

Access Control Matrix

Yrd. Doç. Dr. Özgü Can

Giriş

- Güvenlik sistemleri, hangi şartlar altında sistemin güvenli olduğunu belirtmektedir.
- Erişim Denetim matrisi (Access Control Matrix)
 - Matris yapısını kullanarak izin verilen erişimleri tanımlar.
 - İşletim Sistemleri
 - Veritabanı

Korunum Durumu (Protection State)

- **Sistemin durumu** (the state of a system), sistemin bütün bileşenlerinin (*bellek yeri, ikincil bellekler, yazmaçlar vs..*) mevcut değerlerini belirtir.
- Sistem durumunun güvenlik ile ilgilenen kısmı **sistemin korunum durumunu (protection state of the system)** belirtir.
 - Erişim denetim matrisi mevcut korunum durumunu tanımlar.

Tanım

- *Olası korunum durumları kümesi: P*
- *Sistemin yetkilendirilmiş (authorized) durumda olduğunu belirten durumlar: Q*
 - *Q , P 'nin alt kümesi: $Q \subset P$*
- Mevcut durum $P \setminus Q$ ise sistem güvenilir değildir.
 - $P \setminus Q$ durumu: P 'nin Q içerisinde olmayan bütün elemanlarını ifade eder.

Tanım

- Durumu belirtirken amaç:
 - Q'nun içerisindeki durumları tanımlamak

Güvenlik politikasının bir fonksiyonudur.
- Güvenliği uygularken amaç:
 - **Sistem durumunun her zaman Q'nun bir elemanı olmasını garantilemek**

Sistemin P Q'nun durumlarından birine girmesini önlemek



Güvenlik düzeneğinin bir fonksiyonudur.

Korunum Durumu (Protection State)

- Erişim Denetim Matrisi modeli **korunum durumunu tanımlamak** için kullanılan bir modeldir.
- Her bir öznenin (subject) erişim hakkını diğer varlıklara göre tanımlar.

Korunum Durumu (Protection State)

- Sistem değıştikçe korunum durumu da değışir.
- Bir komut, sistemin durumunu değıştirirse, bir **durum değışimi** (state transition) meydana gelir.
- Çoğunlukla, izin verilen durumlar kümesindeki kısıtlar (constraints) bu değışimleri tümevarımsal olarak kullanır.
 1. İzin verilen durumlar kümesi tanımlanır.
 2. Bu kümedeki elemanlar üzerinde izin verilen işlemler kümesi tanımlanır.

Korunum Durumu (Protection State)

- Pratikte; sistem üzerindeki herhangi bir işlem çoklu durum değişimlerine neden olur.
 - Verinin
 - Okunması
 - Yüklenmesi
 - Üzerinde değişiklik yapılması
 - Yürütülmesi



Değişime
neden olur.

Korunum Durumu (Protection State)

- Bizim ilgilendiğimiz durum değişimleri:
 - Sistemin korunum durumunu etkileyen durum değişimleridir.



Bu nedenle, öznenin yetkilendirildiği eylemlerde değişiklik yapan değişimler konu ile ilgilidir.

Korunum Durumu (Protection State)

- İlgilendiğimiz durum değişimleri:
 - Sistemin korunum durumunu etkileyen durum değişimleridir.
- ÖR:
 - Değişkenin değerini 0 olarak değiştiren program güvenlik durumunu değiştirmez.
 - Değeri değiştirilen değişken, bir sürecin ayrıcalıklarını (privilege) etkiliyorsa, bu program korunum durumunu değiştiriyordur ve değişim kümesi içinde değerlendirilir.

Erişim Denetim Matris Modeli

- Kullanıcının dosyalar üzerindeki haklarını matris içerisinde tanımlar.
 - Korunum durumunun formal bir güvenlik modelidir.
- İlk model → Butler W. Lampson (1971)
- Kullanılan versiyon → Graham ve Denning

Erişim Denetim Matris Modeli

- Nesneler kümesi (set of objects): **O**
 - Korunan varlıklar \rightarrow O
- Özneler kümesi (set of subjects): **S**
 - Süreçler, kullanıcılar, vs..
- Haklar kümesi (set of rights): **R**
- Varlıklar arasındaki ilişkiler **A** **matrisinde** tutulmaktadır.
 - Özneler \rightarrow satır
 - Nesneler \rightarrow sütun

Erişim Denetim Matris Modeli

- Her bir girdi (entry) $a[s, o]$ olarak ifade edilir:
 - $s \in S$
 - $o \in O$
 - $a[s, o] \subseteq R$
- s öznesi o nesnesi üzerinde $a[s, o]$ haklar kümesine sahiptir.
- (S, O, A) üçlüsü \rightarrow Sistemin korunum durumları kümesini ifade eder.

