HGABAC: Towards a Formal Model of Hierarchical Attribute-Based Access Control

Daniel Servos
dservos5@uwo.ca

Sylvia L. Osborn
sylvia@csd.uwo.ca

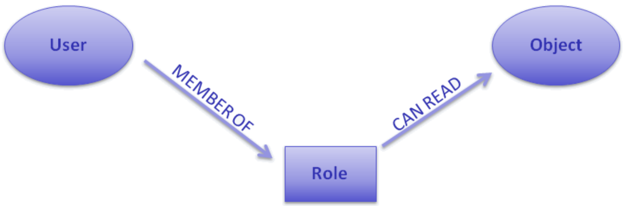
Western
UNIVERSITY · CANADA
Department of Computer Science

The 7th International Symposium on Foundations & Practice of Security, November 2014

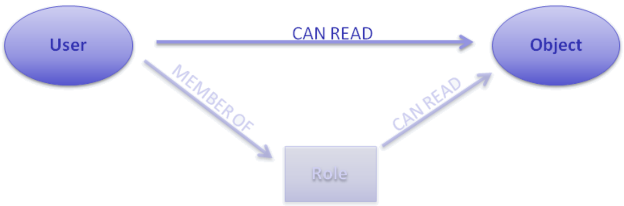
Background

Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC)

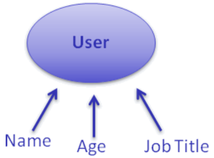


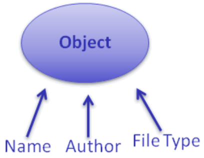
Role-Based Access Control (RBAC)



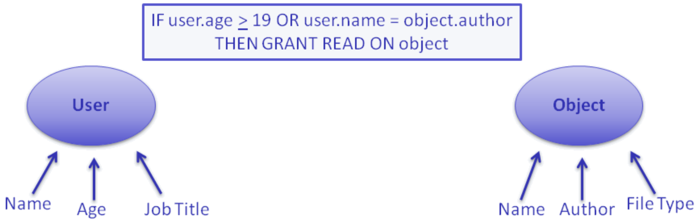
Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC)

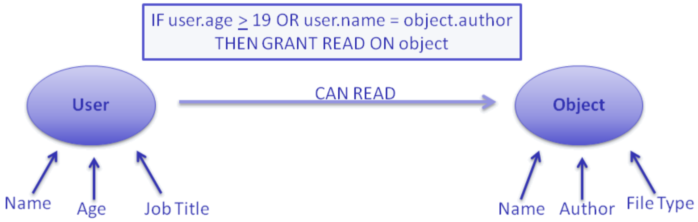




Attribute-Based Access Control (ABAC)


IF user.age ≥ 19 OR user.name = object.author
THEN GRANT READ ON object
User
Name
Age
Job Title
Object
Name
Author
File Type

Attribute-Based Access Control (ABAC)


IF user.age ≥ 19 OR user.name = object.author
THEN GRANT READ ON object
User
CAN READ
Object
Name
Age
Job Title
Name
Author
File Type

Related Work & Current Models

Comparison of Notable Models of Attribute-Based Access Control

	Logic-based Framework for ABAC	ABAC$_\alpha$	ABAC for Web Services	WS-ABAC	ABMAC
Hierarchical	Hierarchical attributes	✗	✗	✗	✗
Object Attributes	✗	✓	✓	✓	✓
User Attributes	✓	✓	✓	✓	✓
Environment Attributes	✗	✗	✓	✓	✓
Connection Attributes	✗	✗	✗	✗	Shown in example but not model
Administrative Attributes	✗	✗	✗	✗	✗
Separation of Duties	✗	✗	✗	✗	✗
General Model	✓	✓	For web services	For web services	For grid computing
Formal Model	Only models policies and evaluation	✓	Simplistic	Simplistic	✓
Administrative Model	✗	Limited	✗	✗	✗
Can Model DAC, MAC, and RBAC	Not demonstrated	✓	Not demonstrated	Not demonstrated	Not demonstrated

Related Work & Current Models

Comparison of Notable Models of Attribute-Based Access Control

	Logic-based Framework for ABAC	ABAC$_\alpha$	ABAC for Web Services	WS-ABAC	ABMAC
Formal Model	policies and evaluation	✓	Simplistic	Simplistic	✓
Administrative Model	✗	Limited	✗	✗	✗
Can Model DAC, MAC, and RBAC	Not demonstrated	✓	Not demonstrated	Not demonstrated	Not demonstrated

Open Problems

- Lack of hierarchical structures comparable to RBAC.
- Lack of group based administration of multiple users.
- Limited work towards a separation of duties model for ABAC.
- Limited work towards a administrative model of ABAC.
- Auditability of ABAC systems.
- Need for formal foundational models of ABAC.

