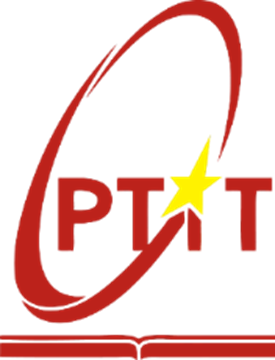
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

KHOA CÔNG NGHỆ THÔNG TIN 1

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***



**Đề Tài : XÂY DỰNG CÁC TÌNH HUỐNG CỦA HỆ THỐNG FILE VÀ XÂY DỰNG MÔ HÌNH AN TOÀN BELL-LA PADULA VÀ BIBA.**

**Giảng Viên : Phạm Hoàng Duy**

**Môn : An Toàn Hệ Điều Hành.**

**Nhóm : 01.**

**Thành Viên :**

1. **Nguyễn Doãn Nhân-B15DCAT128**
2. **Nguyễn Huy Hưng-B15DCAT088**
3. **Nguyễn Văn Tùng-B15DCAT191**
4. **Lưu Quang Huy –B15DCAT091**

Hà Nội, Ngày 09 Tháng 04 Năm 2018.

***Lời nói đầu***

Lời đầu tiên nhóm thực hiện đề tài xin cảm ơn thầy Phan Hoàng Duy ,giảng viên bộ môn An Toàn Hệ Điều Hành đã hướng dẫn tận tình nhóm thực hiện đề tài trong qua trình thực hiện bài tập này.

Ngày nay công nghệ thông tin đang ngày càng phát triển và ngày càng có nhiều phát minh mới .Việc sử dụng các công nghệ mới thì có rất nhiều cái mới mẻ và thuận lời hơn , song song việc đảm bảo tính an toàn bảo mật cho người dùng lại là một vẫn đề rất nan giải .Để hiểu được các cách thức hoạt động và nằm vững các kiến thức liên quan tới việc bảo mật thì chùng ta phải kiến thức chuyên sâu về nó .Và trong bài báo cáo thực hành này thì chung em đã tìm hiểu về cách xây dựng các tình huống của hệ thống file và xây dựng mô hình an toàn Bell-La Padula và Biba .Mong rằng một số kiến thức nhỏ này có thể giúp người dùng hình dung tốt hơn về các phương thức truy cập an toàn trong hệ thống file và nẵm rõ về các mô hình an toàn .

Bài báo cáo còn nhiều thiếu sót rất mong thầy góp ý để bài này và các bài về sau chúng em có thể làm hoàn thiện và tốt hơn. Một lần nữa, chúng em xin chân trọng cảm ơn.

**I Xây dựng các tình huống mô hình của hệ thống file**.

1. Các Định nghĩa về Alloy.

* *Alloy* **:**  Là một gôn ngữ rút gọn dùng để mô tả các thuộc tính có cấu trúc. Alloy hỗ trợ việc mô tả các cấu trúc cơ sở (dạng đồ họa hay các khai báo dạng văn bản), cũng như các ràng buộc phức tạp và các thao tác diễn tả sự thay đổi cấu trúc động (cả hai được biểu diễn bằng các công thức lô-gíc) . Nên ngôn ngữ về alloy thường được để biêu diễn và lập các mô hình .
* Sig: Mô tả một mô hình trong Alloy.
* Fact: Trong alloy fact có thể được sử dụng để định nghĩa các ràng buộc và các ràng buộc này sẽ luôn tự động được áp dụng lên các mô hình để biểu diễn . Fact là một mệnh đề đúng .
* Assert: Dùng để khai báo các ràng buộc giống như fact nhưng assert cho rằng một số thứ phải đúng tùy thuộc vào hành vi của mô hình do vậy assert có thể đúng và nó cũng có thể sai .
* Predicate : là một sự kiện trả về đúng hoặc sai, cũng có thể dùng để định nghĩa các ràng buộc .
* Toán tử " . " : Nó đại diện cho phép kết nối quan hệ nhưng trong cú pháp của Alloy nó có thể được dùng để truy cập tới một trường của sig cụ thể .
* Toán tử "+" : Nó thể hiện phép hợp (union) .
* *Hệ thống file* : Các đối tượng của hệ thống file có thể là file hoặc thư mục. Hệ thống này có các ràng buộc nhất định .

1. Định nghĩa các files và thư mục .

Định nghĩa tập **FSObject** đại diện cho tất cả các files và thư mục của hệ thống file (cú pháp giống định nghĩa một class trong lập trình hướng đối tượng)

**sig FSObject {  
 parent: lone Dir  
 }**

+ Trong tập FSObject định nghĩa một mối quan hệ **parent** - quan hệ giữa các FSObjects và thư mục khác;

+ **lone**: một FSObject có thể có 0 hoặc 1 thư mục cha.

