# Blockchain Development
# Week: 3
# Title: Access Control

Dr Ian Mitchell

Middlesex University,
Dept. of Computer Science,
London

September 26, 2019



## Module Aims

### Aims

Apply and develop Access Control strategies for blockchain.



## Module Objectives

### Knowledge

- Implement Blockchain ACL
- Role-based access control
- Atribute based access control
- Apply different strategies of access control
- Control the authorisation of Participant's access to assets

### Skills

Develop and implement access control for blockchain applications

## Mandatory Access Control (MAC)

- Levels

**MAC**



## Role-Based Access Control (RBAC)

- Academics, Students, Admin, Management, External
- All have different access to Systems
- M:N relationships between users and rights
- users cannot pass access permissions on to other users
- form of mandatory access control
- not multilevel

**RBAC**

A means of restricting access to objects based on the sensitivity of the information contained within the objects and the formal authorisation of subjects to access information of such sensitivity [1]



## RBAC

- What is a Role?
- Set of transactions performed for access
- Transactions are allocated roles by SysAdmin
- Membership of a role
- Academics, Students, Admin, Management, External

**Exam paper: Do's and Don'ts [2]**

- Module Leader writes exam paper.
- Internal moderator reviews exam paper.
- External Examiner checks process
- Module Leader responds
- Administrator signs-off
- Students complete exam

## RBAC

- What is a Role?
- Set of transactions performed for access
- Transactions are allocated roles by SysAdmin
- Membership of a role
- Academics, Students, Admin, Management, External
- **Role Explosion**

**Exam paper: Do's and Don'ts [2]**

- Module leader reviews submitted paper.
- Internal moderator submitting paper.
- External Examiner accessing incorrect papers
- Admin author paper
- Students views paper



## Attribute-Based Access Control



## Hyperledger - Access Control Language

**Review**

- Permissioned blockchain
- Membership Services Provider (MSP)
- Fabric Certificate Authority (FCA)
- FCA issues Enrollment Certificates (e-certs)
- The e-cert is used as a signature
- user must register for e-cert

**Attribute-based Access Control (ABAC)**

- Fabric supports ABAC
- access control based on the attributes associated with the user identity
- Assets
- Participants
- Transactions
- Events
- Business Networks

> *A business network is a collection of participants and assets than [sic] undergo a life cycle described*

# Access Control Language

## Components

- Resources
  - namespace: org.example.*
  - namespace(recursive):org.example.**
  - Class in a ns: org.example.className
  - Instance of a class: org.example.className#ID
- operation
- participant
- transaction
- condition
- action



# Access Control Language, ACL

## Simple Rules

- Rules
- users
- permission
- create
- read
- update
- delete
- evaluated in order, first rule that matches is executed
- no matches, then access is denied
- no rules, no access

**Listing**

```
1 rule {
2   description:
3   participant:
4   operation:
5   resource:
6   action:
7 }
8
```



# ACL

## Simple Rules

**Listing**

```
1 rule ruleName {
2   description:" a brief descript of the
        rule"
3   participant:"namespace.participant"
4   operation:
5   resource:
6   action:
7 }
8
```

**Listing**

```
1 rule ruleName {
2   description:" a brief descript of the
        rule"
3   participant:"namespace.participant"
4   operation:ALL,CREATE,DELETE,READ,UPDATE
5   resource:"namespace.Asset"
6   action:ALLOW,DENY
7 }
8
```

# Access Control Language, ACL

## Conditional Rules

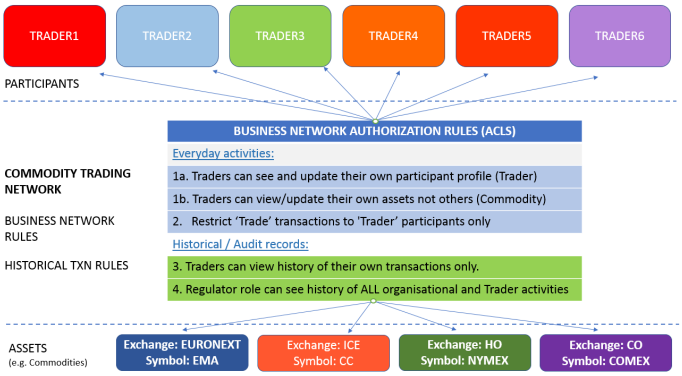
- Rules
- Trader example
- trader 1: current owner / seller
- trader 2: new owner / buyer
- only an owner of an asset should be able to sell it

### Listing

```
1  rule ownerOnlyAccess{
2    description: "Owners get full access and
         the right to sell"
3    participant(p): "org.example.trading.
         Trader"
4    operation: ALL
5    resource(r): "org.example.trading.
         Commodity"
6    condition: (p==r.owner)
7    action:ALLOW
8  }
9
```



