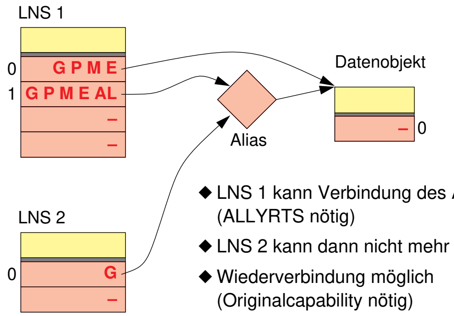
12 Rückruf von Capabilities

- Anwender möchte ausgegebene Capabilities für ungültig erklären
  - sofortiger Rückruf — Rückruf nach einiger Zeit erst wirksam
  - dauerhafter Rückruf — Rückruf nur zeitlich begrenzt wirksam
  - selektiver Rückruf — Rückruf für alle Benutzer eines Objekts
  - partieller Rückruf — Rückruf aller Rechte an einem Objekt
  - Recht zum Rückruf; Rückruf des Rückrufrechts
- ★ Hydra setzt sogenannte Aliase ein
  - Alias ist eine Indirektionsstufe zu Capabilities
  - Statt auf ein Objekt können Capabilities auf Aliase verweisen und diese wiederum auf andere Aliase oder schließlich auf das eigentliche Objekt
  - Verbindung vom Alias zum Objekt kann gelöst werden: Fehler beim Zugriff
  - Recht zum Lösen der Verbindung ***AL**LYRTS*
    (engl. *ally* = verbünden)

12 Rückruf von Capabilities (2)

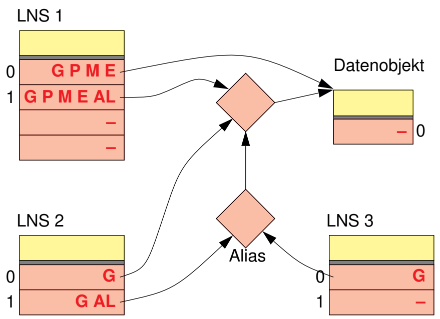
- Beispiel: Weitergabe einer rückrufbaren Capability



- LNS 1 kann Verbindung des Alias lösen (ALLYRTS nötig)
- LNS 2 kann dann nicht mehr zugreifen
- Wiederverbindung möglich (Originalcapability nötig)
- Aliase können hintereinander auftreten

12 Rückruf von Capabilities (3)

- Beispiel: Aliasketten

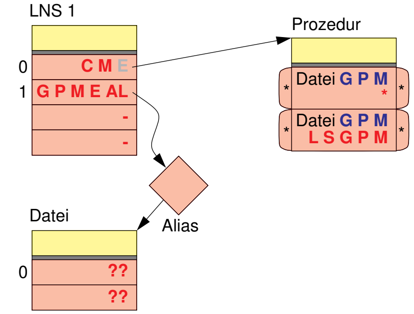


12 Rückruf von Capabilities (4)

- ▲ Problem: Rückruf während der Bearbeitung eines Objekts
  - inkonsistente Zustände möglich
- ★ Lösung in Hydra
  - Parameter-Capabilities, die durch eine rechteverstärkende Parameterschablone angenommen werden, zeigen auf das Originalobjekt
- ▲ Nachteil
  - nicht vertrauenswürdige Prozeduren können rückruffreie Capability erlangen
  - Problem fällt in die selbe Kategorie wie rechteverstärkende Parameterschablonen an sich

12 Rückruf von Capabilities (5)

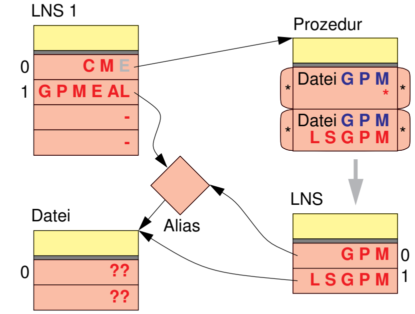
- Beispiel:



Call(0, 2, 1, (–E), 1, (–E))

12 Rückruf von Capabilities (5)

- Beispiel:



Call(0, 2, 1, (–E), 1, (–E))