- Định nghĩa tập **Dir (thư mục)** và **File (tệp tin)** tương tự - cả hai đều là tập con của FSObject và kế thừa các thuộc tính của Dir và File

**sig Dir extends FSObject {**

**contents: set FSObject**

**}**

**sig File extends FSObject { }**

1. Định nghĩa các ràng buộc cơ bản

* Thư mục phải là cha(mẹ )cấp trên của các đối tượng nằm bên trong nó

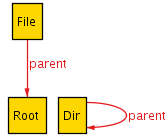
**fact {**

**all d: Dir, o: d.contents | o.parent = d**

**}**

* Có nghĩa là mọi thư mục **d** và các đối tượng **o** của FSObject nằm trong d thì cấp trên của o phải là d.

Nếu bỏ qua ràng buộc này sẽ xảy ra trường hợp có rất nhiều mối quan hệ "parent" nhưng lại không có một mối quan hệ "content" nào!



Mọi đối tượng của FSObject phải là files hoặc thư mục :

**fact {**

**File + Dir = FSObject**

**}**

Fact trên có thể được diễn giải bằng lời: tập tất cả file và thư mục là hệ thống file. Sẽ không có một đối tượng nào của hệ thống file không phải file hay thư mục cả.

* Hệ thống file có một thư mục Root :

**one sig Root extends Dir { } { no parent }**

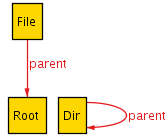
* Hệ thống file có kết nối :

**fact {**

**FSObject in Root.\*contents**

**}**

Nếu bỏ qua ràng buộc này sẽ xảy ra trường hợp:



Đường dẫn trong hệ thống file không tuần hoàn

**assert acyclic {**

**no d: Dir | d in d.^contents**

**}**

Sẽ không có thư mục nào chứa chính nó nghĩa là quan hệ giữa các thư mục không quay vòng. Như vậy quan hệ này được thể hiện như là tập đóng bắc cầu và khai báo thông qua toán tử ^ .

\*Kiểm chứng mô hình check acyclic for 5 :

Giả sử rằng mô hình A,B cùng bao gồm 2 phân loại ( classified và unclassified)

* chính sách của Meta A-1: không có thông tin đọc, có nghĩa là các object ở cấp unclassified là không thể đọc các object ở cấp classified.
* Chính sách của Meta A-2: không viết lại, ngăn ngửa các object ở cấp classified ghi các object ở mức unclassified.
* chính sách của Meta B-1 : Không có thông tin đọc, có nghĩa là các object ở cấp unclassified là không thể đọc các object ở cấp độ classified, ngoải các chủ đề tại cấp classified không thể đọc các object cấp topsecret.
* Chính sách của Meta B-2: chỉ cho phép viết cho các topsecret và unclassified. Các object ở cấp topsecret có thể viết trên các object ở cấp unclassified. Các object ở cấp topsecret không thể ghi các object ở cấp classified.
* 2 chính sách này có thể được viết bằng alloy bằng sử dụng đoạn mã sau

**no write down A1**: no ((Classified.~sec).~R & UNClassified.~level)

**no write lại A2**: no ((UNClassified.~sec).~W & Classified.~level)

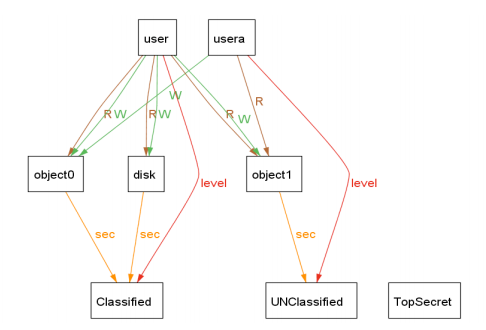
**no read up (B1)**:

* no ((Classified.~sec).~R & UNClassified.~level)
* no ((TopSecret.~sec).~R & Classified.~level)

**no write down (B2)**:

* no ((TopSecret.~sec).~W & Classified.~level)
* some ((UNClassified.~sec).~W & Classified.~level)

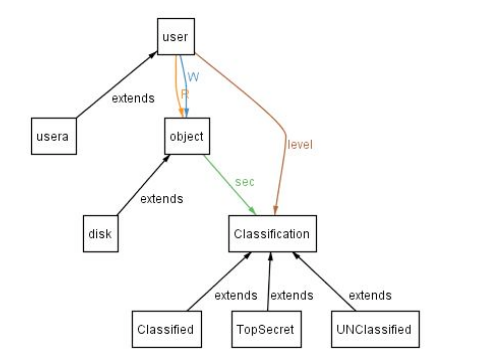
\*Kết hợp các mô hình A và B tạo ra một sự không nhất quán không ghi A-2 của mô hình A mâu thuẫn với chính sách ghi B-2 trong mô hình B



Tính không nhất quán của A và B

Ta thấy người sử dụng ở cấp độ classified đã viết truy cập vào Object1 đó là ở cấp unclassified, do đó phá vỡ chính sách không viết xuống.