Related Work & Current Models

Comparison of Notable Models of Attribute-Based Access Control

	Logic-based Framework for ABAC	ABAC$_\alpha$	ABAC for Web Services	WS-ABAC	ABMAC
Formal Model	policies and evaluation	✓	Simplistic	Simplistic	✓
Administrative Model	✗	Limited	✗	✗	✗
Can Model DAC, MAC, and RBAC	Not demonstrated	✓	Not demonstrated	Not demonstrated	Not demonstrated

Open Problems

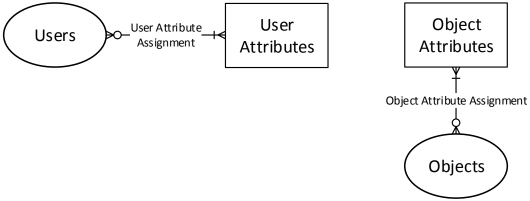
- **Lack of hierarchical structures comparable to RBAC.**
- **Lack of group based administration of multiple users.**
- Limited work towards a separation of duties model for ABAC.
- Limited work towards a administrative model of ABAC.
- Auditability of ABAC systems.
- **Need for formal foundational models of ABAC.**

Related Work & Current Models

Comparison of Notable Models of Attribute-Based Access Control

	Logic-based Framework for ABAC	ABAC$_\alpha$	ABAC for Web Services	WS-ABAC	ABMAC	HGABAC
Hierarchical	Hierarchical attributes	✗	✗	✗	✗	✓
Object Attributes	✗	✓	✓	✓	✓	✓
User Attributes	✓	✓	✓	✓	✓	✓
Environment Attributes	✗	✗	✓	✓	✓	✓
Connection Attributes	✗	✗	✗	✗	Shown in example but not model	✓
Administrative Attributes	✗	✗	✗	✗	✗	✓
Separation of Duties	✗	✗	✗	✗	✗	✗
General Model	✓	✓	For web services	For web services	For grid computing	✓
Formal Model	Only models policies and evaluation	✓	Simplistic	Simplistic	✓	✓
Administrative Model	✗	Limited	✗	✗	✗	✗
Can Model DAC, MAC, and RBAC	Not demon-strated	✓	Not demon-strated	Not demon-strated	Not demon-strated	✓

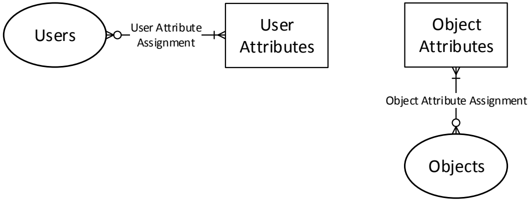
HGABAC Model



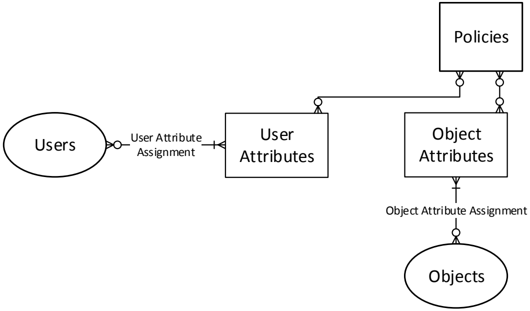
HGABAC Model

Attributes

$$attr = (name, type, value)$$



HGABAC Model



Policy Language

- Three-valued logic (True, False and Undefined).
- Boolean statements using AND, OR, and NOT logical operations.
- AND, OR and NOT truth tables from Kleene K3 logic.
- Support for value and set comparison operations $<$, $>$, $\leq$, $\geq$, $=$, $\neq$, $\in$, $\subset$, etc.

