



Criptografía y Seguridad

Seguridad en
aplicaciones:
Control de acceso

Matriz de control de acceso

- Modelo más simple
 - Una fila por usuario
 - Una columna por objeto
 - Acciones permitidas en la intersección

	Obj1	Obj2	Obj3	Obj n
Usuario1	r	r	r x				r w x
Usuario2		o x					
...							
...							
Usuario n	o x						

Características

- Permite implementar cualquier política
- Rápido crecimiento
 - Ejemplo: 100.000 archivos, 500 usuarios
 - = 50.000.000 de entradas
- Desperdicio de espacio
- Facilidad para determinar accesos
- Complejidad para soportar altas y bajas
 - Cambia la estructura de la matriz
- Administración compleja (elemento a elemento)

Listas de Control de Acceso (ACL)

- Representan columnas de una matriz de acceso

	Obj1	Obj2	Obj3	Obj n
Usuario1	r	r	r x				r w x
Usuario2		o x					
...							
...							
Usuario n	o x						

ACLs:

Obj1: { (usuario1, r), (usuario n, ox) }

Obj2: { (usuario1, r), (usuario2, ox) }

Obj3: { (usuario1, rx) }

...

Definición

- Sean
 - S conjunto de sujetos
 - O conjunto de objetos
 - R conjunto de acciones (derechos)
- $ACL(o) = \{ (s_i, r_i) / s \in S, r \in R \}$
 - (s_i, r_i) implica que s_i accede a o con cualquier derecho de i_r
- Si un sujeto no tiene entrada en $ACL(o)$ no tiene ningún derecho



Principio de denegar por defecto

Modificación de ACLs

- Derecho de pertenencia (own)
 - Puede ser asignado a quien crea el objeto
 - Puede ser asignado según el tipo de objeto
- Transferencia de permisos
 - Traspaso: cambiar el sujeto de una entrada de ACL
 - (Usuario 1, t rwx) \rightarrow (Usuario 2, t rwx)
 - Delegación
 - (U1, t rx) \rightarrow (U1, t rx) , (U2, rx)
 - (U1, t rx) \rightarrow (U1, t rx) , (U2, r)

Usuarios privilegiados

- Usuarios a los cuales no aplican los ACLs
 - Ejemplo: root en Linux → tiene todos los accesos
- Usuarios con ACLs especiales
 - Ejemplo: administrator en Windows 200x
 - Tiene el derecho (take ownership) sobre todos los objetos

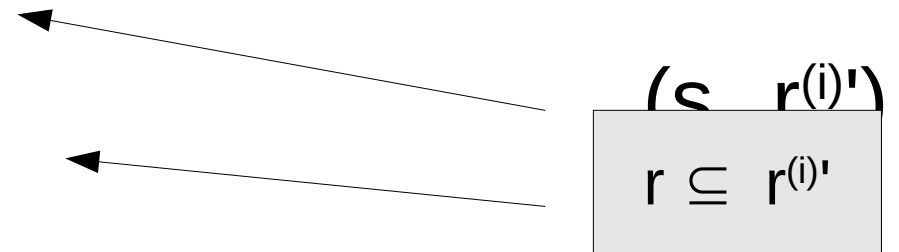
Manejo de volumen: Grupos

- Aunque menos que la matriz, los ACLs pueden crecer mucho
- En la práctica, se utilizan grupos (roles)
 - No pertenecen al modelo clásico de ACLs

- $g = \{ s_1, s_2, s_i / s \in S \}$

- $(g, r) \subset ACL(o) \Leftrightarrow (s_1, r') \subset ACL(o) \wedge \dots \wedge$

$\subset ACL(o)$



- Reducen tamaño de ACLs
- Agregan una nueva dimensión de complejidad

Grupos y usuarios - Conflictos

- Más de un AC pueden aplicar al mismo tiempo
- Escenario

- $ACL(o) = \{ (Pablo, rw), (Profesores, r) \}$
- $Pablo \in Profesores$

- Grant-All

- Requiere que todos los ACs otorguen el derecho

En el ejemplo, Pablo puede leer, pero no escribir o

Requiere además, definir un orden de evaluación de ACLs

- First-Rule

- El primer ACL encontrado es utilizado

En el ejemplo, Pablo puede leer y escribir

Derechos por defecto

- Permiten definir ACs a utilizar sobre sujetos para los cuales no hay un AC en la lista
- Escenario
 - $ACL(o) = \{ (Pablo, x) (*, r) \}$
- Forma 1: Override
 - Si el sujeto no tiene un AC, utilizarlo. Sino el default

