University of Victoria

GENI Cloud PlanetLab (SFA)

ABAC Integration

David Cheperdak

djbchepe@cs.uvic.ca

Sponsored by the National Science Foundation

Project Objectives

- Develop:
  - ABAC based authentication mechanism for PL
  - API specification (including authorization)
  - automated testing framework
- Integrate:
  - libABAC into PlanetLab

Sponsored by the National Science Foundation

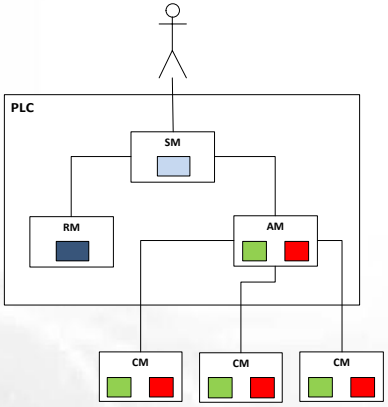
Why ABAC?

1. Automate authorization for users
2. Interoperability
3. Automated delegation
4. Automated agents

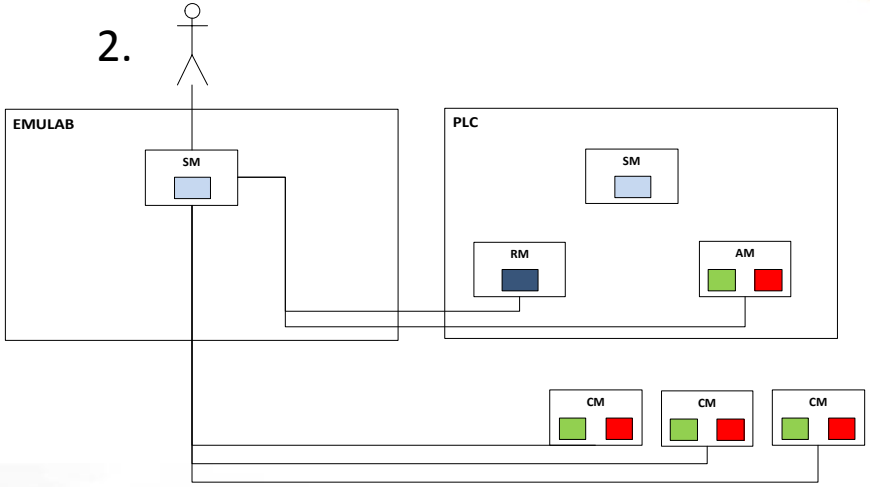
Sponsored by the National Science Foundation

PlanetLab Configurations

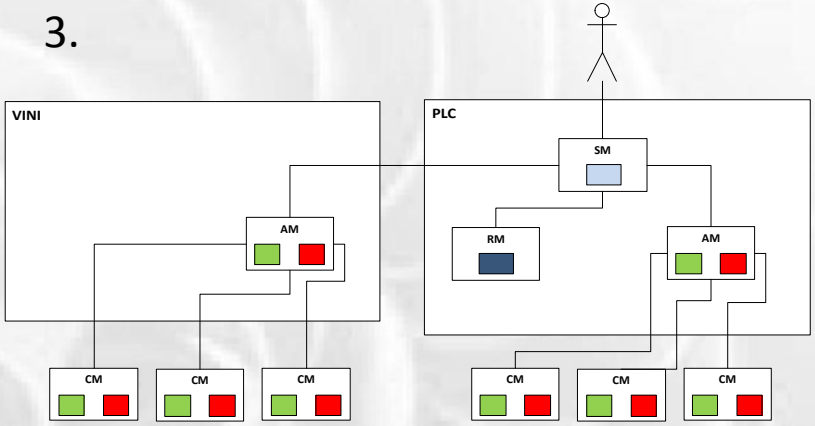
1.



