CHAPTER 14

System Protection

(20 Questions)

01: What are the ***Goals of Protection***?

02: What are the ***principles of Protection***?

03: What is ***Principle of Least Privilege***?

04: How many ***types for Grain Aspect***?

05: How can the ***Domain*** be?

06: What are the ***Access Matrix Properties***?

07: Who can define access column for the created object?

08: What is the ***use of access matrix***?

09: What is ***Access Matrix Design***?

10: How is ***the Implementation of Access Matrix***?

11: What is the ***Comparison of Implementations of Access Matrix***?

12: What is ***Access Control***?

13: What does ***Solaris 10 provide***? Why?

14: Mention the **Various options to remove the access right of a domain to an object**?

15: What is the ***role of Revocation of Access Right with Access-List scheme***?

16: What is the ***role of Revocation of Access Right with Capability-List scheme***?

17: Give ***two examples for Capability-Based Systems***?

18: What is ***Specification of Protection in a Programming Language***?

19: What can ***Language implementation provide***?

20: What is ***Interpret Protection Specifications to generate***?

End of Questions (Chapter 14).

Chapter 14

System Protection

1: **أهداف الحماية (Goals of Protection):**

a) في نموذج حماية واحدة، الكمبيوتر يتألف من مجموعة من الـ Objects, Hardware, or Software.
b) كل object له اسم فريد (unique) ويمكن الوصول إليه من خلال a well-defined set من العمليات.
c) مشكلة الحماية – ضمان أن يتم الوصول إلى كل object بالشكل الصحيح وتتم فقط بواسطة هذه العمليات التي تسمح لها بالقيام بذلك.

2: **مبادئ الحماية (Principles of Protection):**

a) المبدأ التوجيهي (Guiding principle): مبدأ الأقل امتيازا (principle of least privilege).
b) يجب الاعتبار من جانب (aspect) "grain".
c) المجال (Domain) قد يكون مستخدم أو عملية أو اجراء.

3: **مبدأ الأقل امتيازا (principle of least privilege):**

- البرامج والمستخدمين والأنظمة يجب ان تعطى امتيازات كافية لأداء مهامهم.
- الضرر محدود إذا كان له علة (bug) أو تحصل على سوء معاملة (abused).
- يمكن أن تكون ثابتة () "خلال حياة النظام، خلال حياة العمل).
- أو ديناميكيا (تغيرت عن طريق عملية عند الحاجة) – تبديل المجال أو تصعيد الامتياز (privilege escalation).
- "الحاجة للمعرفة" مفهوم مماثل فيما يتعلق بالحصول على البيانات.

4: **Grain aspect له نوعان:**

- **الخام (Rough-grained):** إدارة الامتياز أسهل، وأبسط ولكن least privilege الان منجز في أجزاء كبيرة (large chunks). مثل عمليات اليونكس التجارية إما لها قدرات المستخدم المرتبطة بها أو من الجذور.
- **النهاية (Fine-grained):** الإدارة هنا أكثر تعقيدا وأكثر حملا (more overhead) ولكنه أكثر حماية.

5: **المجال (Domain) قد يكون** مستخدم أو عملية أو اجراء.

6: **خصائص مصفوفة الوصول (Access Matrix Properties):**

1) عرض الحماية كـ مصفوفة (matrix).
2) Rows تمثل الـ domains.
3) Columns تمثل الـ objects.

7: **المستخدم الذي أنشأ الـ object** هو من يستطيع تعريف access column لهذا الـ object.

8: **استخدام مصفوفة الوصول (use of access matrix)** يمكن توسيعه ليشمل الحماية الديناميكية مثل:

- العمليات للإضافة وحذف access rights.
- Access rights خاص.
- Copy و Owner ينطبق على object.
- Control ينطبق على domain object.

9: **تصميم مصفوفة الوصول (access matrix)** تفصل الآلية من السياسة ولكنه لا يحل مشكلة الحجز العام (general confinement problem):

- الآلية (Mechanism):
  - نظام التشغيل يقوم بتوفير مصفوفة الوصول + القوانين.
  - الضمانات من أن المصفوفة تكون معالجة فقط من قبل وكلاء مصرح بهم وأن القوانين منفذة بصرامة (strictly).
- السياسة (Policy):
  - المستخدم هو من يملي يضع السياسة.
  - من يستطيع الوصول إلى object ما وبأي طريقة.