En el ejemplo, Pablo puede ejecutar o

- Forma 2: Augment
 - Partir del default y agregar ACs explicitos

En el ejemplo, Pablo puede leer y ejecutar o

Revocación

- ¿Como quitar derechos de un sujeto a un objeto?
 - El dueño quita el AC del sujeto del ACL
 - O quita el/los derechos necesarios dentro del AC
- Si hay transferencia de permisos
 - Mucho más complejo
 - Puede requerir borrar en cascada permisos delegados

S1 delega en S2. S2 delega en S3
S1 revoca a S2. Acto seguido S3 delega en S2

Ejemplo: Windows (archivos)

- Derechos: leer, escribir, ejecutar, borrar cambiar permisos, tomar control
- ACLs:
 - Incluyen usuarios y grupos
 - Tres posibles estado por derecho
 - Otorgado (+)
 - No asignado ()
 - Revocado (-)
 - Los derechos están agrupados: No access (todos-), read (leer+, ejecutar+), change(leer+,escribir+,ejecutar+,borrar+), full control (todos +)

Ejemplo: Windows (archivos)

- Acceso a un archivo
 1. Buscar todos los ACs en el ACL del archivo que referencien al usuario y a sus grupos
 2. No hay ningún AC → **denegado**
 3. Acceso revocado en un AC → **denegado**
 4. Al menos un AC permite el derecho → **aceptado**
 5. **denegado**

Listas de Capacidades

- Representan filas de una matriz de acceso

	Obj1	Obj2	Obj3	Obj n
Usuario1	r	r	r x				
Usuario2		o x					
...							
...							
Usuario n	o x						

CAPs:

usuario1: { (obj1, r), (obj2, r), (obj3, rx) }

usuario2: { (obj2, ox) }

usuario n: { (obj1, ox) }

...

Definición

- Sean
 - S conjunto de sujetos
 - O conjunto de objetos
 - R conjunto de acciones (derechos)
- $CAP(s) = \{ (o_i, r_i) / o \in O, r \subseteq R \}$
 - (o_i, r_i) implica que s accede a o_i con cualquier derecho de i_r
- s no tiene derechos sobre objetos que no estén en $CAP(s)$



Principio de denegar por defecto

Funcionamiento

- Las capacidades funcionan como entradas
 - La posesión indica derechos del sujeto sobre el objeto
 - Ejemplo: La capacidad (o1, rwx) puede ser presentada por diferentes sujetos
- A diferencia de los ACLs, el sistema no controla estos datos
- Deben implementarse mecanismos de protección
 - Evitar que un usuario altere Capacidades
 - Evitar que un usuario cree Capacidades

Implementación

- Tags
 - Marcas de bits controladas por hardware que impiden la modificación de registros en procesos de bajo privilegio
- Paging / segmentos protegidos
 - Las capacidades se almacenan en un segmento de memoria marcado como de solo lectura
 - Los procesos acceden indirectamente a dichas capacidades
 - Sino se podrían copiar las capacidades
 - Ejemplo: Descriptores de archivo en Linux

Implementación

- Criptografía
 - Asociar a cada capacidad un hash criptográfico cifrado con una clave conocida solo por el sistema
 - Al presentar la capacidad, el sistema verifica el hash

Controles

- Control de copia
 - Como la tenencia de una capacidad implica acceso, se debe restringir su copia
 - Acceso indirecto a las capacidades
 - Copia controlada por el sistema
- Amplificación
 - Posibilidad de contar con capacidades extendidas temporalmente
 - Al ejecutar ciertas funciones
 - Ejemplo: User mode vs Kernel mode

Revocación

- Implica revisar todas las listas de capacidades
 - Demasiado costoso
 - A veces imposible (sistemas remotos)
- En la práctica: indirección
 - Las capacidades son índices dentro de una tabla que no es accesible a los procesos
 - Se invalida la entrada correspondiente en la tabla

Ejemplo de Capacidades

- Sistema distribuido de archivos Tahoe
- Para acceder a un archivo se requiere una capacidad
- Hay capacidades de escritura, lectura y verificación:
 - URI:CHK:6hwdguhr5dvgte3qhosev7zszq:lgi66a5s6gchcu4yyaji;3:10:8448

Clave de encriptación

Info de validación

ACLs y Capacidades

- Ambos modelos son teóricamente equivalentes
- ACLs: dado un objeto ¿Quiénes pueden usarlo y como?
 - Asociado a procesamiento imperativo
 - Históricamente, el más desarrollado
 - Ejemplo: Windows / Linux
- Capacidades: dado un sujeto ¿Qué objetos puede acceder y como?
 - Asociado a procesamiento declarativo
 - Ejemplo: IDSs - Sistemas de respuesta de incidentes