Policy Language

- Three-valued logic (True, False and Undefined).
- Boolean statements using AND, OR, and NOT logical operations.
- AND, OR and NOT truth tables from Kleene K3 logic.
- Support for value and set comparison operations $<$, $>$, $\leq$, $\geq$, $=$, $\neq$, $\in$, $\subset$, etc.

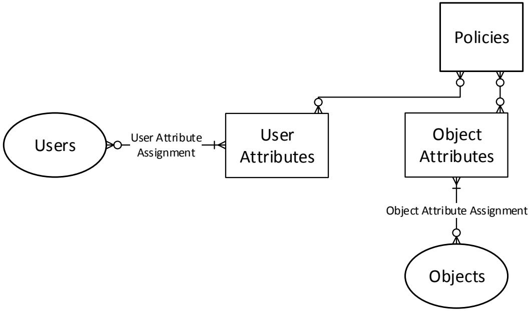
Examples

(a) *user.id IN* $\{5, 72, 4, 6, 4\}$ *OR user.id* $=$ *object.owner*

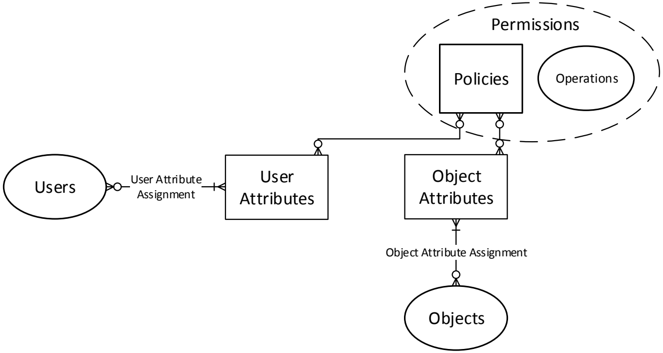
(b) *object.required_perms SUBSET user.perms AND user.age* $>=$ *18*

(c) *user.admin OR (user.role* $=$ *"doctor" AND user.id* $!=$ *object.patient)*

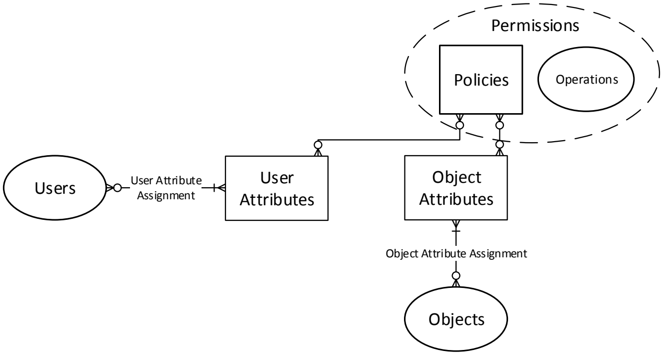
HGABAC Model



HGABAC Model



HGABAC Model

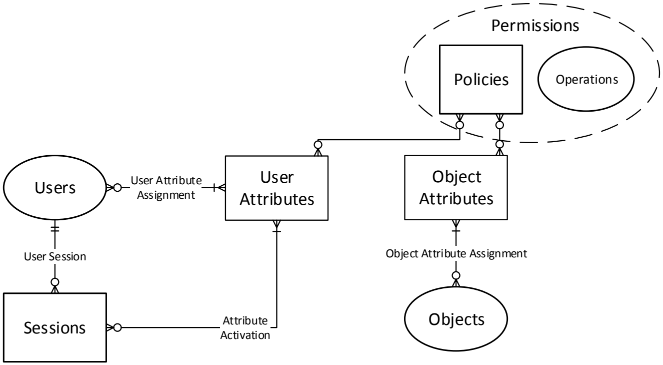


Permissions

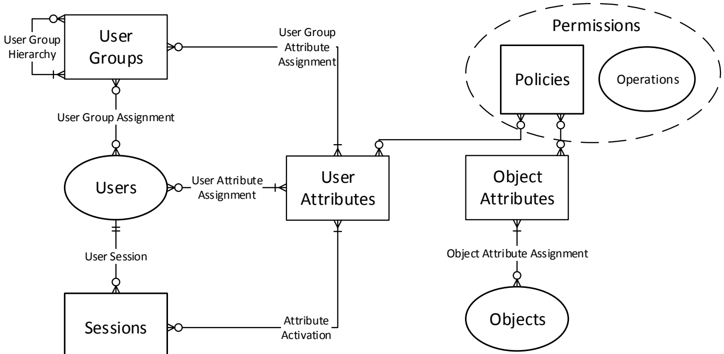
$user.id = object.patient\ OR\ user.role =$ *"doctor"* $\rightarrow$ **read**