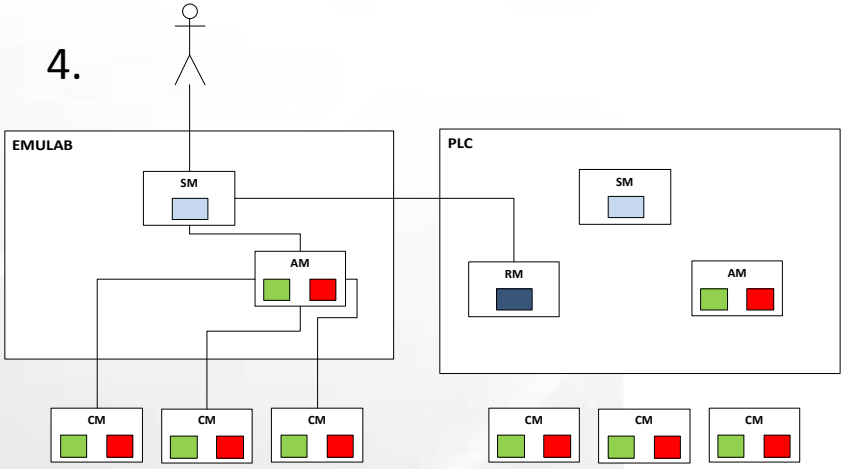
2.



3.



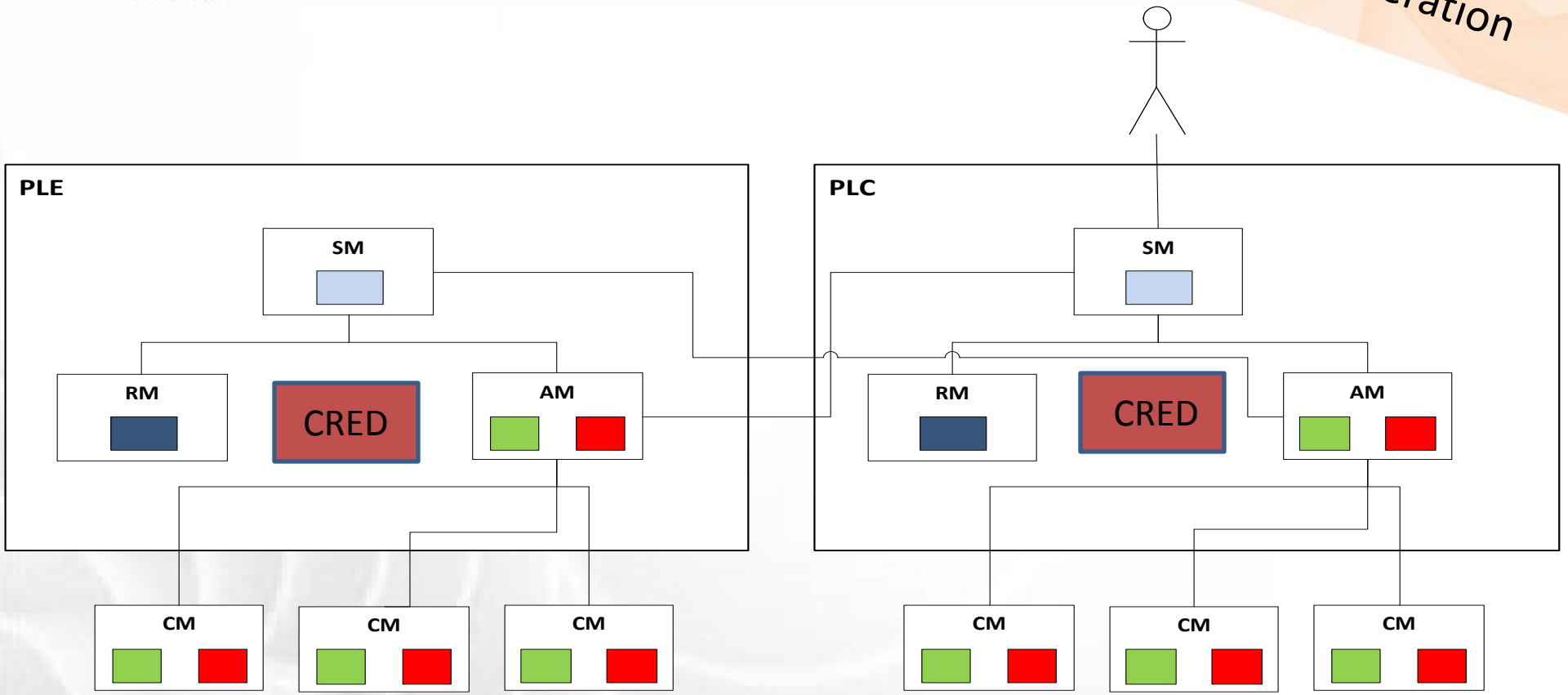
4.



