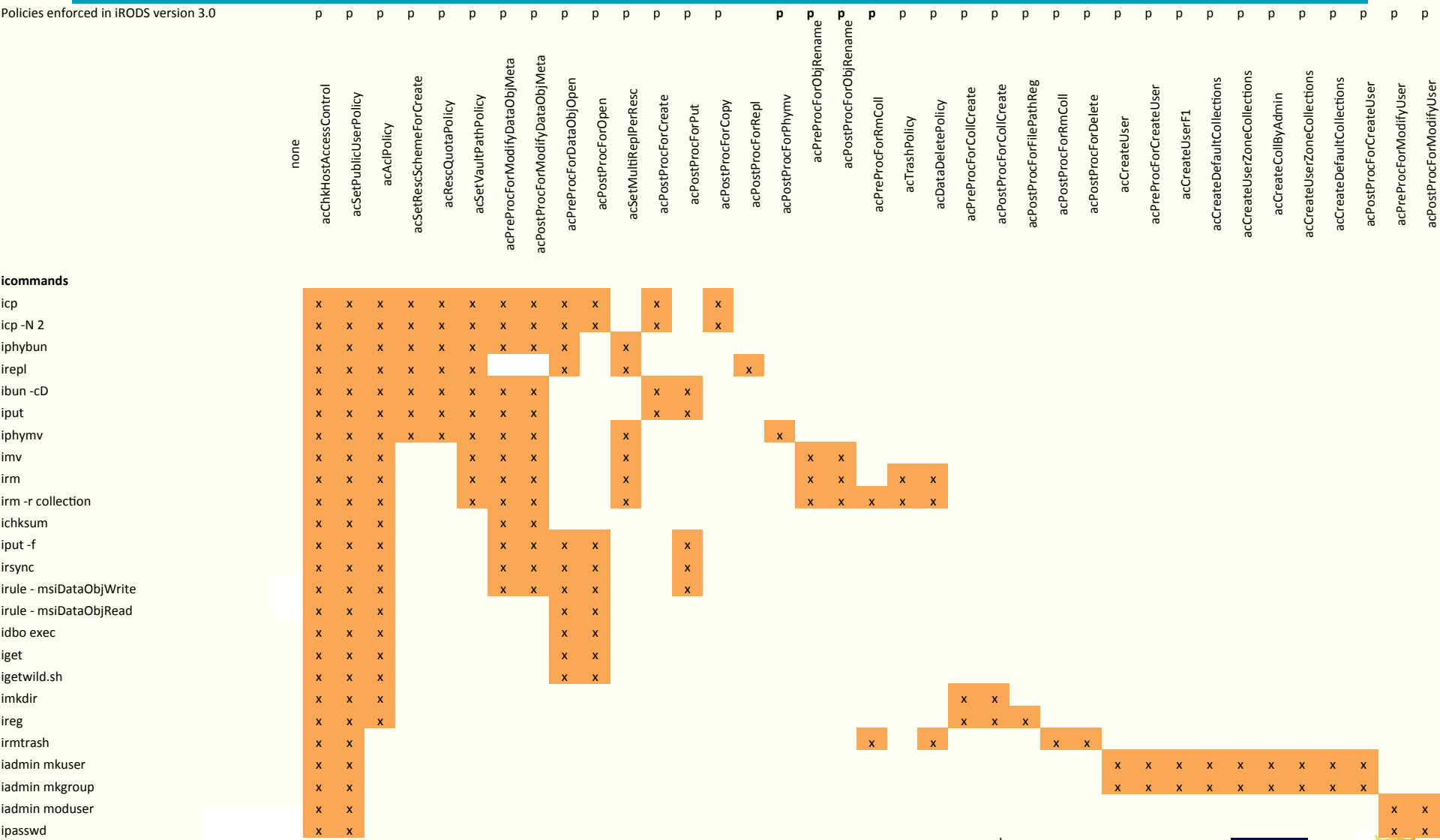
PAM Authentication – iRODS 3.2

- **For additional security, when using PAM (system passwords), 'iinit' will create a separate iRODS password that is then used for the other i-commands (stored in the .irodsA file).**
  - The generated iRODS passwords will be valid for 2 weeks (or other defined period) and can be renewed during that period via another 'iinit' command.
- **Since system passwords are being transferred (and iRODS passwords back), the session for the 'iinit' protocol needs to be encrypted.**
  - This is done via SSL and the OpenSSL libraries. As such, your iRODS server needs to have a proper X.509 certificate for SSL to use. You can use either a self-signed certificate (best for testing) or a certificate from a trusted CA.
- **There is a new iadmin sub-command, 'rpp' (remove PAM-derived Password) for the admin to remove these generated passwords for a specified user if needed**

Authorization

- **Access controls are maintained for each file for users and groups**
  - DATA_ACCESS_DATA_ID
  - DATA_ACCESS_TYPE
  - DATA_ACCESS_USER_ID
  - USER_ID
  - USER_GROUP_ID
- **Each access is processed at policy enforcement points**
  - acChkHostAccessControl
  - acSetPublicUserPolicy
  - acAclPolicy (additional checks for admin in the data grid)

Policy Constraints on Actions



DataNet FEDERATION CONSORTIUM

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Audit Trails

- **Micro-services for parsing audit trails**
  - msiGetAuditTrailInfoByActionID
  - msiGetAuditTrailInfoByKeywords
  - msiGetAuditTrailInfoByObjectID
  - msiGetAuditTrailInfoByTimeStamp
  - msiGetAuditTrailInfoByUserID
- **Turn on audit trail in iRODS/server/icat/src/icatMidLevelRoutines.c**

Example Action IDs for Audit

- **Located in iRODS/server/icat/include/icatDefines.h**
  - ACCESS_GRANTED 1000
  - REGISTER_DATA_OBJ 2010
  - REGISTER_DATA_REPLICA 2011
  - REGISTER_RESOURCE 2030
  - DELETE_USER_RE 2040
  - REGISTER_ZONE 2064
  - MOD_USER_NAME 2070
  - MOD_USER_PASSWORD 2076
  - ADD_AVU_METADATA 2110
  - RENAME_COLLECTION 2131

Example Audit Rule

```
myTestRule {
#Parse audit trails for specific audit action
#  2040  - delete user
  msiGetAuditTrailInfoByActionID(*Id,*Buf,*Status);
  writeBytesBuf("stdout",*Buf);
}
INPUT *Id="2040"
OUTPUT ruleExecOut
```

Vulnerability Assessments

- **Collaboration with Dr. Barton Miller, University of Wisconsin**
  - Built on experience with Storage Resource Broker
  - Analyses were used to improve security of code
- **Question: When vulnerabilities are identified:**
  - Immediately publish?
  - Generate patch for current version and then publish?
  - Include patch in next release and then publish?
  - Retrofit patches to prior versions and then publish?

Conclusion

- **iRODS is middleware and depends upon the security of the underlying operating systems**
  - Users authenticate to iRODS
  - iRODS authenticates to each server / storage system
  - iRODS checks access controls for each file
  - iRODS checks policies that can impose additional controls on actions
- **Assessment policies can check collection properties**
  - Integrity needs to be validated independently of ingestion
  - Audit trails can be parsed to verify compliance over time