$user.role =$ *"doctor"* $\rightarrow$ **write**

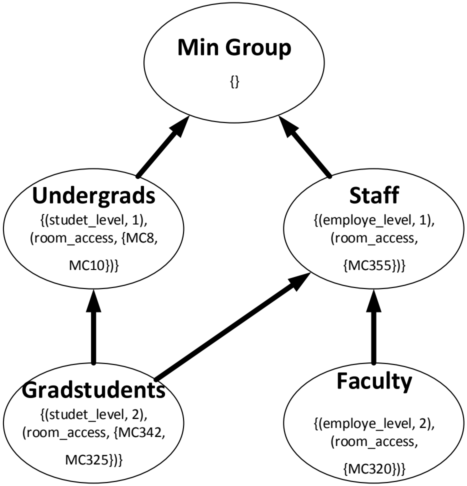
HGABAC Model



HGABAC Model

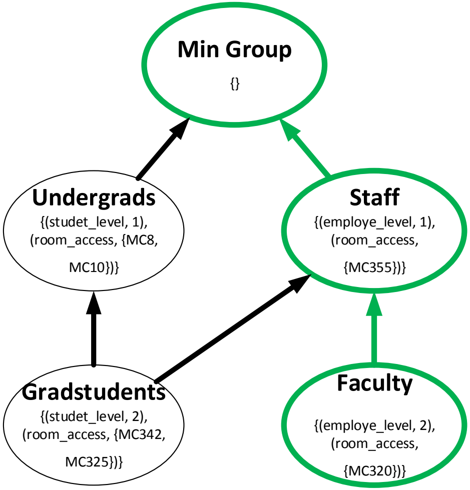


Group Graph



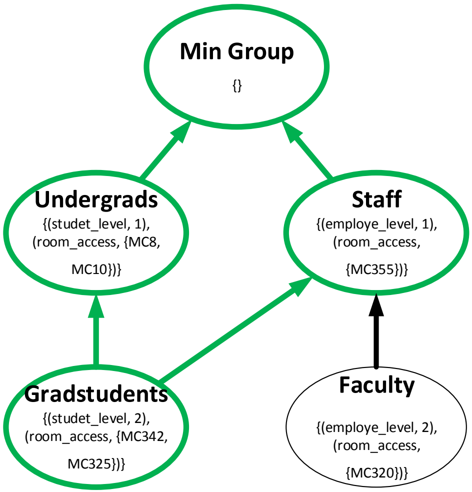
Min Group
{}
Undergrads
{(studet_level, 1), (room_access, {MC8, MC10})}
Staff
{(employe_level, 1), (room_access, {MC355})}
Gradstudents
{(studet_level, 2), (room_access, {MC342, MC325})}
Faculty
{(employe_level, 2), (room_access, {MC320})}

Group Graph


Min Group
{}
Undergrads
{(studet_level, 1), (room_access, {MC8, MC10})}
Staff
{(employe_level, 1), (room_access, {MC355})}
Gradstudents
{(studet_level, 2), (room_access, {MC342, MC325})}
Faculty
{(employe_level, 2), (room_access, {MC320})}

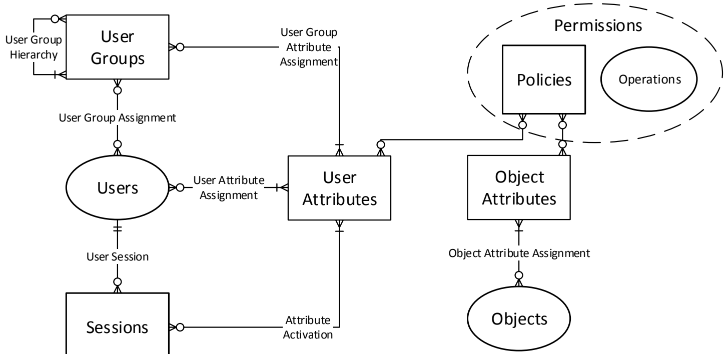
Effective: employe_level = {1, 2}
room_access = {MC355, MC320}