# Case Study - Trader Network






# Rule Implementation

## Trader Network

- Rules are generic and allow all access
- Add 6 traders to the network
- Add more commodities to the network
- Look at rules
- Remove this rule
- keep the admin rules (on next slide)

### Listing

```
17     description: "Allow all participants
         access to all resources"
18     participant: "ANY"
19     operation: ALL
20     resource: "org.example.trading.*"
21     action: ALLOW
22 }
23 */
```

# Trader Network I

## Permissions.acl

```
24 rule traderSeeThemselvesOnly{
25   description: "Trader can only see themselves"
26   participant(p):"org.example.trader.Trader"
27   operation: READ, UPDATE
28   resource(r): "org.example.trader.Trader"
29   condition: (p.getIdentifier() == r.getIdentifier())
30   action: ALLOW
31   }
32 rule traderUpdateReadTheirCommodities{
33   description: "trader can see/sell/update their own commodities"
34   participant(p): "org.example.trader.Trader"
35   operation: ALL
36   resource(r): "org.example.trader.Commodity"
37   condition: (p.getIdentifier()==r.owner.getIdentifier())
38   action: ALLOW
39 }
40 rule traderToSubmitTX{
41   description:"Enable Traders to trade, submit transactions"
42   participant:"org.example.trader.Trader"
43   operation:ALL
44   resource:"org.example.trader.Trade"
45   action: ALLOW
46   }
47
48 rule SystemACL {
49     description:  "System ACL to permit all access"
50     participant: "org.hyperledger.composer.system.Participant"
51     operation: ALL
```



# Trader Network II

## Permissions.acl

```
52     resource: "org.hyperledger.composer.system.**"
53     action: ALLOW
54 }
55
56 rule NetworkAdminUser {
57     description: "Grant business network administrators full access to user resources"
58     participant: "org.hyperledger.composer.system.NetworkAdmin"
59     operation: ALL
60     resource: "**"
61     action: ALLOW
62 }
63
64 rule NetworkAdminSystem {
65     description: "Grant business network administrators full access to system resources"
66     participant: "org.hyperledger.composer.system.NetworkAdmin"
67     operation: ALL
68     resource: "org.hyperledger.composer.system.**"
69     action: ALLOW
70 }
```



# Rule

## Traders can only see themselves

- If we have six traders, should traders be able to see each other?
- So here is a rule that allows traders to see themselves only
- This is a condition.
- The solution slightly abuses the resource
- A resource does not have to be an asset, it can also be a participant
- Condition is when the identifiers are equal allow updates and reads

```
1 rule traderSeeThemselvesOnly{
2   description: "Trader can only see
      themselves"
3   participant(p):"org.example.trader.
      Trader"
4   operation: READ, UPDATE
5   resource(r): "org.example.trader.Trader"
6   condition: (p.getIdentifier() == r.
      getIdentifier())
7   action: ALLOW
8   }
9
```

## Rule

Traders can only trade assets they own

- You should not be able to trade something you don't own
- rule is to allow traders to only sell assets they own
- This is a more traditional condition
- participant: trader
- resource: commodity (asset)
- condition: traderID == owner ID

```
rule traderUpdateReadTheirCommodities{
  description: "trader can see/sell/update their own commodities"
  participant(p): "org.example.trader.Trader"
  operation: ALL
  resource(r): "org.example.trader.Commodity"
  condition: (p.getIdentifier()==r.owner.getIdentifier())
  action: ALLOW
}

```



## Rule

Traders To Submit Transactions

- allow access to transaction, trade
- participant: trader
- resource: trade (transaction)
- condition: none
- action: allow
- operation: all

```
rule traderToSubmitTX{
  description:"Enable Traders to trade, submit transactions"
  participant:"org.example.trader.Trader"
  operation:ALL
  resource:"org.example.trader.Trade"
  action: ALLOW
  }

```



## Case Study - Line of Credit

## Summary

- Top-to-bottom evaluation
- most specific to least specific
- As soon as participant, operation and resource match then subsequent rules are not executed
- If no ACL rule fires then AC is denied



## References I

[1] David F Ferraiolo et al. "Proposed NIST standard for role-based access control". In: *ACM Transactions on Information and System Security (TISSEC)* 4.3 (2001), pp. 224–274.

[2] I. Mitchell and S. Hara. "BMAR - blockchain for medication administration records". In: *Blockchain and Clinical Trial - Securing Patient Data*. Ed. by H. Jahankhani. Advanced Sciences and Technologies for Security Applications. Springer, 2019.



## Web Resources

- `http://hyperledger.org`