Erişim Denetim Matrisi Modeli

		Nesneler (objects)					
		o_1	...	o_m	s_1	...	s_n
Özneler (subjects)	s_1						
	s_2						
	...						
	s_n						



- Özneler $S = \{ s_1, \dots, s_n \}$
- Nesneler $O = \{ o_1, \dots, o_m \}$
- Haklar $R = \{ r_1, \dots, r_k \}$
- Girdiler $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \dots, r_y \} \rightarrow s_i$ öznesi, o_j nesnesi üzerinde r_x, \dots, r_y haklarına sahiptir.

Erişim Denetim Matris Modeli

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

- Sistemde:
 - 2 süreç
 - 2 dosya
- Haklar kümesi = {read, write, execute, append, own}
- Her bir süreç kendisi ile aynı numaradaki dosyaya sahiptir.
- Süreçler hem satırda hem sütunda yer almaktadır.
 - Böylece; bir süreç hem işlemin hedefi hem de işlemci olabilmektedir.

Erişim Denetim Matris Modeli

- Hakların anlamlarının yorumlanması sistemden sisteme değişmektedir.
- Dosya okuma-yazma-ekleme yapma 
- Süreçten okuma yapma 
 - Modele göre değişir:
 - okunan süreçten mesajları kabul ediyor ya da
 - okunan sürecin durumuna bakıyor olabilir.
- Hakların yorumlanması nesneye göre değişebilir.

Erişim Denetim Matris Modeli

ÖZET:

- Erişim denetim matrisi modeli korunum durumunun soyut bir modelidir.
- Erişim denetim matrisinde yer alan bir ayrıntıdan bahsedilirken sistemin gerçekleştirimini dikkate almak gerekir.

Erişim Denetim Matris Modeli

- `own` → Ayrıcalıklı bir haktır.
- Bir çok sistemde nesneyi yaratan öznenin özel ayrıcalıkları vardır:
 - Kendisi ve diğer kullanıcılar için hak yaratma ya da silme
 - **Process 1** herhangi bir **x** öznesi için **A[x, file1]** izninde değişiklikler yapabilir.

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

Erişim Denetim Matris Modeli

- UNIX sistemi
 - Bir süreç bir dosyaya eriştiğinde haklar:
 - Read
 - Write
 - Execute

Erişim Denetim Matris Modeli

- Read
 - Dizinin içeriğini listeleyebilir.
- Write
 - Dizin ve onun alt dizinlerinde dosya yaratabilir, dosyaları değiştirebilir ya da silebilir.
- Execute
 - Dizin ve onun alt dizinlerindeki dosyalara erişebilir.

UNIX Sistemi

Erişim Denetim Matris Modeli

- Süreçler arasında:
 - Read
 - Sinyal alımı
 - Write
 - Sinyal gönderimi
 - Execute
 - Sürecin alt süreç olarak yürütülmesi


UNIX Erişim İzinleri

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwX

Value	Permission	Directory Listing
0	No read, no write, no execute	---
1	No read, no write, execute	--x
2	No read, write, no execute	-w-
3	No read, write, execute	-wx
4	Read, no write, no execute	r--
5	Read, no write, execute	r-x
6	Read, write, no execute	rw-
7	Read, write, execute	rwX

UNIX Erişim İzinleri

```
% ls -l /etc/passwd
-rw-r--r--  1 root      sys      1306 Jan 31 17:07 /etc/passwd
uuugggooo    owner    group    size    date    mod    name
```



chown - change file ownership
chgrp - change group status of a file
chmod - change access permissions for one or more files

Erişim Denetim Matris Modeli

- Superuser
 - Dosyaların yaratıcısından bağımsız olarak, bütün dosyalara erişim hakkına sahiptir.
 - Sistemdeki bütün nesnelerin sahibidir.
 - Yine de erişim hakları kısıtlandırılabilir.
 - Dosyalar üzerinde değişiklik yapan sistem çağrılarını ve komutlarını kullanarak dizinlerde değişiklik yapamaz.
 - Dosyaları yaratmak, ismini değiştirmek ya da silmek için özel sistem çağrıları ve komutları kullanmalıdır.