Group Graph

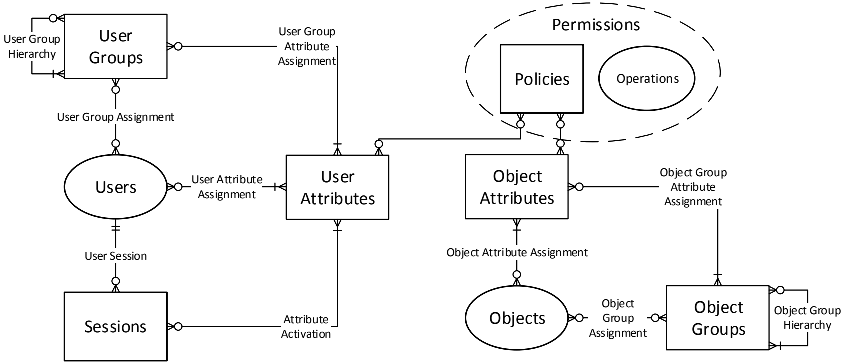

Min Group
{}
Undergrads
{(studet_level, 1), (room_access, {MC8, MC10})}
Staff
{(employe_level, 1), (room_access, {MC355})}
Gradstudents
{(studet_level, 2), (room_access, {MC342, MC325})}
Faculty
{(employe_level, 2), (room_access, {MC320})}

Effective: employe_level = {1}
student_level = {1,2}
room_access = {MC8, MC10, MC355, MC342, MC325}

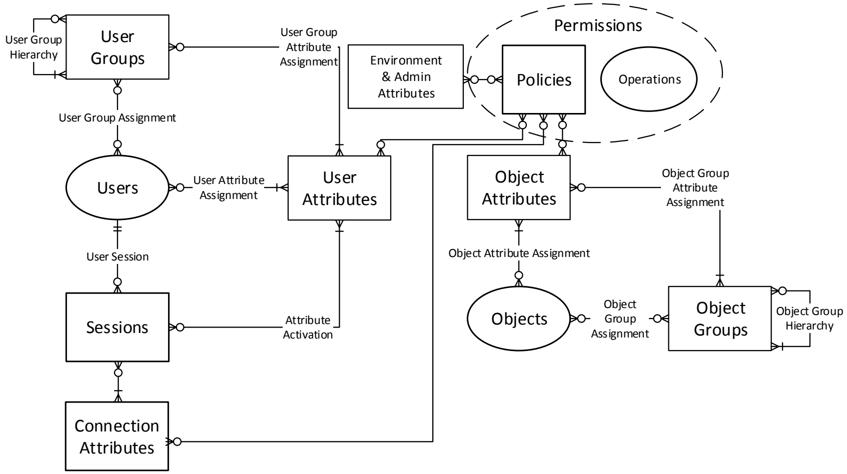
HGABAC Model



HGABAC Model



HGABAC Model

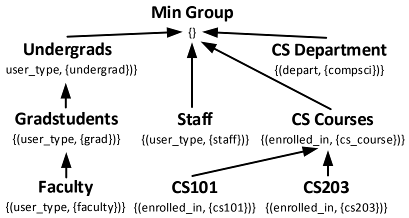


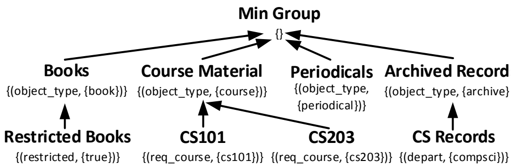
Use Cases

- Provide access control for a hypothetical university library.
- Access control is desired on four different kinds of resources; books, course material (textbooks, lecture notes, etc.), periodicals, and archived records.

Use Cases

- Provide access control for a hypothetical university library.
- Access control is desired on four different kinds of resources; books, course material (textbooks, lecture notes, etc.), periodicals, and archived records.
- Assumed User and Object Group Graphs:





Use Cases

Case A

Undergraduate students may check out any unrestricted book and any course materials for a course in which they are enrolled.