\*Để tìm hiểu tính không nhất quán của hệ thống ta có mô hình:



Có 3 phân loại classified, topsecret, unclassified. Mỗi user có 1 mức nhất định gắn liền với 1 phân loại. Mỗi object có quan hệ bảo mật với phân loại. User có thể đọc và ghi các mối quan hệ với object. Hai phần mở rộng usera và disk là các thể hiện của user và object.

Giả sử có 1 tập S với thành phần userA ở mức Unclasssfied nhưng có quyền truy cập vào 1 thành phần mạng có nhãn disk đang ở mức classified. Chúng ta có thể kiểm tra nếu mô hình nếu thêm vào S có thể gây ra xung đột hợp lý bằng cách:

fact businessPolicyS {

usera.level = UNClassified

some usera.R

some usera.W

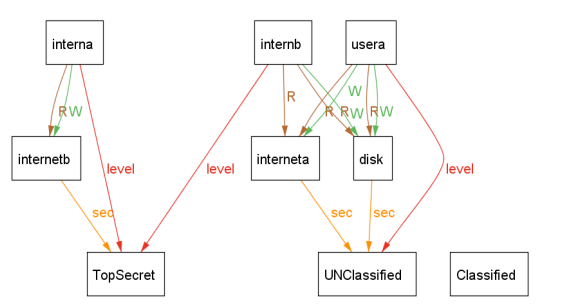
Classified in disk.sec

disk in usera.R

}

Mã này nói rằng UserA ở mức unclassfied có 1 số object có thể đọc được bởi userA và 1 số object có thể ghi được bởi userA, disk ở mức classified. Cuối cùng nó nói rằng userA có thể truy cập vào disk để đọc.

KQ: do sự không thống nhất không thể thêm vào.



\*Ngăn cản user truy cập đồng thời vào 1 số tài nguyên nhất định:

Ví dụ : 1 User đã đọc disk có thể bị từ chối quyền ghi vào internet

fact SOCPolicy {

internet in intern.W //ghi truy cập inter net

no disk & intern.R // đọc truy cập từ disk

}

Mã được liệt kê ở trên cho biết sự tách biệt của chính sách liên quan. Nó đọc rằng 1 thực tập không thể có quyền đọc truy cập vào disk trong khi có quyền truy cập ghi vào internet.

\*Một số code kiểm tra trên alloy:

sig FSObject { parent: lone Dir }

sig Dir extends FSObject { contents: set FSObject }

sig File extends FSObject { }

fact { all d: Dir, o: d.contents | o.parent = d }

fact { File + Dir = FSObject }

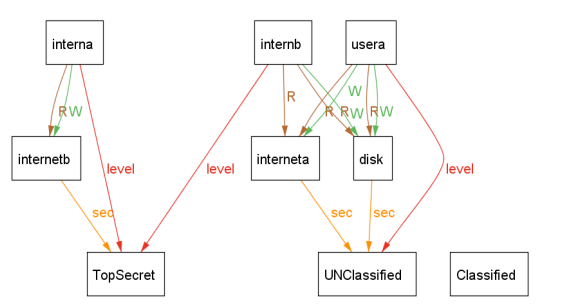
one sig Root extends Dir { }

fact { FSObject in Root.\*contents }

+assert acyclic { no d: Dir | d in d.^contents }

check acyclic for 5

Đường dẫn hệ thống file là không tuần hoàn:



\*Ngăn cản user truy cập đồng thời vào 1 số tài nguyên nhất định:

Vd: 1 user đã đọc disk có thể bị từ chối quyền ghi vào internet

fact SOCPolicy {

internet in intern.W //ghi truy cập inter net

no disk & intern.R // đọc truy cập từ disk

}

Mã được liệt kê ở trên cho biết sự tách biệt của chính sách liên quan. Nó đọc rằng 1 thực tập không thể có quyền đọc truy cập vào disk trong khi có quyền truy cập ghi vào internet.

\*Một số code kiểm tra trên alloy:

sig FSObject { parent: lone Dir }

sig Dir extends FSObject { contents: set FSObject }

sig File extends FSObject { }

fact { all d: Dir, o: d.contents | o.parent = d }

fact { File + Dir = FSObject }

one sig Root extends Dir { }

fact { FSObject in Root.\*contents }

assert acyclic { no d: Dir | d in d.^contents }

check acyclic for 5

Đường dẫn hệ thống file là không tuần hoàn:

d) Kiểm tra các mô hình hệ thống file tuân thủ và vi phạm ràng buộc bằng alloy.