Sponsored by the National Science Foundation

PlanetLab Configurations

Full Federation



SM: Slice Manager

RM: Registry Manager

AM: Aggregate Manager

CM: Component Manager

Color	Name
	Slice Interface
	Registry Interface
	Management Interface
	Research Interface

Sponsored by the National Science Foundation

ABAC Integration Process

- Clearly define and document API:
  - dependencies
  - functionality
  - authorization mechanisms
  - specification
- Verify, repair and standardize existing authorization mechanisms
- Integrate ABAC into PlanetLab
- Test and verify:
  - framework interoperability
  - ABAC functionality
  - PlanetLab functionality
- Analyze:
  - PlanetLab performance

Sponsored by the National Science Foundation

Authorization Overview:

Current

Current authorization process

Client side (Request Action)	Server side (Execute Action)
SSH Key Credentials supplied to SSL XMLRPC	SSH Key Credentials received SSL XMLRPC
Key to Slice association	Key to Slice association
User GID to Cert. verification	User GID to Cert. verification
Perform Action Cert. verification	Perform Action Cert. verification
Cert. Trust Verification	Cert. Trust Verification

Sponsored by the National Science Foundation

Authorization Overview:

Integrated

libABAC authorization process

Client side (Request Action)	Server side (Execute Action)
SSH Key Credentials supplied to SSL XMLRPC	SSH Key Credentials received SSL XMLRPC
	Key to Slice association **(ABAC + PlanetLab)**
	User GID to Cert. verification **(ABAC)**
	Perform Action Cert. verification **(ABAC)**
	Cert. Trust Verification **(ABAC)**

Sponsored by the National Science Foundation

Authorization Mechanisms:

Current

Server Side API: CreateSliver()

```
#API Found in the PlanetLab Aggrgegate Manager
def CreateSliver(api, xrn, creds, rspec_str, users, call_id):
    . . . Continued . . .
   if not credential:
        credential = api.getCredential()
    hrn, type = urn_to_hrn(xrn)
    valid_cred = api.auth.checkCredentials(creds, 'createsliver', hrn)[0]
    caller_hrn = Credential(string=valid_cred).get_gid_caller().get_hrn()
    threads = ThreadManager()
    for aggregate in api.aggregates:
        if caller_hrn == aggregate and aggregate != api.hrn:`
            continue
        server = api.aggregates[aggregate]
     threads.run(_CreateSliver, aggregate, server, xrn, credential, rspec.toxml(),
users, call_id)
    . . . Continued . . .
```

- Authorization by SSH Key:
  - confirmation must occur every time the API is called

Sponsored by the National Science Foundation

Authorization Mechanisms:

Current

- Every manager within SFA PlanetLab:
  - Requires comprehensive authorization checking
  - Utilize a variety of authorization libraries
- ABAC Integration
  - Standardizes authorization mechanism

Authorization Mechanisms:

Current

Server Side API: CreateSliver()

```
def checkCredentials(self, creds, operation, hrn = None):
        valid = []
        # check if a credential is associated with an instance
         if not isinstance(creds, list):
            creds = [creds]
          for cred in creds:
            try:
               # authorize operation by a particular user on a slice
                 self.check(cred, operation, hrn)
               valid.append(cred)
            except:
               cred_obj=Credential(string=cred)
                continue
        if not len(valid):
             raise InsufficientRights('Access denied: %s -- %s' %
(error[0],error[1]))
```

- Authorization by SSH Key:
  - preliminary authorization mechanisms including the presence of a certificate associated with the instance