Use Cases

Case A

Undergraduate students may check out any unrestricted book and any course materials for a course in which they are enrolled.

```
“undergrad” IN user.user_type AND (
    (object.object_type = “book” AND NOT object.restricted) OR
    (object.object_type = “course” AND user.enrolled_in IN object.req_course)
) → check_out_book
```

Use Cases

Case B

Faculty may check out any book, periodical or course material as well as any archived record from their department.

Use Cases

Case B

Faculty may check out any book, periodical or course material as well as any archived record from their department.

```
“faculty” IN user.user_type AND (
    object.object_type IN {“book”, “periodical”, “course”} OR (
        object.object_type = “archive” AND object.depart IN user.depart
    )
) → check_out_book
```

Use Cases

Case C

Students enrolled in a computer science course may access periodicals from the university network.

Use Cases

Case C

Students enrolled in a computer science course may access periodicals from the university network.

Four connection attributes are required which represent the user's IP address; *"ip_octet_1"* represents the first digit of the user's IP address, *"ip_octet_2"*, the second and so on.

It is assumed that IP addresses matching the pattern "192.168.*.*" are internal to the university's network.

Use Cases

Case C

Students enrolled in a computer science course may access periodicals from the university network.

Four connection attributes are required which represent the user's IP address; *"ip_octet_1"* represents the first digit of the user's IP address, *"ip_octet_2"*, the second and so on.

It is assumed that IP addresses matching the pattern "192.168.*.*" are internal to the university's network.

"cs_course" IN user.enrolled_in AND
 connect.ip_octet_1 = 192 AND
  connect.ip_octet_2 = 168 AND
   object.object_type = "periodical"
→ **check_out_book**

Evaluation

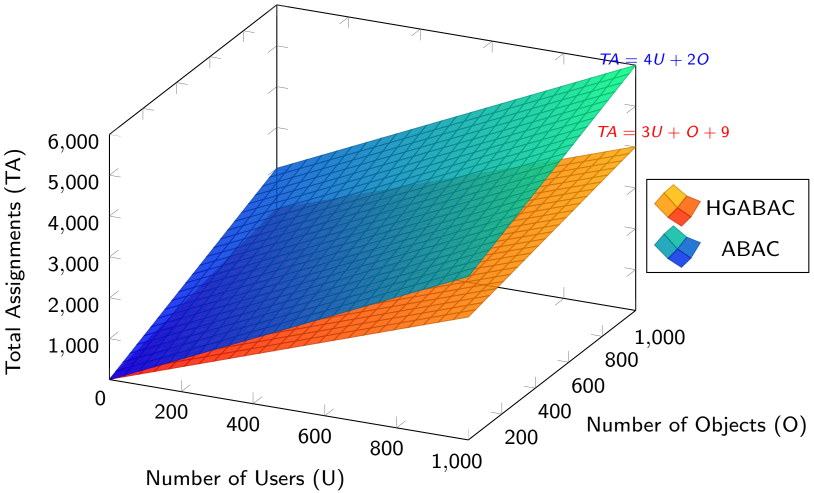
- Aimed to test whether the hierarchical user and object groups of the HGABAC model provide an advantage over non hierarchical ABAC models.
- Each model evaluated on basis of number of attribute and group assignments needed to full the requirements of each use case.
- Assumed non hierarchical models support environment and connection attributes for cases 4 and 5.
- Worst case (each user is enrolled in each course and each object is of an object type such that it will have the most attributes) is assumed.
- A constant number of courses and departments are assumed.

Results

Case A

Undergraduate students may check out any unrestricted book and any course materials for a course in which they are enrolled.