Mô hình 3 :

// Một đối tượng Hệ thống file

abstract sig FSObject { }

// Đối tượng hệ thống file phải là thư mục hoặc file

sig File, Dir extends FSObject { }

// Một hệ thống file

sig FileSystem {

root: Dir,

live: set FSObject,

contents: Dir lone-> FSObject,

parent: FSObject ->lone Dir

}{

// Thư mục gốc không có cấp trên

no root.parent

// Các đối tượng tồn tại có thể truy cập tới từ thư mục gốc

live = root.\*contents

//mối quan hệ content chỉ được định nghĩa trên các đối tượng tồn tại

contents in live->live

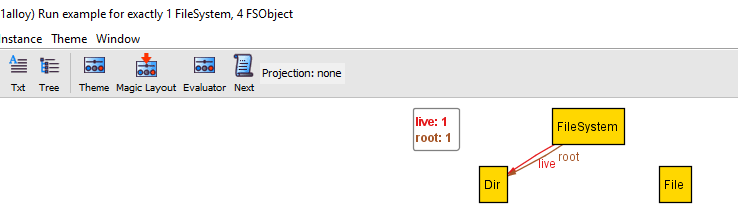
// Parent là ngược lại của các nội dung nó chứa (content)

parent = ~contents

}

pred example { }

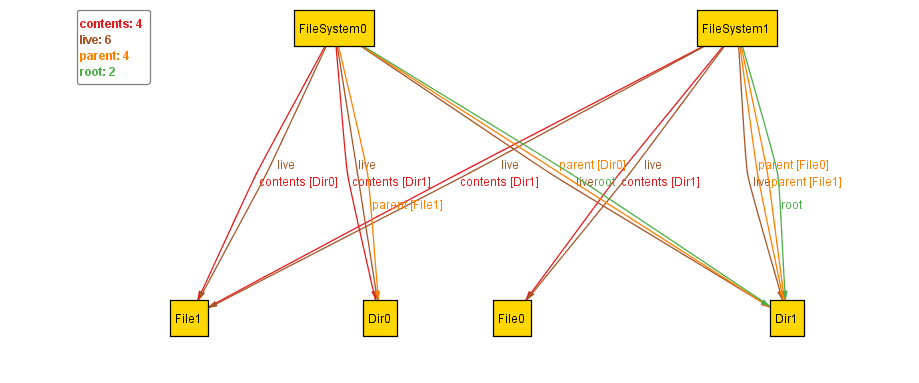
run example for exactly 1 FileSystem, 4 FSObject

Chạy mô hình qua phầm mềm alloy4.2 ta có giao diện có kết quả sau:

Và khi thay đổi một tình huống khác với:

run example for exactly 2 fileSystem , 4 FSObject .

kết quả :



Chúng ta có thể thấy răng hệ thống file đã tuân thủ các ràng buộc đặt ra.

Tài liệu tham khảo qua :

Mô Hình 4:

Trong mô hình này chúng ta sẽ bổ sung thêm các thao tác di chuyển, xóa các đối tượng hệ thống file.

// Định nghĩa các đối tượng hệ thống file

abstract sig FSObject { }

sig File, Dir extends FSObject { }

// Một hệ thống file

sig FileSystem {

live: set FSObject,

root: Dir & live,

parent: (live - root) ->one (Dir & live),

contents: Dir -> FSObject

}{

// Các đối tượng tồn tại có thể truy cập tới từ thư mục gốc

live in root.\*contents

// Parent là ngược lại của các nội dung nó chứa (content)

parent = ~contents

}

// Di chuyển đối tượng x tới thư mục d

pred move [fs, fs': FileSystem, x: FSObject, d: Dir] {

(x + d) in fs.live

fs'.parent = fs.parent - x->(x.(fs.parent)) + x->d

}

// Xóa file hoặc thư mục x

pred remove [fs, fs': FileSystem, x: FSObject] {

x in (fs.live - fs.root)

fs'.parent = fs.parent - x->(x.(fs.parent))}

// Xóa tất cả đối tượng x

pred removeAll [fs, fs': FileSystem, x: FSObject] {

x in (fs.live - fs.root)

let subtree = x.\*(fs.contents) |

fs'.parent = fs.parent - subtree->(subtree.(fs.parent))