Secretos compartidos

- Implementación de políticas de separación de privilegios
- Metodo (t,n) -threshold
 - Se crean n partes o n capacidades (sombras)
 - Cualesquiera t permiten acceder al objeto
 - Cualesquiera $k < t$ NO permiten acceder al objeto
- Implementación
 - Control via sistema
 - Criptografía: threshold cryptography

Metodo de Shamir

- Construcción de un método (t,n) threshold
- Principio:
 - Un polinomio de grado t puede ser especificado mediante su evaluación en t puntos diferentes
- Construcción:
 - Armar $P(x) = a_t x^t + a_{t-1} x^{t-1} + \dots + a_1 x + a_0 \text{ mod } p$
 - Sea s el secreto a compartir.
 - $p > s, p > n, a_0 = s$
 - $P(1), P(2), \dots P(n)$ son las sombras

Metodo de Shamir

- Reconstrucción del secreto

- Sean $(s_{i_1}, s_{i_2}, s_{i_t}) / s_{i_1} = P(i_1)$, t sombras cualesquiera
- Se interpola el polinomio (metodo de Lagrange)

$$P(x) = \sum_{a=1}^t \left(s_{i_a} \prod_{b=1, b \neq a}^t \frac{x - i_b}{i_a - i_b} \right)$$

- Se evalúa $P(0) = s$

Ejemplo

- Secreto a compartir: $s = 7$, esquema (3,5)
- $P(x) = 5x^2 + 3x + 7 \mod 11$
- Sombras:
 - $P(1) = 5 + 3 + 7 \mod 11 = 4$
 - $P(2) = 20 + 6 + 7 \mod 11 = 0$
 - $P(3) = 45 + 9 + 7 \mod 11 = 6$
 - $P(4) = 80 + 12 + 7 \mod 11 = 2$
 - $P(5) = 125 + 15 + 7 \mod 11 = 4$
 - Sombras: (1,4), (2,0), (3,6), (4,2), (5,4)

Ejemplo

- Sombras: (1,4), (2,0), (3,6), (4,2), (5,4)
- Tomando tres cualesquiera:
 - (2,0) (3,6) (5,4)

$$P(x) = \sum_{a=1}^t \left(s_{i_a} \prod_{b=1, b \neq a}^t \frac{x - i_b}{i_a - i_b} \right)$$

$$P(x) = \left[0 \frac{(x-3)(x-5)}{(2-3)(2-5)} + 6 \frac{(x-2)(x-5)}{(3-2)(3-5)} + 4 \frac{(x-2)(x-3)}{(5-2)(5-3)} \right]$$

$$P(x) = \left[0 - 3(x^2 - 7x + 10) + (4/6)(x^2 - 5x + 6) \right] \bmod 11$$

$$P(x) = 5x^2 + 3x + 7$$

$$s = P(0) = 7$$

Intentar con (1,4), (3,6), (4,2)

Oauth (2.0)

- Open Standard for Authorization
- Comienza en 2006 por necesidades de Twitter
- Oauth 1.0 – Publicado en 2010
 - RFC 5849
 - Estructura similar a OpenID
- Oauth 2.0 – Publicado en octubre 2012
 - Mucho mas simple que la versión anterior
 - No es compatible
 - Es el estándar de facto para sitios públicos

Oauth 2.0

- Permite al dueño de un recurso delegar en una aplicación el acceso al mismo
- Roles:

