Security

in eXist-db

Flickr: CC-BY: security by protohiro

In the Past

eXist-db 1.4 and before

- **Unix like Security Model (attempted)**
  - ‘root’ user => ‘admin’ user
  - ‘wheel’ group => ‘DBA’ group
  - Used ‘rwu’ flags and not ‘rwx’
- **Users and Groups**
  - All stored in an XML file in the database
  - /db/system/users.xml

In the Past

eXist-db 1.4 and before

- **Internally**
  - Each Permission Object (in memory):
    - String for Username
    - String for Password
    - int for Permission mode
    - Validation compares strings and int
  - Stored on disk as (12 bytes):
    - int for user id
    - int for group id
    - int for Permission mode

In the Past

eXist-db 1.4 and before

- **Problems**
  - Validation - String comparison is *slow*
  - Changing XML file could result in inconsistent state
  - No control over execution of XQuery/XSLT
  - Permissions checks incomplete or incorrect
  - No centralised Security Manager
  - No user/group metadata
  - **Too simple – users often built their own!**

So, back to the Lab…

Flickr: CC-BY: two female laboratory workers by BiblioArchives / LibraryArchives

Development was sponsored by:

- Karl-Jaspers Centre, **University of Heidelberg**
- Eric Palmer, **University of Richmond**

- Dmitriy Shabanov and Me (Adam Retter)!

Thank You ☺

Completely Redesigned

eXist-db 2.0

- **Features**
  - Centralised Security Manager*
  - Multiple Realms (e.g. LDAP, OpenID, OAuth)*
  - Unix Style Permissions (performance)
  - Access Control Lists
  - Permissions now follow the Unix Model (rwx)
  - Permissions are correctly enforced
  - Better Password Security
  - Extensible

Centralised Security Manager

- **All Security Tasks managed by Security Manager**
- **All authentication is through the Security Manager**
- **Configurable** (/db/system/security/config.xml)
  - **Configure Realms**
- **Storage**
  - **XML file per-user/group, per-realm**

    e.g. /db/system/security/exist/accounts
    /db/system/security/exist/groups
  - **Transparently synced with in-memory model (safe)**
  - **Metadata for user/group**

Multiple Realms

- **eXist-db Internal Realm (default)**

- **LDAP (supports Microsoft Active Directory also)**
- **OpenID**
- **OAuth**

- **SM will authentication against each realm in turn**

- **User/Group appears in eXist-db, postfixed with '@realm'**
  - e.g. adam.retter@ad.domain.com