Erişim Denetim Matris Modeli

- Nesneler:
 - Dosyalar
 - Cihazlar
 - Süreçler
 - Mesajlar

Erişim Denetim Matris Modeli

- LAN üzerinde bir erişim denetim matrisi:

host names	<i>telegraph</i>	<i>nob</i>	<i>toadflax</i>
<i>telegraph</i>	own	ftp	ftp
<i>nob</i>		ftp, nfs, mail, own	ftp, nfs, mail
<i>toadflax</i>		ftp, mail	ftp, nfs, mail, own

- Network protokollerine verilen haklar:
 - own → server
 - ftp → sisteme FTP ile erişim
 - nfs → sisteme NFS ile erişim
 - mail → mail işlemleri için SMTP kullanımı

Erişim Denetim Matris Modeli

- Programlama dillerinin erişimini modellemek için de erişim denetim matrisi kullanılır.
 - Değişkenler (Variables) → Nesne
 - Yordamlar (Procedures) → Özne

Erişim Denetim Matrisi Modeli

- **ÖR:** `counter`'ı arttıran `inc_ctr` yordamı ve azaltan `dec_ctr` yordamı
- Haklar = $\{+, -, call\}$
 - $+$ → arttırma
 - $-$ → azaltma
 - `call` → yordamı çağırma

	<i>counter</i>	<i>inc_ctr</i>	<i>dec_ctr</i>	<i>manager</i>
<i>inc_ctr</i>	+			
<i>dec_ctr</i>	-			
<i>manager</i>		call	call	<div><div>call</div><div>recursive</div></div>

Erişim Denetim Matris Modeli

- Erişim denetim matrisinde girdiler her zaman haklar olmak zorunda değildir.
- Girdiler, başka verileri temel alan belirli durumların haklarını belirleyen fonksiyonlar olabilir.
 - Önceki erişimlerin geçmişi (history)
 - Gün-zaman bilgisi
 - Nesne üzerinde başka öznelerin hakları
 - ÖR fonksiyon: Bernstein durumu

Erişim Denetim Matris Modeli

- **Bernstein durumu**

- Verinin tutarlı olmasını garantiler.
- Veriyi okumak isteyenler aynı anda veriyi okuyabilirler, ancak dosya üzerinde yazma işlemi gerçekleştirilmek isteniyorsa, yazma işlemi bitene kadar kimse dosya üzerinde yazma ya da dosyayı okuma işlemi gerçekleştiremez.
 - Bir süreç dosyaya yazıyorsa, diğer süreçler dosyaya erişemez.
 - Yazma işlemi bittikten sonra diğer süreçler dosyaya erişebilir.

Korunum Durum Değişimleri (Protection State Transitions)

- Süreçler işlemleri yürüttükçe korunum sisteminin durumu değişir.
- $| -$ değişimi ifade eder
 - $X_i | -_{\tau} X_{i+1}$: τ komutu sistemin durumunu X_i 'den X_{i+1} 'e değiştirir.
 - $X_i | -^* X_{i+1}$: Sistemi X_i 'den X_{i+1} 'e değiştiren bir komutlar dizisi
- Değişimler komutlar ile ifade edilir.

Korunma Durum Değişimleri (Protection State Transitions)

- Komutlar erişim denetim matrisini günceller.
 - Matrisin hangi girdisinin değişeceği ve nasıl değişeceğini belirtir.
- Her bir komut için, başlangıçtaki X_i durumunu X_{i+1} sonuç durumuna değiştiren işlemler dizisi vardır.

Korunum Durum Değişimleri (Protection State Transitions)

- **Primitive Commands**

- Erişim denetim matrisinde değişiklik yapar.

Her bir komuttan önce korunum durumu

(S, O, A)

Her bir komuttan sonra korunum durumu

(S', O', A')

Korunum Durum Değişimleri (Protection State Transitions)

- **create subject s**
 - Yeni bir s öznesi yaratır.
 - Öznenin daha önce yaratılmamış olması gerekir.
 - Her hangi bir hak yaratmaz.
 - Sadece matrisi değiştirir.

Korunum Durum Değişimleri (Protection State Transitions)

- `create object` `o`
 - Yeni bir `o` nesnesi yaratır.
 - Nesnenin daha önce yaratılmamış olması gerekir.
 - Her hangi bir hak yaratmaz.
 - Sadece matrisi değiştirir.

Korunum Durum Değişimleri (Protection State Transitions)

- **enter** r **into** $A[s, o]$
 - s öznesi için o nesnesi üzerinde r hakkını ekler = $A[s, o]$ hücreesine r hakkını ekler
 - Hak daha önce yaratılmış ise modelin temsiline göre ya bir kopyası oluşturulur ya da herhangi bir işlem gerçekleştirilmez.