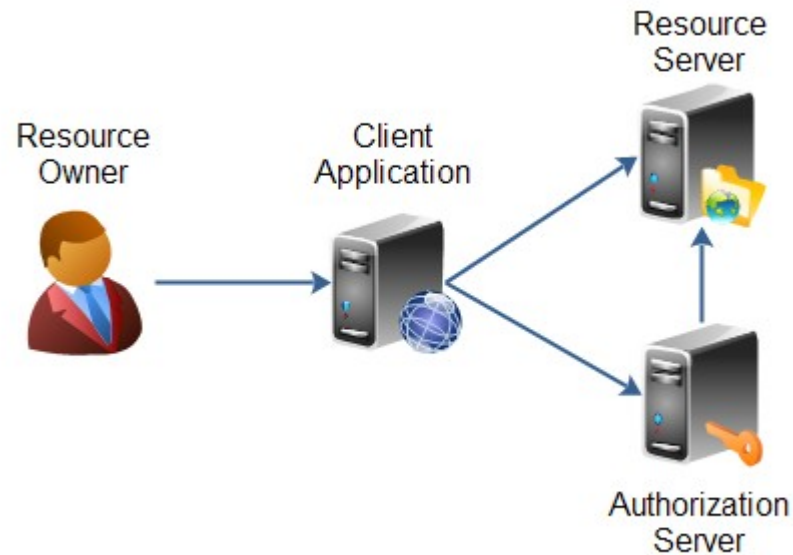


Imagen obtenida de: <http://tutorials.jenkov.com/oauth2/>

Oauth 2.0 – Dance

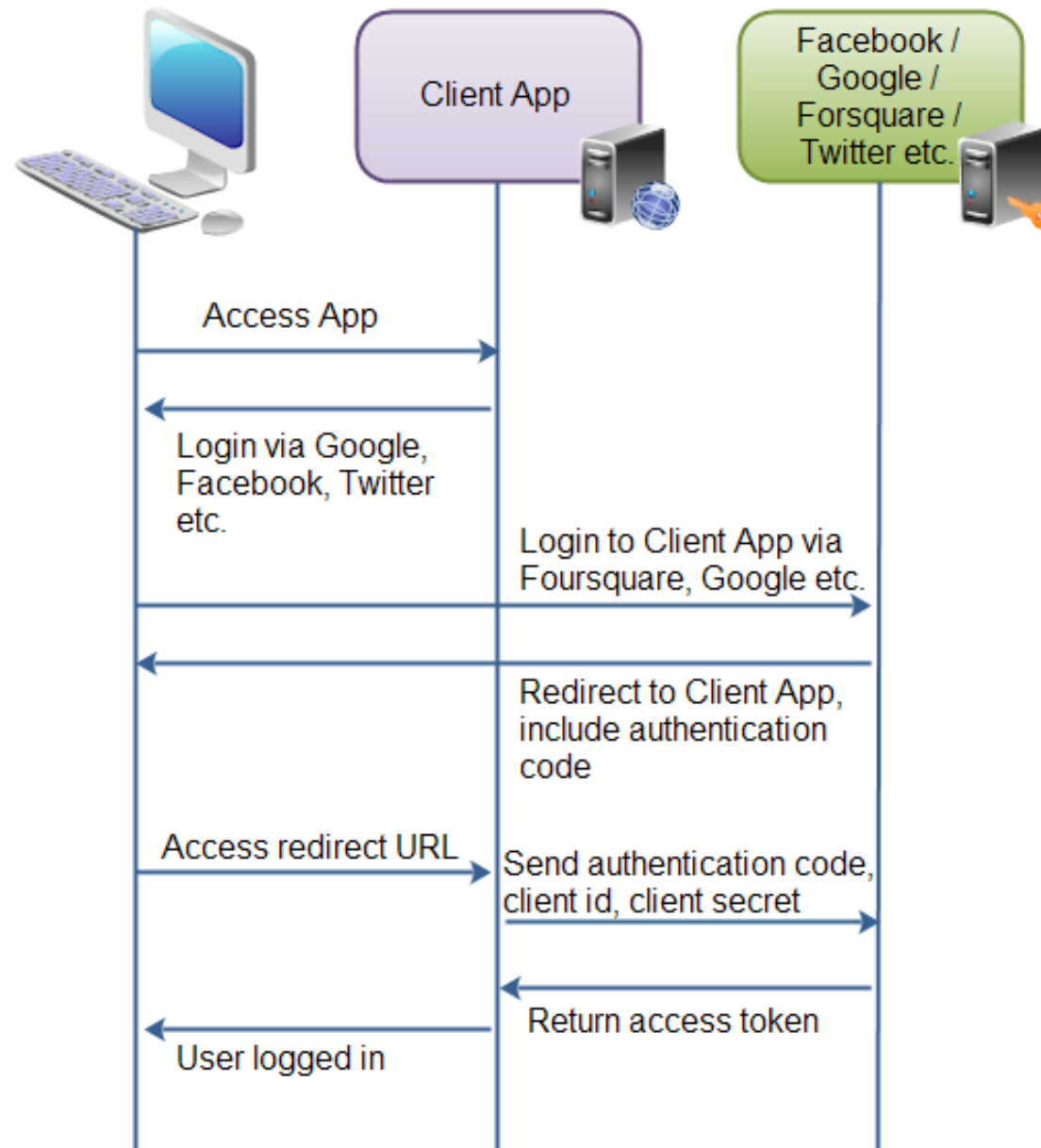


Imagen obtenida de: <http://tutorials.jenkov.com/oauth2/>

Oauth 2.0 – Tipos de clientes

- Confidencial

- Típicamente una aplicación en un servidor
- Puede mantener un secreto compartido con el servidor
 - Client-secret