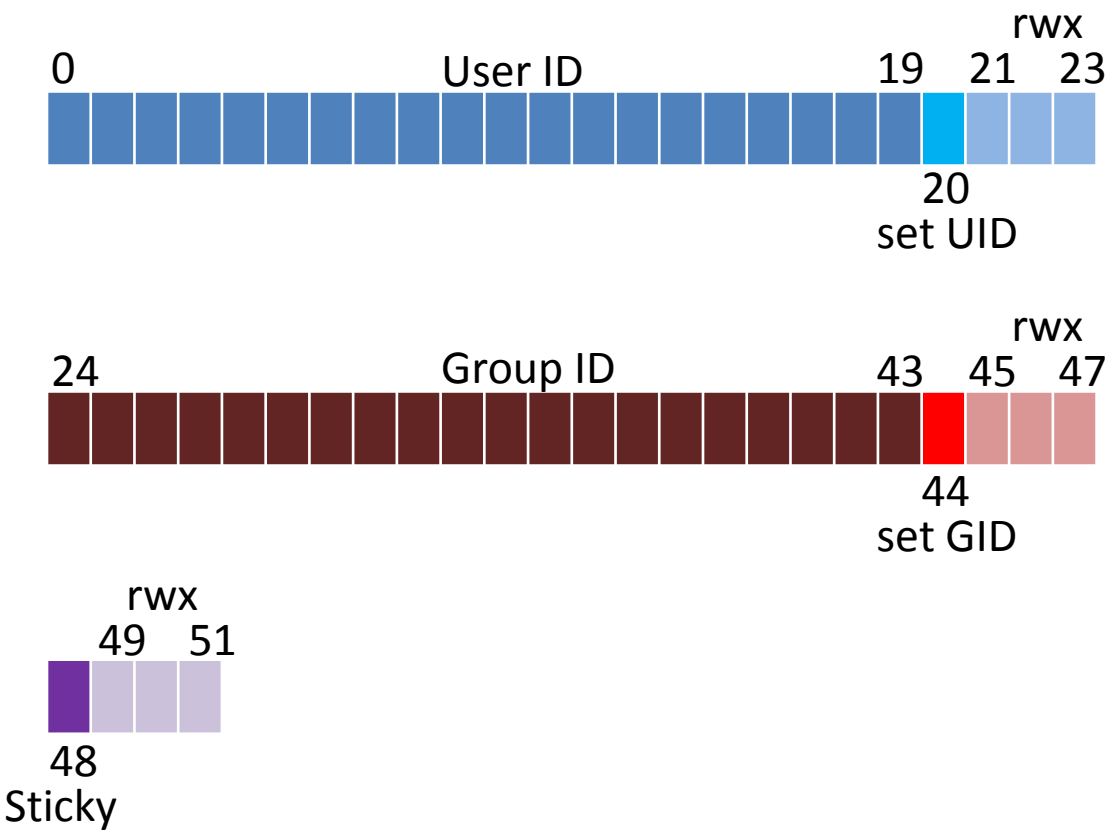
```
<security-manager xmlns="http://exist-db.org/Configuration" last-account-id="13" last-group-id="11" versi
    <realm id="LDAP" principals-are-case-insensitive="true">
        <context>
            <url>ldap://ad.my-domain.de:389</url>
            <domain>ad.my-domain.de</domain>
            <search>
                <base>ou=my-office,dc=ad,dc=my-domain,dc=de</base>
                <default-username>exist@ad.my-domain.de</default-username>
                <default-password>my-password</default-password>
                <account>
                    <search-filter-prefix>objectClass=user</search-filter-prefix>
                    <search-attribute key="name">sAMAccountName</search-attribute>
                    <search-attribute key="dn">distinguishedName</search-attribute>
                    <search-attribute key="memberOf">memberOf</search-attribute>
                    <search-attribute key="primaryGroupID">primaryGroupID</search-attribute>
                    <search-attribute key="objectSid">objectSid</search-attribute>
                    <metadata-search-attribute key="http://axschema.org/namePerson">name</metadata-search
                    <metadata-search-attribute key="http://axschema.org/contact/email">mail</metadata-sea
                </account>
                <group>
                    <search-filter-prefix>objectClass=group</search-filter-prefix>
                    <search-attribute key="name">sAMAccountName</search-attribute>
                    <search-attribute key="dn">distinguishedName</search-attribute>
                    <search-attribute key="primaryGroupToken">primaryGroupToken</search-attribute>
                    <search-attribute key="objectSid">objectSid</search-attribute>
                    <search-attribute key="member">member</search-attribute>
                    <whitelist>
                        <principal>Domain Users</principal>
                    </whitelist>
                </group>
            </search>
            <transformation>
                <add-group>biblio.users</add-group>
            </transformation>
        </context>
    </realm>
</security-manager>
```

Security Manager Co
LDAP Real

Unix Style Permissions (performance)

- **Approach:**
  - **Every single bit counts**
  - **Binary Math is very very fast**
    - Each Permission Object
      - In Memory Model is same as on-disk
      - int for UserId
      - int for GroupId
      - int for Permission mode
      - Limit to 1048575 users and 1048575 groups
      - Validation compares bit masks (very fast!)
      - **TOTAL – 52 bits** (4 spare for future) = 7 bytes.
        - Saved 41% over 1.4.x

Unix Style Permission Bitmap



Just 52 bits!

1.4.x was at *least* 96 bits

Access Control Lists (ACL)

- When Permissions are fast – you can have more!
- Complement the Unix Style Permission
  - Any Collection or Resource may have an ACL
- Resolve many limitations of eXist-db user/group model.