Korunum Durum Değişimleri (Protection State Transitions)

- **delete** r **from** $A[s, o]$
 - s öznesi için o nesnesi üzerinde r hakkını siler = $A[s, o]$ hücrelerine r hakkını siler
 - Böyle bir hak mevcut değilse herhangi bir işlem gerçekleştirilmez.

Korunum Durum Değişimleri (Protection State Transitions)

- **destroy subject s**
 - s öznesini siler.
 - Matriste s öznesinin satır ve sütunu silinir.

Korunum Durum Değişimleri (Protection State Transitions)

- `destroy object o`
 - `o` nesnesini siler.
 - Matriste `o` nesnesinin sütunu silinir.

Korunum Durum Değişimleri (Protection State Transitions)

- Bu komutlar birlikte kullanılabilir ve çoklu işlemler gerçekleştirilebilir.
- [UNIX] f dosyasını sahibine $read(r)$ ve $write(w)$ hakları ile yaratan p süreci ile ilgili erişim denetim matrisinde değişiklik yapan komut:

```
command create•file( $p$ ,  $f$ )  
  create object  $f$ ;  
  enter  $own$  into  $A[p, f]$ ;  
  enter  $r$  into  $A[p, f]$ ;  
  enter  $w$  into  $A[p, f]$ ;  
end
```


Korunum Durum Değişimleri (Protection State Transitions)

- [UNIX] p süreci yeni bir q süreci yaratmak isterse erişim denetim matrisinde değişiklik yapan komut:

```
command spawn • process( $p$ ,  $q$ )  
  create subject  $q$ ;  
  enter  $own$  into  $A[p, q]$ ;  
  enter  $r$  into  $A[p, q]$ ;  
  enter  $w$  into  $A[p, q]$ ;  
  enter  $r$  into  $A[q, p]$ ;  
  enter  $w$  into  $A[q, p]$ ;  
end
```

- İki süreç birbirine sinyal gönderebilmektedir.

Korunum Durum Değişimleri (Protection State Transitions)

- **Mono-Operational**

- Komut sadece tek bir *primitive* i çağırır.

- f dosyasının sahiplerine p öznesini ekleyen komut:

```
command make•owner( $p$ ,  $f$ )  
  enter own into  $A[p, f]$ ;  
end
```

- Herhangi bir kullanıcı hakkını silmez.

- f 'nin bir den fazla sahibi olabilir.

Korunum Durum Değişimleri (Protection State Transitions)

- Bazı primitive'lerin yürütülmesinde belirli önkoşulların sağlanması gerekmektedir.
- ÖR: p sürecinin q sürecine eğer p f dosyasının sahibi ise f dosyasını okuma hakkı vermesi:

```
command grant • read • file • 1( $p$ ,  $f$ ,  $q$ )  
  if own in  $A[p, f]$   
  then  
    enter  $r$  into  $A[q, f]$ ;  
  end
```

Korunum Durum Değişimleri (Protection State Transitions)

- Koşullar `and` kullanılarak birlikte kullanılabilir.
 - `or` **ve** not kullanılmaz

Korunum Durum Değişimleri (Protection State Transitions)

- Eğer p öznesi f dosyasının sahibi ise ve q öznesi üzerinde c hakkına sahipse, p öznesinin q öznesine f dosyası üzerinde r ve w haklarını vermesi:

```
command grant • read • file • 2( $p, f, q$ )  
  if own in  $A[p, f]$  and  $c$  in  $A[p, q]$   
  then  
    enter  $r$  into  $A[q, f]$ ;  
    enter  $w$  into  $A[q, f]$ ;  
  end
```

Korunum Durum Değişimleri (Protection State Transitions)

- Tek bir koşullu komutlar → **Monoconditional**

```
command grant•read•file•1(p, f, q)  
  if own in A[p, f]  
  then  
    enter r into A[q, f];  
  end
```

- İki koşullu komutlar → **Biconditional**

```
command grant•read•file•2(p, f, q)  
  if own in A[p, f] and c in A[p, q]  
  then  
    enter r into A[q, f];  
    enter w into A[q, f];  
  end
```

Özet

- Erişim denetim matrisi anlatılabilir bir güvenlik politikasını açıklar.
- Alan kısıtından dolayı pratikte direkt olarak kullanılmaz.
 - Bir çok sistemde binlerce nesne ve özne olabilir.
- Basitliğinden dolayı güvenlik problemlerinin teorik analizleri için uygundur.