}

run move for 2 FileSystem, 4 FSObject

run remove for 2 FileSystem, 4 FSObject

run removeAll for 2 FileSystem, 4 FSObject

// Thao tác di chuyển không thêm hoặc xóa bất kì đối tượng hệ thống file nào

moveOkay: check {

all fs, fs': FileSystem, x: FSObject, d:Dir |

move[fs, fs', x, d] => fs'.live = fs.live

} for 5

// thao tác xóa chỉ xóa chính xác file hoặc thư mục được chỉ định

removeOkay: check {

all fs, fs': FileSystem, x: FSObject |

remove[fs, fs', x] => fs'.live = fs.live - x

} for 5

// thao tác xóa toàn bộ xóa chính xác toàn bộ cây thư mục được chỉ định

removeAllOkay: check {

all fs, fs': FileSystem, d: Dir |

removeAll[fs, fs', d] => fs'.live = fs.live - d.\*(fs.contents)

} for 5

// thao tác xóa và xóa toàn bộ có cùng tác động lên files

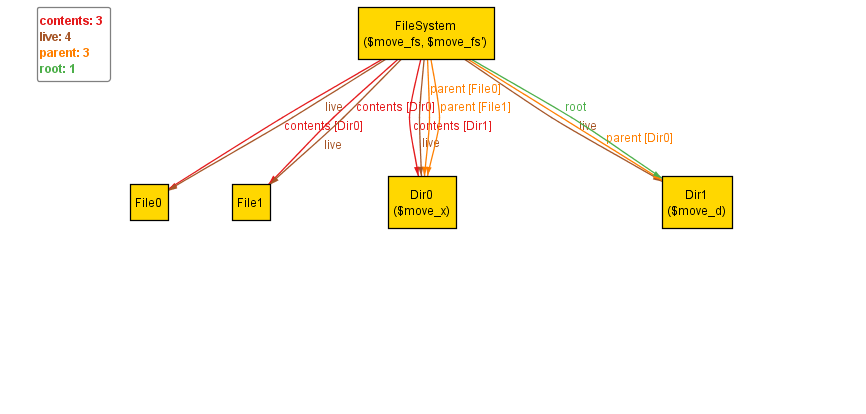
removeAllSame: check {

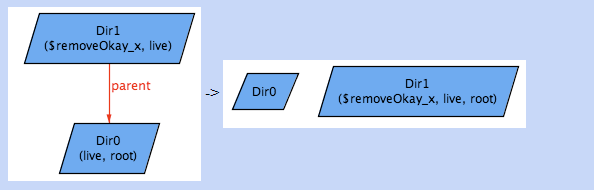
all fs, fs1, fs2: FileSystem, f: File |

remove[fs, fs1, f] && removeAll[fs, fs2, f] => fs1.live = fs2.live

} for 5

Chạy mô hình có kết quả:





Hình ảnh kết quả qua trang alloy tutorials để xét trường hợp vi phạm.

-Trước khi di chuyển, thư mục Dir0 là thư mục gốc, còn Dir1 là thư mục bị xóa. Sau khi xóa, không còn mối quan hệ parent tồn tại giữa hai thư mục này nữa nhưng Dir1 đã trở thành thư mục gốc

->Vi phạm ràng buộc hệ thống file.

- Điều này xảy ra do ràng buộc live in root.\*contents quy định thư mục gốc trong hệ thống file cũng là thư mục gốc của mọi đối tượng có quan hệ content với nó. Nên nếu mô hình sau khi thực hiện thao tác xóa có chứa một quan hệ content không rỗng, nó sẽ có chung mối quan hệ parent với mô hình trước khi xóa. Nhưng nếu thao tác xóa khiến mô hình sau khi thực thi thao tác này có một mối quan hệ content rỗng, thư mục gốc có thể trở thành bất kì thư mục nào đang tồn tại trong hệ thống file. Chúng ta có thể sửa lại ràng buộc của thao tác xóa này như sau:

pred remove [fs, fs': FileSystem, x: FSObject] {

x in (fs.live - fs.root)

fs'.root = fs.root

fs'.parent = fs.parent - x->(x.(fs.parent))

}

pred removeAll [fs, fs': FileSystem, x: FSObject] {

x in (fs.live - fs.root)

fs'.root = fs.root

let subtree = x.\*(fs.contents) |

fs'.parent = fs.parent - subtree->(subtree.(fs.parent)) }

=> Chúng Ta đã hoàn thiện một mô hình hệ thống file tuân thủ các ràng buộc.

**II Xây dựng mô hình an toàn bell-la Padula và Biba**

**\* Mô hình an toàn bell-la Padula:**

Năm1975,Bell và Lapadula hình thức hóa mô hình bảo mật đa cấp MAC(sau này được gọi là mô hình BLP). BLP là một mô hình máy trạng thái kiểm soát các yếu tố bí mật trong kiểm soát truy xuất. Các quyền hạn truy xuất được định nghĩa thông qua cả ma trận kiểm soát truy xuất và các mức bảo mật. Các quy tắc bảo mật ngăn ngừa thông tin rò rỉ từ mức bảo mật cao xuống mức thấp.