- Access Control Lists consist of Access Control Entry(s) (ACE)
  - ACL may have a maximum of 255 ACEs (just 1 byte itself!)
  - Evaluation of ACEs is top-to-bottom
  - Order in the ACL is significant!

- When present, ACL is evaluated before Unix Style Permission

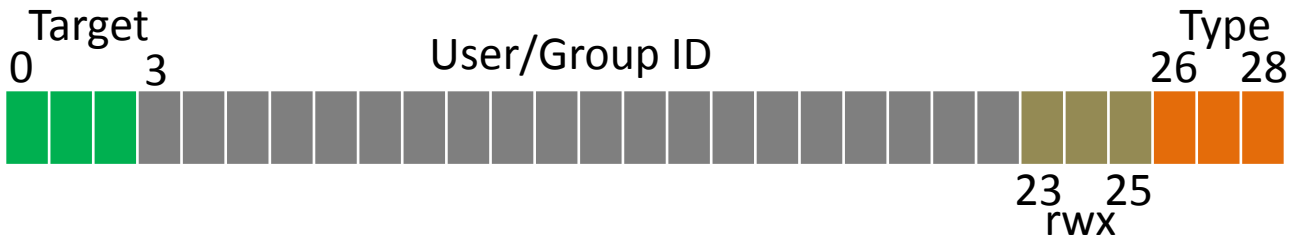
- Very very powerful, but difficult to master!

Access Control Entrys (ACEs)

- **Associates individual group or user with access rights**
- **Consist of:**
  - **Target Type** – User or Group
  - **ID** – The Id of the User or Group targeted
  - **Mode** – Unix like, e.g. rwx
  - **Access Type** – Allowed or Denied
- **TOTAL – 29 bits** (3 spare for future) = 4 bytes.
- **Preserved across backups**
- **Java/Web Admin client shows a “+” if present!**

Access Control Entry Bitmap

Just 29 bits!



Permission XML Serialisation

With ACL and ACEs

```
<sm:permission xmlns:sm="http://exist-db.org/xquery/securitymanager"
  owner="admin" group="dba" mode="rw-------">
  <sm:acl entries="2">
    <sm:ace index="0" target="USER" who="adam" access_type="DENIED" mode="rwx"/>
    <sm:ace index="1" target="GROUP" who="users" access_type="ALLOWED" mode="r-x"/>
  </sm:acl>
</sm:permission>
```

Access Control List Demo

- Examining permissions
  - sm:get-permissions(…)
- Adding ACEs
  - sm:add-user-ace(…)
  - sm:add-group-ace(…)
- Understanding Allowed and Denied Types
- Understanding order of evaluation
- Adding vs. Inserting ACEs

Permission Enforcement

- **eXist-db aligns with Unix Permissions Model**
  - **Including 'rwx', need 'x' for XQuery scripts!**

- **Permissions are enforced by:**
  - **Permissions on Permission via. AOP**
    - **@PermissionRequired Java Annotation**
    - **Secure, woven in at Compile Time!**
  - **Calls to Permission::validate(…)** (TODO move to AOP)
  - **Anyone can use @PermissionRequired** (its simple!)

- **Checked on db operations and credential operations**
- **DBA group user like root/wheel, can be all powerful!**

Changes Required!

Flickr: CC-BY: Steelers Riot. by drewzhrodague

Password Security

- eXist-db internal Realm
  - Passwords are one way hashed for security ☺
  - Previously eXist-db <= 1.4.x used MD5 Hashes
    - MD5 was good several years ago
    - No longer considered secure
  - eXist-db 2.0+ now uses RIPEMD-160 Hashes
    - No Known Weaknesses
    - Why not SHA?
      - SHA-1 is insecure, need SHA-2
      - RIPEMD is public and open, EU funded
      - RIPEMD is just 160 bits, most likely faster?

Questions?

I now drive…

Thanks

Flickr: CC-BY: Anti mob control Motines by Isbi Armour