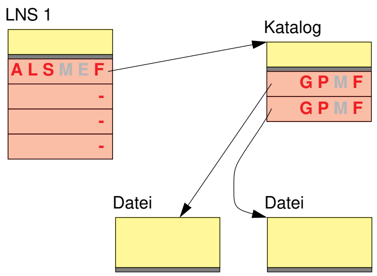
13 Garantierter Zugriff

- Schutz vor Rückruf
  - Modifizierende Benutzer eines Objekts
    - kooperierende Benutzer: Rückruf keine Gefahr
    - nicht kooperierende Benutzer: Rückruf nötig
  - Benutzer eines Objekts ohne modifizierende Zugriffe
    - Aufruf von Prozeduren ist unkritisch, da Aufrufe nicht rückrufbar sind
    - lesende Zugriffe auf Objekte sind kritisch:
      Verhindern des Rückrufs ist jedoch nicht ausreichend
      – Daten im Objekt könnten gelöscht oder verfälscht werden
      – Capabilities im Objekt oder deren Rechte könnten entfernt werden

★ Hydra führt das Einfrierrecht *(Freeze right FRZRTS)* ein
  - Einfrieren nimmt Modifikationsrecht weg
  - ein Objekt kann nur gefroren werden, wenn alle Capabilities in der C-List bereits das Einfrierrecht haben

13 Garantierter Zugriff (2)

- Beispiel:



14 Bewertung von Hydra

- Hydra demonstrierte die Beherrschbarkeit einer ganzen Reihen von Sicherheitsproblemen
  - ◆ Ergebnisse flossen in eine ganze Reihe von Systemen
  - ◆ reine Capability-basierte Systeme haben sich jedoch nie durchgesetzt
- Hydras Probleme
  - ◆ lagen im wesentlichen nicht am Capability-Mechanismus
  - ◆ es gabe keine vernünftigen Editoren und Compiler
  - ◆ Hardware besaß keine Spezialhardware zur Unterstützung von Paging

D.5 Schutz von Dateien

1 Zugriffkontrolle in UNIX

- Beispiel für Zugriffskontrolle in einem klassischen Betriebssystem
  - ➤ zu schützende Objekte: Verzeichnisse und Dateien
  - ➤ Subjekte: Benutzer, Benutzergruppen und Prozesse
- ➥ Zugriffskontrollkonzept zunächst sehr beschränkt
  - ➤ spezielle Objekte (Geräte, Prozesse, Arbeitsspeicher) werden als Dateien modelliert (*special files*)
  - ➤ Konzept wurde nachträglich auch auf einige weitere Systemobjekte (Semaphore, Shared Memory Segmente, Message Queues) erweitert
- ➥ Zugriffskontrolle deckt die meisten relevanten Objekte im System ab
  - ➤ ABER: die Abbildung auf den Typ "Datei" mit den Operationen *read*, *write*, *ioctl* ist äußerst grobgranular

1 Zugriffkontrolle in UNIX (2)

- Zugriffsrechte pro Datei
  - ➤ lesen, schreiben, ausführen
  - ➤ ggf. ergänzt durch individuellere ACLs
- Zugriffsrechte im Dateibaum
  - ➤ lesen, schreiben (=Einträge erzeugen/löschen), durchgreifen
  - ➤ Rechte auf Directories limitieren ggf. den Zugriff auf darunterliegende Dateien
  - ➤ Durchgriffsrecht ohne Leserecht wird oft zum "Verstecken" von Dateien verwendet → fragwürdiger Schutz
  - ➤ Schreibrecht auf Directory kann mit Rechten darunterliegender Dateien kollidieren
    - – Löschen und Neuanlegen einer Datei, auf die man kein Schreibrecht hätte ist möglich

2 Zugriffskontrolle unter Windows

- Authentisierung eines Benutzers führt zur Vergabe eines *Access Tokens* (enthält Security-Ids des Benutzers und seiner Gruppen, Default-ACL, …)
  - ➤ entspricht einer Credential-Struktur
  - ➤ Access Token wird Prozess zugeordnet und an neu erzeugte Prozesse vererbt
- Objekte enthalten Security Descriptor
  - ➤ SID des Benutzers und der Gruppe
  - ➤ Discretionary ACL mit Liste von ACE (Access Control Elements)
    - – Erlaubnis- und Verbots-ACEs
    - – geben SID und erlaubte/verbotene Operationen an
    - – pro Objekt 16 verschiedene Zugriffsrechttypen möglich (z. B. synchronize, write_owner, write_DAC, read, write)
- Zugriffsberechtigung bei open
  - ➤ anschließend Vergabe eines Object Handles (Capability) an den Prozess zur Vorlage bei den konkreten Zugriffe