Để biểu diễn mô hình BLP, chúng ta sử dụng các ký hiệu sau đây:

• S là tập các chủ thể;

• O là tập các đối tượng;

• A={execute, read, append, write} là tập các quyền truy xuất;

• L là tập các mức bảo mật với phép quan hệ thứ tự bộ phận ;

• Một trạng thái được định nghĩa là một bộ ba (b, M, f) trong đó:

-b là một bộ ba (s,o,a),mô tả chủ thể s hiện tại đang thực hiện thao tác a trên đối tượng o.

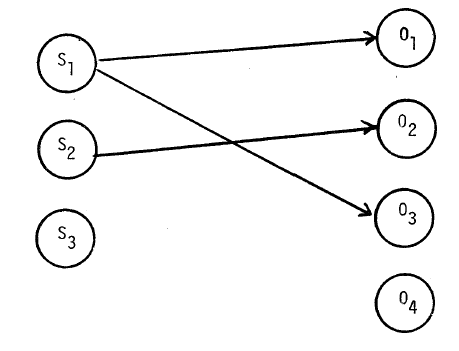
- M là một ma trận kiểm soát truy xuất M= (Mso)s ∈ S,o ∈ O.

- f =(fS,fC,fO), trong đó:

-fS:S → L cho biết mức bảo mật cao nhất mỗi chủ thể có thể có.

- fC:S → L cho biết mức bảo mật hiện tại của mỗi chủ thể, chúng ta luôn luôn có: fC(s) ≤ fS(s)hoặc viết là “ Fs chi phối fC”.

-fO:O → L cho biết mức bảo mật của mỗi đối tượng



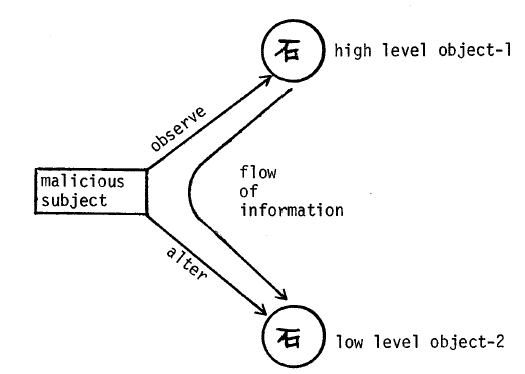
các chủ thể truy xuất các đối tượng

BLP định nghĩa bảo mật qua tính chất của các trạng thái.Tính chất thứ nhất là tính chất bảo mật đơn giản (simple security property), ký hiệu là ss-property.

ss-properties Một trạng thái (b, M, f) thỏa mãn tính chất ss-property, nếu mỗi phần tử (s, o, a) b, thao tác truy xuất a là read hoặc write, mức bảo mật của chủ thể s chi phối lớp đối tượng o, nghĩa là: fO(o) ≤ fS(s). Đặc trưng này đáp ứng được chính sách bảo mật truyền thống no read-up.

Tuy nhiên, tính chất ss-property không đảm bảo ngăn ngừa việc chủ thể ở mức bảo mật thấp đọc nội dung của một đối tượng có mức bảo mật cao Điều này phát sinh yêu cầu về tính chất khác, gọi là tính chất sao (star property), ký hiệu là \*-property

[1].



Luồng thông tin đòi hỏi \*-property

\*-properties Một trạng thái (b,M,f) thỏa mãn tính chất \*-property, nếu mỗi phần tử (s, o, a) b, thao tác truy xuất a là append hoặc write, mức bảo mật hiện tại của chủ thể s bị chi phối bởi lớp đối tượng o, nghĩa là: fC(s) ≤ fO(o). Đây là một chính sách no writedown.Hơn nữa,nếu có tồn tại một phần tử(s,o,a) b, a là append hoặcwrite,thì chúng ta phải có fO(o') ≤ fO(o)với mọi đối tượng o’ mà (s, o’, a’) b và a’ là read hoặc write.

Các chủ thể nắm giữ các quyền truy xuất có thể cấp lại các quyền này cho các chủ thể khác. Trong mô hình BLP, những chính sách như vậy có thể được đặc tả bằng một ma trận kiểm soát truy xuất và phải tuân thủ tính chất bảo mật tùy ý (discretionary security property), ký hiệu là ds-property.

ds-properties Một trạng thái (b, M, f) thỏa mãn tính chất ds-property, nếu mỗi phần tử (s, o, a) b chúng ta cóa ∈ Mso.

Định nghĩa (trạng thái bảo mật)Một trạng thái được gọi là bảo mật nếu cả ba tính chất bảo mật đều được thỏa mãn.

Bell và LaPadula đề xuất và chứng minh định lý cơ bản về bảo mật sau đây.