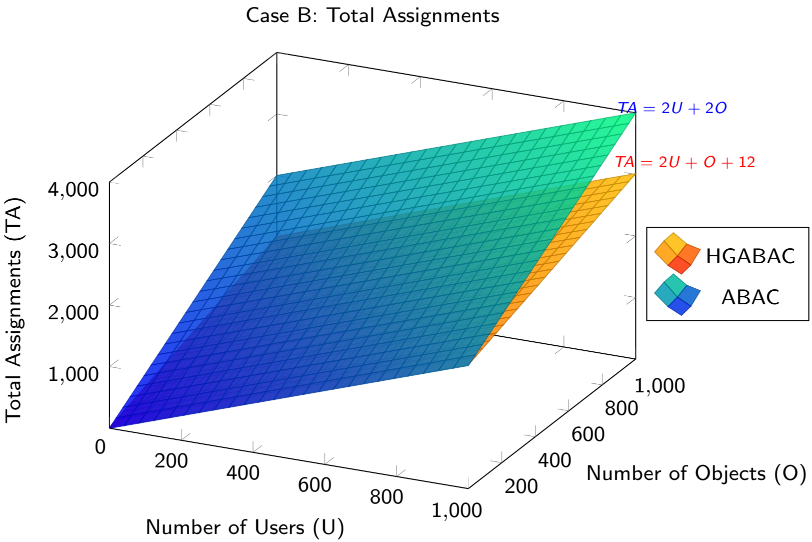
Case A: Total Assignments



Results

Case B

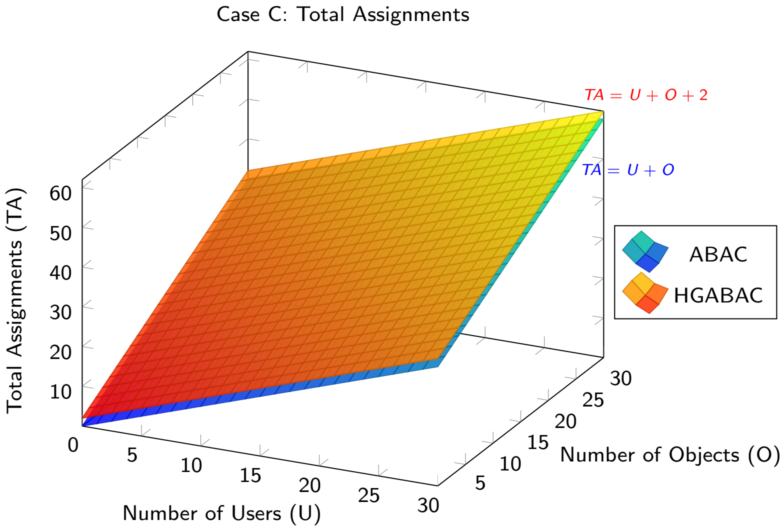
Faculty may check out any book, periodical or course material as well as any archived record from their department.



Results

Case C

Students enrolled in a computer science course may access periodicals from the university network.



Emulating Traditional Models
DAC Style Configuration

- Assigning each user an "id" attribute which contains a unique identifier.
- Assigning each object an attribute for each access mode (e.g. "read" and "write") which contains the set of user ids corresponding to users who have access to that object for the given access mode.
- Policy is simply: *(user.id IN object.read)* $\rightarrow$ **read**
  (user.id IN object.write) $\rightarrow$ **write**
- For administration add "owner" attribute to objects that contains a single user id corresponding to the owner of the object. Policy is:

  (user.id = object.owner) $\rightarrow$ **admin_operation**

Emulating Traditional Models
MAC Style Configuration

- HGABAC's user groups allow configurations that emulate MAC style lattice based access control.
- For MAC with liberal *-property, each user is assigned only to a single read group and a single write group. Each read group is assigned a single attribute named "read" with a value equal to its clearance level and each write group is assigned a single attribute named "write" with a value equal to its clearance level.
- Policy is simply: *(object.level IN user.read)*$\rightarrow$ **read**
  (object.level IN user.write) $\rightarrow$ **write**
- Users are limited to only activating attributes inherited from groups of a single security level in any given session.

Emulating Traditional Models
MAC Example

Security Lattice

$$\begin{tikzcd}
 & TS \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
S_1 \arrow[d, dash] & S_2 \arrow[dl, dash] \arrow[dr, dash] & S_3 \arrow[d, dash] \\
C_1 \arrow[dr, dash] & & C_2 \arrow[dl, dash] \\
 & U &
\end{tikzcd}$$

Liberal-* Group Graph

$$\begin{tikzcd}
 & TSR \arrow[dl] \arrow[d] \arrow[dr] & & & UW \arrow[dl] \arrow[dr] & \\
S_1R \arrow[d] & S_2R \arrow[dl] \arrow[dr] & S_3R \arrow[d] & C_1W \arrow[d] \arrow[dr] & & C_2W \arrow[dl] \arrow[d] \\
C_1R \arrow[dr] & & C_2R \arrow[dl] & S_1W \arrow[dr] & S_2W \arrow[d] & S_3W \arrow[dl] \\
 & UR \arrow[dr] & & & TSW \arrow[dll] & \\
 & & min\_group & & &
\end{tikzcd}$$