10: **عادة تكون تنفيذ مصفوفة الوصول عن طريق مصفوفة متفرقة (a sparse matrix) وفيها خيارات:**

- الخيار الأول: الجدول الشامل (Global Table)
  - مخزن أوامر ثلاثي (domain, object, right-set) في الجدول.
  - ولكن الجدول قد يكون كبير – أي سيكون غير مناسب في الذاكرة الرئيسية.
  - صعب لتجميع الكائنات (to group object) اعتبر الكائن أن جميع domains ممكن قراءتها.
- **الخيار الثاني: قوائم الوصول للكائنات (Access lists for objects)**
  - كل عامود (column) منفذ كقائمة الوصول لـ كائن واحد (one object).
  - نتيجة قائمة كل كائن يتألف من زوجين من الأوامر <domain, rights-set> المعرفة لكل domains مع مجموعة غير خالية من الـ access rights للكائن.
  - متوسعة بشكل سهل ليحتوي على default set وأيضا السماح للوصول.
- **الخيار الثالث: قائمة القدرة للمجالات (Capability lists for domains)**
  - بدلا من object-based القائمة تكون domain based.
  - قائمة القدرة على domain هي قائمة الكائنات جنبا إلى جنب مع عمليات تتيح عليها.
  - الكائن يكون متمثل باسمه وعنوانه ويدعى بالقدرة (capability).
  - امتلاك القدرة (Possession of capability) تعني الوصول مسموح به.
  - قائمة القدرة مرتبطة مع الـ domain ولكن لا يمكن الوصول إليها مباشرة من قبل الـ domain:
    - Rather، الكائن المحمي، محتفظ بها من قبل OS والوصول إليها بشكل غير مباشر.
    - مثل Source pointer.
    - Idea يمكن توسيعها إلى تطبيقات.
- **الخيار الرابع: مفتاح القفل (Lock-key)**
  - حل وسط (Compromise) بين قوائم الوصول وقوائم القدرة.
  - كل كائن له قائمة من الأساليب القليلة الفريدة (unique bit patterns) تسمى locks.
  - كل domain له قائمة من الأساليب القليلة الفريدة (unique bit patterns) تسمى keys.
  - العملية في الـ domain يمكن الوصول فقط إلى كائن لو كان الـ domain له key يطابق واحد من الـ locks.

11: **المقارنات بين تطبيقات مصفوفة الوصول (Comparison of Implementations of Access Matrix):**

- **هناك العديد من المفاضلات (trade-off) لأخذ الاعتبار:**
  - الجدول الشامل (Global Table) بسيط ولكنه من الممكن أن يكون كبير.
  - قوائم الوصول (Access Lists) تتوافق مع (correspond) احتياجات المستخدمين:
    - يحدد مجموعة access rights للـ domain غير المترجمة وهي صعبة للغاية.
    - كل وصول إلى كائن يجب أن يتم فحصه. والعديد من الكائنات والـ access rights بطيئة.
  - قوائم القدرات (Capability Lists) مفيدة لمعلومات الترجمة (localization info) للعملية المعطاة، ولكن إلغاء الإمكانيات قد تكون غير فعالة.
  - مفتاح القفل (Lock-Key) فعالة ومرنة، والمفاتيح قد تمر بشكل حر من الـ domain إلى domain، وسهل الإلغاء.
- **العديد من الأنظمة تستخدم مزيج من قوائم الوصول والامكانيات:**
  - أول access إلى كائن => access list قامت بالبحث:
    - لو تم السماح لها، فإن capability قد تم انشاءه وارفق في العملية، وليس من الضروري أن يتم التحقق من المداخل الإضافية (Additional accesses).
    - بعد أخر access، يتم تدمير capability.
    - يعتبر نظام الملف مع ACLs لكل ملف.

12: **التحكم في الوصول (Access Control):** هو حماية يمكن تطبيقها على الموارد غير الملف.