Định lý Nếu mọi phép biến đổi trạng thái trong một hệ thống là bảo mật và trạng thái ban đầu là bảo mật thì với đầu vào tùy ý, mọi trạng thái sinh ra là bảo mật.

Luồng thông tin

Để kiểm tra một hệ thống bảo mật(theo mô hình BLP),chúng ta cần kiểm tra trạng thái mới (b’, M’, f’) được sinh ra từ trạng thái (b, M, f) có bảo mật hay không. Ta xem xét một số khái niệm sau đây.

Ta nói rằng có một luồng thông tin hợp lý trực tiếp từ một đối tượng o tới đối tượng o' nếu có một chủ thể s sao cho hai điều kiện sau đây được thỏa.

1. (s, o, a) b và a là observe.

2. (s,o', a’) b và a’ là alter.

Một dãy o1,...,on được gọi là luồng thông tin hợp lý nếu có một luồng thông tin hợp lý trực tiếp từ oi tới oi+1 với1 ≤ i < n.

Chúng ta nói có một luồng thông tin hợp lý từ đối tượng o tới đối tượng o' nếu có một luồng thông tin hợp lý o1,...,on sao choo = o1,o' = on.

Định nghĩa(Luồng thông tin an toàn)Một luồng thông tin hợp o1,...,on được gọi là an toàn nếu fO(o1) ≤ fO(on).

**\*Mô hình toàn vẹn Biba**

Được xuất bản lần đầu tiên vào năm 1977, mô hình mạng dựa trên lưới này có những đặc tính xác định sau:

**Tính toàn vẹn đơn giản-thuộc tính này** cho biết rằng một chủ thể ở một mức độ toàn vẹn không được phép đọc một đối tượng có tính toàn vẹn thấp hơn.

**Star \* integrity property** - thuộc tính này cho biết rằng một đối tượng ở một mức độ toàn vẹn không được phép ghi vào một đối tượng có tính toàn vẹn cao hơn.

**Thuộc tính lời mời** - Tài sản này cấm một chủ thể ở một mức độ toàn vẹn từ việc viện đến một chủ thể ở cấp độ toàn vẹn cao hơn.

- Một cách dễ dàng để giúp bạn nhớ những quy tắc này là phải lưu ý rằng thuộc tính Star trong cả hai Biba và Bell-LaPadula đều phải viết. Chỉ cần nhớ, "Nó được viết trong các ngôi sao!"

- Biba chỉ giải quyết mục tiêu duy nhất của sự toàn vẹn-bảo vệ hệ thống để truy cập bởi người sử dụng trái phép. Tính khả dụng và bảo mật không được kiểm tra. Nó cũng giả định rằng các mối đe dọa nội bộ đang được bảo vệ bởi thực hành mã hóa tốt, và do đó tập trung vào các mối đe dọa bên ngoài.

- Để nhớ mục đích của mô hình Biba, chỉ cần ghi nhớ rằng "i" ở Biba là viết tắt của tính toàn vẹn.

- Hãy nhớ rằng mô hình Biba đề cập đến tính toàn vẹn và như vậy, viết cho một đối tượng ở mức cao hơn có thể gây nguy hiểm cho tính toàn vẹn của hệ thống.

Để diễn tả mô hình Biba,chúng ta sử dụng các quy ước sau đây:

• S là tập các chủ thể;

• O là tập các đối tượng;

• L là tập các mức toàn vẹn với phép sắp thứ tự bộ phận ;

• fS:S → Lcho biết mức toàn vẹn của mỗi chủ thể;

• fO:O → Lcho biết mức bảo mật của mỗi đối tượng.

Hàm fS và fO chỉ định mức toàn vẹn cho các chủ thể và các đối tượng. Những mức bảo mật này là cơ sở để mô tả các tính chất toàn vẹn để ngăn ngừa các thao tác kiểu như việc “làm sạch” các thực thể ở mức cao bằng cách “làm bẩn” các thực thể ở mức thấp.Chúng ta có thể phát biểu các tính chất trong hai trường hợp, mức toàn vẹn cố định trong đó mức toàn vẹn không thay đổi và mức toàn vẹn biến đổi ứng với mức toàn vẹn có thể thay đổi được.

Các mức toàn vẹn cố định

Hai tính chất toàn vẹn sau đây ngăn ngừa việc làm vấy bẩn các chủ thể và các đối tượng bằng các thông tin bẩn.

Simple integrity property Nếu chủ thể s có thể sửa đổi (biến đổi) đối tượng o, thì fS(s) ≥ fO(o). Đây là quy tắc không ghi-lên (no write-up).

Integrity \*-property Nếu chủ thể s có thể đọc (quan sát) đối tượng o, thì s chỉ có thể ghi lên đối tượng p nếu fO(p) ≤ fO(o).