Sponsored by the National Science Foundation

Authorization Mechanisms:

Current

- Every manager within SFA PlanetLab:
  - Does not support attribute driven resource association
  - Every user must be bound to a slice instead of facilitating a generic association such as role
- ABAC Integration
  - Supports generic role driven association

Authorization Mechanisms:

Server Side API: CreateSliver()

```
def check(self, cred, operation, hrn = None):
        self.client_cred = Credential(string = cred)
        self.client_gid = self.client_cred.get_gid_caller()
        self.object_gid = self.client_cred.get_gid_object()
                    ...Continued...
         # verify the client_gid matches client's certificate
        if self.peer_cert:
            self.verifyPeerCert(self.peer_cert, self.client_gid)
         # validate client authorization to perform operation
        if operation:
                if not self.client_cred.can_perform(operation):
                    raise InsufficientRights(operation)
         # verify the certificate signature
        if self.trusted_cert_list:
            self.client_cred.verify(self.trusted_cert_file_list,
self.config.SFA_CREDENTIAL_SCHEMA)
         else:
            raise MissingTrustedRoots(self.config.get_trustedroots_dir())
                    ...Continued...
        return True
```


Sponsored by the National Science Foundation

Authorization Mechanisms:

Current

- Every manager within SFA PlanetLab:
  - User groups are used to verify slice association
  - A User must be verified if they can perform and action based on a unique key
  - User credentials must be checked if it is trusted
- ABAC Integration
  - User groups are replaced by roles that can be easily allocated and de-allocated
  - Performing actions within PL now support role driven verification

Authorization Mechanisms:

Integrated

libABAC Integration

```
#ABAC Library Calls from Python
def authorize(self, principal):
      store = CredentialManager()
      context = Context()
      context.load_directory(keystore)
      # verify the certificate signatures, obtain user role and
permissions
       (success, credentials) = context.query(role, principal)
if success:
      for cred in credentials:
```

```
//Determine if principal possesses role  if so return a proof of that,
otherwise return a partial proof of it
public QueryResult query( String role, String principal)
 {
          derive_implied_edges();
          Query q = new Query(g);
          Graph<Role, Credential> rg = q.run(role, principal);
/* return all credentials (edges) and boolean if the query found
           principle vertices */
          return new QueryResult(rg.getEdges(), q.successful());
 }
```


Sponsored by the National Science Foundation

Conclusions

deter ABAC Integration

- **Simplifies authorization procedures:**
  - bundling authority to a given certificate (Credential)
  - bundling a proof of identity, authorization and role with a certificate(Credential)
- **Authorization reduces authorization complexity by:**
  - eliminating confirmation of identify, role and permissions client side
- **Improves security by:**
  - Standardizing security and authorization mechanisms across frameworks
  - Enforcing role, authority and identity through signed certificates
  - Scalability as permissions, role and association among users change
  - Easy to revoke an identity without modifying global permissions

Sponsored by the National Science Foundation

Timeline

- September to October 2011
  - System analysis (Omni, PlanetLab, Emulab)
  - Prototyping, documentation and UML
- November 1st to Nov 30th
  - Final integration, expanding authorization features
  - Initial development of automated tester (Draft test Cases)
- December 1st to December 20th
  - Automated testing (Finalize test cases, verification)
  - Performance analysis (PlanetLab and ABAC)

Sponsored by the National Science Foundation

Future Work

Development:

- Automated Tester
  - Integration testing
  - ABAC Verification
- ABAC Integration
  - Integrate ABAC into all essential API within PlanetLab
- PlanetLab
  - Ensure PlanetLab API incorporate functionally correct authorization mechanisms
  - Remove redundant or dead code
- Specification
  - API specification will be developed

Sponsored by the National Science Foundation

Questions and Feedback

- Source Contribution:
  - Commit to source tree
- Questions?
- Feedback?
  - Reasonable?
  - Present errors?
  - Other ABAC project deadlines and goals?
  - Inter-platform ABAC testing?

David Cheperdak

djbchepe@cs.uvic.ca

Sponsored by the National Science Foundation