D.6 Verschlüsselung von Dateien

1 Motivation

- 2004 in Londoner Taxis liegen geblieben:
  4.973 Laptops, 5.838 PDAs, 63.135 Mobiltelefone
- Reuters, 2005: Citigroup meldet den Verlust von Bändern mit den Daten - einschl. Sozialversicherungsnummern - von 3,9 Mio. Kunden
- Leyden, 2004: Kundendatenbank und die aktuellen Zugriffscodes für das Intranet eines der größten Finanzdienstleister in EU waren auf einer Platte, die bei eBay zum Verkauf angeboten worden war
- Noguchi, 2005: Laptop mit Namen und Sozialvers.nummern von 16.500 MCI Mitarbeitern wurde gestohlen.
- Reuters, 2005: Iron Mountain ... meldet Verlust von Bändern mit den Daten von 600.000 Time Warner Arbeitern

...

2 Anforderungen

- Schutz der Dateiinhalte
- Schutz der Metadaten
- Schutz muss auch gewährleistet sein
  - ➤ bei direktem Zugriff auf Datenträger
  - ➤ Transport der Daten über Netzwerke
  - ➤ für Datensicherung und Archivierung
    - – langfristige Schlüsselaufbewahrung
    - – automatische Schlüsselaktualisierung
- Mehraufwand für Benutzer begrenzen
  - ➤ Schutz möglichst transparent gewährleisten
- Einsatz in Mehrbenutzerumgebungen
  - ➤ Gruppenzugriffe
- Umgang mit Schlüsselverlust

3 Programme zur Dateiverschlüsselung

- Benutzer kann individuell für einzelne Dateien entscheiden, ob und wie sie verschlüsselt werden sollen
- Realisierung
  - ◆ Werkzeuge
    - ➤ PGP, GnuPG, Ncrypt
  - ◆ Betriebssystemmechanismen

\+ Verschlüsselung nur dort, wo es tatsächlich erforderlich ist

\+ unterschiedliche Schlüssel für unterschiedliche Dateien möglich

– Datei-Metadaten (Name, Attribute) sind nicht verschlüsselt

– großes Risiko undichter Stellen (*Leakage*)

4 Undichte Stellen bei Verschlüsselung einzelner Dateien

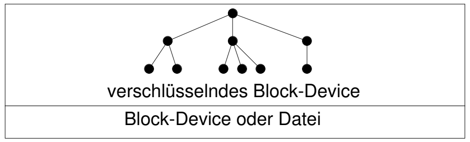
- Um mit den Daten zu arbeiten, müssen sie unverschlüsselt vorliegen
- Viele Programme erzeugen temporäre Kopien
  - ➤ bei Abstürzen des Programms oder des Rechners können die Kopien unbemerkt liegen bleiben
  - ➤ auch wenn die Dateien gelöscht werden, bleiben die Daten auf der Platte, bis der Platz wiederverwendet wird
- Absturz/Abschalten des Rechners während die Verschlüsselung noch läuft
- Swap-Partition enthält Teile des Arbeitsspeichers

➥ Verschlüsselte Daten sickern in unverschlüsselte Teile des Speichermediums durch

➥ Dateiverschlüsselung wird wirkungslos

5 Verschlüsselung von Dateisystem-Partitionen

- Schicht zwischen Dateisystem und Plattentreiber sorgt für Verschlüsselung
  - ➤ Pseudo-Plattentreiber - verhält sich "nach oben" wie eine Partition, ver-/entschlüsselt und greift auf Dateien oder "normale" Partition zu



  - ➤ Aufbau auf herkömmlichem Block-Device: alle Daten werden einheitlich verschlüsselt
  - ➤ Aufbau auf Datei: Benutzer kann sich mehrere kleine Dateisysteme schaffen, die unterschiedlich verschlüsselt sind
- ◆ Beispiel: *dm-crypt* für Linux