Luồng thông tin

Chúng ta nói có một luồng thông tin hợp lý trực tiếp từ đốitượng o tới đốitượng o' nếu có một chủ thể s sao cho s có thể quan sát ovà biến đổio'.

Một dãy o1,...,on được gọi là một luồng thông tin hợp lý nếu có một luồng thông tin hợp lý trực tiếp từ oi tớioi+1 với mỗi 1 ≤ i < n.

Chúng ta nói có một luồng thông tin hợp lý từ đối tượng o tới đối tượng o' nếu có một luồng thông tin hợp lý o1,...,on sao choo = o1,o' = on.

Định nghĩa Một luồng thông tin o1,...,on được gọi là an toàn nếu fO(o1) ≥ fO(on).

Các mức toàn vẹn biến đổi

Hai tính chất sau đây giúp tự động điều chỉnh mức toàn vẹn của một thực thể nếu nó thực hiện tương tác với thông tin ở mức thấp hơn.

Subject low watermark property chủ thể s có thể đọc (quan sát) một đối tượng o tại bất kỳ mức toàn vẹn nào. Mức toàn vẹn mới của chủ thể là inf(fS(s),fO(o)), trong đó fS(s) vàfO(o)là các mức toàn vẹn trước khi thao tác được thực hiện.

Object low watermark property chủ thể s có thể sửa (biến đổi) đối tượng o tại bất kỳ mức toàn vẹn nào. Mức toàn vẹn mới của đối tượng là inf(fS(s),fO(o)), trong đó fS(s) và fO(o)là các mức toàn vẹn trước khi thao tác được thực hiện.

Luồng thông tin

Chúng ta nói rằng có một luồng thông tin trực tiếp từ đối tượng o tới đối tượng o', ký hiệu bởi (s,o,o')nếu có một chủ thể trước hết quan sát ovà sau đó biến đổi o'

Một dãy o1,s1,o2,...,on − 1,sn − 1,on được gọi là một luồng thông tin nếu (oi,si,oi+1) là luồng thông trực tiếp với mọi 1 ≤ i < n.

Chúng ta nói có một luồng thông tin từ một đối tượng o tới đối tượng o' nếu có tồn tại một luồng thông tin o1,s1,o2,...,on − 1,sn − 1,on sao choo = o1,o' = on.

Cho a ∈ read,write. Theo các quy tắc toàn vẹn biến đổi, khi một chủ thể thực hiện một thao tác truy xuất a, Fs hoặc fO có thể bị thay đổi.Sự thay đổ inày được biểu diễn bởi ánh xạ α:α(s,o,read,fS,fO) = f'S

trong đó f'S được định nghĩa như sau: với mỗi chủ thể r:

f'S(r) = fS(r)nếur ≠ s

f'S(r) = inf (fS(s),fO(o)) nếu r = s

Tương tự chúng ta có thể định nghĩa:

α(s,o,write,fS,fO) = f'O

trong đóf'O được định nghĩa như sau: với mỗi đối tượng o':

f'O(o') = fO(o')nếuo' ≠ o

f'O(o') = inf(fS(s),fO(o))nếuo' = o

Gọi s là chủ thể đã thực hiện đọc từ ovà ghi vàoo'. Định nghĩa:

Ω (s,o,o',fS,fO) = (f'S,f'O)

trong đóf'S = α(s,o,read,fS,fO)vàf'O = α(s,o',read,f'S,fO).

Định nghĩa Một luồng thông tin o1,s1,o2,...,on − 1,sn − 1,on được gọi là an toàn nếu fO,n(o1) ≥ fO,n(on). Trong đó:

(fS,1,fO,1) = Ω (s,o1,o2,fS,fO) (fS,i+1,fO,i+1) = Ω (si,oi,oi+1,fS,i,fO,i)

Thực tế, một chủ thể có thể triệu gọi thực thể khác, ví dụ, một tiến trình gọi một tiến trình khác trong lúc đang thực thi.Mô hình Biba có thể được mở rộng để xử lý thao tác dạng này [1]. Chúng ta xét hai tính chất sau.

Invoke property chủ thể s1 chỉ có thể triệu gọi chủ thể s2 nếu fS(s2) ≤ fS(s1).

Ring property một chủ thể s1 có thể đọc các đối tượng ở tất cả các mức toàn vẹn. Nó chỉ có thể sửa đổi đối tượng o với fO(o) ≤ fS(s) và nó chỉ có thể triệu gọi chủ thể s2 nếu fS(s1) ≤ fS(s2).

**III) Tài Liệu tham khảo :**

1. <http://alloytools.org/tutorials>.
2. Bài giảng An Toàn Hệ Điều Hành, Học viện Công nghệ Bưu Chính Viễn Thông