- Publico

- Típicamente una aplicación desktop, movil o que corre en un navegador
- No puede mantener un secreto de forma confiable

Información del cliente

- Los clientes deben registrarse por adelantado
- Cada cliente requiere:
 - Client ID – Identificador único
 - Client Secret – Secreto compartido
 - Solo para clientes confidenciales
 - Redirect URI(s)
 - Lista de direcciones validas a donde redirigir los accesos

Oauth 2.0 – Grant types

- Authorization Code
 - El servidor de autorización retorna un código
 - El cliente consigue el access token con el código, su client id y su client secret.
- Implicit
 - El servidor de autorización retorna directamente el access token
- Resource Owner Password Credentials
 - En lugar de redirigir el pedido, el cliente captura y envía usuario y contraseña
- Client credentials
 - Autorización a nivel cliente (via cliente secret)

Oauth 2.0 – Ejemplo de mensajes

- Login redirect (URL)

```
https://  
login.salesforce.com/  
services/oauth2/authorize?  
response_type=code&client_i  
d=8483756923465.as.org&redi  
rect_uri=https%3A%2F  
%2Fwww.example.com%2Fback
```

Oauth 2.0 – Ejemplo de mensajes

- Callback response (URL que recibe el callback)

`https://app.example.com/
oauth_callback?
code=aWekysIEeqM9PiThEfm0C
nr6MoLIfwWyRJcq0qHdF8f9INo
kharAS09ia7UNP6RiVScerfhc4
w%3D%3D`

Oauth 2.0 – Ejemplo de mensajes

- Token Request (post al Provider – form url encoding)

```
code=aWekysIEeqM9PiThEfM0C  
nr6MoLIfwWyRJcq0qHdF8f9INo  
kharAS09ia7UNP6RiVScerfhc4  
w==&grant_type=authorizati  
on_code&client_id=ffdskhfe  
ihoaw&client_secret=khfeai  
hdiu38nd&redirect_uri=...
```

Oauth 2.0 – Ejemplo de mensajes

Response token (json response body)

```
{  
  "id": "https://login.salesforce.com/id/  
00D5000Z3ZEAW/00550001fg50AQ",  
  "issued_at": "1296458209517",  
  "refresh_token": "5Aep862eW05D.7wJBuW5aaARbb  
xQ83jMRnbFNT5R8X2GUKNA==",  
  "instance_url": "",  
  "signature": "0/1Ldval/  
TIPf2tTgTKUAXRy44VwEJ7ffsFLMWFcNoA=",  
  "access_token": "00D500000000IZ3Z!  
AQ0AQDpEDKYsn7ioKug2aSmgCjgrPjG9eRLz"  
}
```

Oauth 2.0 – OpenID Connect

- Agrega autenticacion a Oauth2
- Se basa en un tipo especial de token (bearer token) al obtener el access token:

```
"id_token": "  
eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImIzcyI6ICJodHRw  
Oi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5NzYxMDAxIiw  
KICJhdwQiOiAic2ZCaGRSa3F0MyIsCiAibm9uY2UiOiAib0wUzZfV3pBMk1qIi  
wKICJleHAiOiAxMzExMjg5NzYxMDAxIiwKICJmcmVudCI6IjE5eV8hZ  
1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqr00F4daGU96Sr_P6qJp6IcmD3HP9  
90bi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJNqeGpe-gccM  
g4vfKjkM8FcGvnzZUN4_KSP0aAp1t0J1zZwgjxqGByKHi0tX7TpdQyHE5lcMiKP  
XfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJb0EoRoSK5hoDalrcvR  
YLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVvk4XUVrWOLrLl0  
Nx7RkKU8NXNHq-rvKMzqg  
",  
"access_token": "jdj2V32hkKG"  
"token_type": "Bearer"
```



Base 64 - URLSAFE

Oauth 2.0 – OpenID Connect

- El bearer id token es un documento JSON (JWT – Json Web Token)
 - Firmado digitalmente por el provider

```
{
  "sub"      : "alice",                <<< Subject
  "iss"      : "https://openid.c2id.com", <<< Issuer
  "aud"      : "client-12345",         <<< Audience - destinatario
  "nonce"    : "n-0S6_WzA2Mj",        <<< Nonce
  "auth_time": 1311280969,              <<< Cuando fue autenticado
  "acr"      : "c2id.loa.hisec",       <<< Como fue autenticado (OPT)
  "iat"      : 1311280970,             <<< Issue Time
  "exp"      : 1311281970,             <<< Expiration time
}
```

Lectura recomendada

Capítulo 15

Computer Security Art and Science
Matt Bishop

OAuth2.0 – RFC 6749