13: **Solaris 10** يقوم بتوفير التحكم في الوصول المستند على الدور (RBAC: Role-Based Access Control) **لتنفيذ least privilege:**

- الامتياز هو right لتنفيذ استدعاء النظام أو استخدام خيار داخل استدعاء النظام.
- يمكن تعيينه إلى العمليات.
- المستخدمين الذين تم تعيينهم الأدوار التي تمنحهم الوصول إلى الامتيازات (privileges) والبرامج.
  - تمكن الدور عبر كلمة مرور للحصول على امتيازاتها (its privileges).
- مشابه لـ Access Matrix.

14: **هناك خيارات متنوعة لإزالة access right of a domain من الـ object:**

1. Immediate vs, delayed
2. Selective vs, general
3. Partial vs. total
4. Temporary vs. permanent

15: **دور الـ Revocation مع مخطط قائمة الوصول:** حذف access right من قائمة الوصول

- بسيط: يقوم بالبحث عن قائمة الوصول ويزيل المدخل.
- مباشر أو عام أو انتقائي (selective) كلي أو جزئي، دائم أو مؤقت.

16: **دور الـ revocation مع مخطط قائمة القدرة:** مطلوب لتحديد القدرة في النظام قبل أن يتم إلغاء القدرة

- إعادة اكتسابها (Reacquisition): حذف متكرر، مع الطلب والحرمان إذا ألغي.
- عودة المؤشرات (Back-pointers): مجموعة من المؤشرات من كل كائن إلى كل القدرات من ذلك الكائن.
- المراوغة (Indirection): نقاط القدرة لمدخل الجدول الشامل حيث نقاط الكائن يقوم بإزالة المدخل من الجدول الشامل وهو ليس انتقائي (selective).
- المفاتيح: bits موحد مرتبط مع القدرة، وتوجد عند يتم انشاء القدرة:
  - Master Key تكون مرتبطة مع الكائن، والمفتاح يتطابق مع master key للوصول.
  - الازالة (Revocation): ينشئ master key جديد.
  - قرار سياسي لمن يستطيع انشاء أو تعديل المفاتيح – صاحب الكائن أو أخرين.

17: **من أمثلة النظم القائمة على القدرة (Capability-Based Systems):**

- **Hydra:**
  - مجموعة ثابتة من الـ access rights معرفة ومفسرة من قبل النظام:
    a) قراءة وكتابة وتنفيذ كل مقطع ذاكرة.
    b) يمكن للمستخدم أن يعلن حقوق إضافية أخرى وتسجيل ذلك مع نظام الحماية.
    c) عملية الوصول يجب أن يحمل القدرة ومعرفة اسم العملية.
    d) تضخيم الحقوق (Rights amplification) المسموح به من الإجراءات الجديرة بالثقة لنوع معين.
  - تفسير حقوق المعرفة من قبل المستخدم يقوم فقط من قبل برنامج المستخدم؛ والنظام يقوم بتوفير حماية الوصول لاستخدام هذه الحقوق.
  - العمليات على كائنات معرفة اجرائيا – الإجراءات هي كائنات يكون الوصول إليها بشكل غير مباشر من خلال القدرة.
  - يحل المشاكل المتبادلة للأنظمة الفرعية المشبوهة.
  - يتضمن مكتبة من إجراءات أمنية مكتوبة مسبقا.
- **Cambridge CAP System:**
  - بسيطة ولكنها قوية.
  - قدرة البيانات: يقوم بتوفير معايير القراءة والكتابة والتنفيذ لشرائح (segments) التخزين الشخصية المرتبطة مع الكائن المنفذة في الـ microcode.
  - قدرة البرنامج: التفسير (interpretation) المغادر إلى النظام الفرعي، من خلال إجراءاتها المحمية.
    a) فقط لها access لنظامها الفرعي الخاص بها.
    b) المبرمجين يجب عليهم تعلم مبادئ وتقنيات الحماية.

18: **مواصفات الحماية في لغة البرمجة** تسمح بوصف رفيع المستوى من السياسات للتخصيص واستخدام الموارد.

19: **تنفيذ اللغة** تستطيع توفير برنامج تقوية الحماية (protection enforcement) عندما يكون فحص الاجهزة المدعومة تلقائيا غير متوفرة.

20: **تفسير مواصفات الحماية للتوليد (to generate)** تدعو أياً كان نظام الحماية التي تم توفيرها بواسطة الجهاز ونظام التشغيل.

END of chapter 14