Strict-* Group Graph

$$\begin{tikzcd}
 & TSR \arrow[dl] \arrow[d] \arrow[dr] & & UW \arrow[ddddl] \\
S_1R \arrow[d] & S_2R \arrow[dl] \arrow[dr] & S_3R \arrow[d] & C_1W \arrow[dddl] \\
C_1R \arrow[dr] & & C_2R \arrow[dl] & C_2W \arrow[ddl] \\
 & UR \arrow[dr] & & S_1W \arrow[dl] \\
 & & min\_group & S_2W \arrow[l] \\
 & & & S_3W \arrow[ul] \\
 & & & TSW \arrow[uul]
\end{tikzcd}$$

Liberal *-property Attributes:

g	**direct(g)**	**effective(g)**
min_group	∅	∅
UR	"UR"	"UR"
C_1R	"C1R"	"UR", "C1R"
C_2R	"C2R"	"UR", "C2R"
S_1R	"S1R"	"UR", "C1R", "S1R"
S_2R	"S2R"	"UR", "C1R", "C2R", "S2R"
S_3R	"S3R"	"UR", "C2R", "S3R"
TSR	"TSR"	"UR", "C1R", "C2R", "S1R", "S2R", "S3R", "TSR"
TSW	"TSW"	"TSW"
S_1W	"S1W"	"TSW", "S1W"
S_2W	"S2W"	"TSW", "S2W"
S_3W	"S2W"	"TSW", "S3W"
C_1W	"C1W"	"TSW", "S1W", "S2W", "C1W"
C_2W	"C2W"	"TSW", "S2W", "S3W", "C2W"
UW	"UW"	"TSW", "S1W", "S2W", "S3W", "C1W", "C2W", "UW"

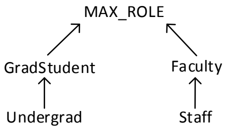
Emulating Traditional Models

RBAC Style Configuration

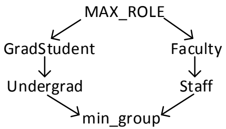
- HGABAC's user groups can also enforce hierarchical RBAC style access control by having each user group represent a role and its assigned attributes, represent permissions.
- Each group is assigned a single attribute named "perms" that contains the set of permissions that group grants.
- Objects are tagged with an attribute for each access mode that contains the set of permissions that grant that access mode on the object.
- Policy is simply: *(user.perms IN object.read)* → **read**
  (user.perms IN object.write) → **write**
- Emulating the separation of duty style constraints possible in NIST RBAC is left to future work.

Emulating Traditional Models
RBAC Example

Role Hierarchy



Group Graph



Role	**Direct Permissions**
Undergrad	P_1
Staff	P_2
GradStudent	P_3, P_4
Faculty	P_5, P_6
MAX_ROLE	$\emptyset$

g	**direct(g)**	**effective(g)**
min_group	$\emptyset$	$\emptyset$
Undergrad	P_1	P_1
Staff	P_2	P_2
GradStudent	P_3, P_4	P_1, P_3, P_4
Faculty	P_5, P_6	P_2, P_5, P_6
MAX_ROLE	$\emptyset$	P_1, P_2, P_3, P_4, P_5, P_6

Conclusions and Future Work
Conclusions:

- Introduced a new model of ABAC, entitled HGABAC, that supports boolean rule based ABAC, hierarchical user and object groups, as well as environment, connection and administrative attributes.
- Showed that adding user and object groups enables greater flexibility when modelling real world situations.
- Demonstrated that hierarchical user and object groups can simplify administration by reducing complexity in terms of the number of attribute and group assignments required.
- Showed that HGABAC is able to emulate the traditional models including hierarchical RBAC.

Conclusions and Future Work
Future Work:

- Extending HGABAC to support features required for real world use.
- Support for separation of duty.
- Delegation.
- Administrative model.
- Expanding the policy language or alternatively exploring using/extending XACML.
- Conditional user and object group membership.
- Reference implementation.