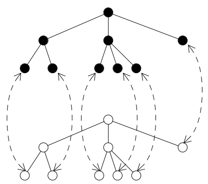
5 Verschlüsselung von Dateisystem-Partitionen (2)

+ Vorteile
  - ➤ Datei-Metadaten (Name, Attribute) werden mitverschlüsselt
  - ➤ alle Daten auf der Partition werden grundsätzlich verschlüsselt
  - ➤ Risiko undichter Stellen wird erheblich reduziert

– Nachteile
  - ➤ Dateibaum und Dateiattribute nicht mehr erkennbar
    - – z. B. kein selektives Backup mehr möglich
  - ➤ ein gemeinsamer Schlüssel für alle Daten der Partition
  - ➤ Austausch von Schlüsseln schwierig
  - ➤ so lange nicht ALLE Partitionen verschlüsselt werden bleibt das Risiko undichter Stellen
    - – problematisch vor allem bei kleinen, benutzerspezifischen Dateisystemen

6 Verschlüsselnde Dateisysteme

- Aufsatz eines verschlüsselnden Dateisystems auf einem normalen unverschlüsselten Dateisystem
  - ➤ *Stacked Filesystem*
- ◆ Grundprinzip:
  Inhalt und Meta-Daten von Dateien des verschlüsselnden Dateisystems werden in Dateien des darunterliegenden Dateisystems abgelegt



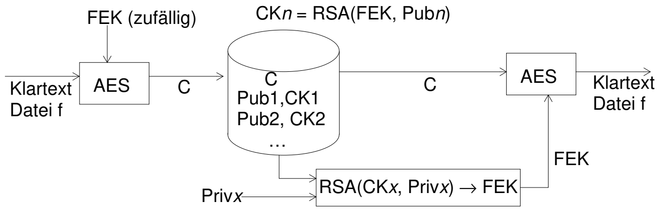
  - ➤ Struktur der Partition bleibt sichtbar (Directorystruktur und Dateiattribute)

6 Verschlüsselnde Dateisysteme (2)

- Design-Alternativen
- ◆ Verschlüsselung von Datei- und Directorynamen
  - + zusätzliche Sicherheit, da keine Schlüsse aus den Datei-Metadaten gezogen werden können
  - – Management wird komplexer (z. B. Backup-Restaurierung)
- ◆ Realisierung als Teil des Betriebssystemkerns
  - ➤ z. B. EFS unter Windows, eCryptfs in Linux
  - + bessere Performance
  - – weniger flexibel
- ◆ Realisierung als Anwendungsprozess (*Userland File System*)
  - ➤ z. B. CryptoFS oder EncFS unter Linux
  - ➤ mehr Flexibilität, schlechterer Performance
- ◆ Ein- oder Mehrbenutzerfähigkeit
  - ➤ wird ein einheitlicher Schlüssel verwendet oder kann jeder Benutzer eigene Schlüssel einsetzen?

7 Verschlüsselnde Dateisysteme — EFS

- seit Windows 2000, Erweiterungen in XP und Vista
  - ➤ Verschlüsselung mit AES, früher DES3 oder DESX
  - ➤ setzt auf NTFS auf (nicht auf FAT-Dateisystemen)
- Mehrbenutzerfähig
  - ➤ Datei mit zufälligem Schlüssel (FEK - File Encryption Key) verschlüsselt
  - ➤ FEK wird mit Public Key aller berechtigten Benutzer verschlüsselt (CK), Ablage in Attributen der verschlüsselten Datei



7 Verschlüsselnde Dateisysteme — EFS (2)

▲ Kritische Punkte

- Speicherung der privaten Schlüssel
  - ➤ idealerweise auf Smartcard (ab Vista unterstützt)
  - ➤ sonst verschlüsselt (mit Passwort der Benutzers oder Smartcard-Schlüssel) auf der Festplatte
    - – privater Schlüssel kann bei login in EFS-Prozess geladen werden und steht danach für den Benutzer zur Verfügung
    - – EFS-Prozess sperrt Seiten mit Schlüssel-Information gegen Paging
    - – im Hauptspeicher sind die privaten Schlüssel aber vorhanden
    - – Angriffe auf den EFS-Prozess?
    - – Benutzerpasswort meist deutlich schwächer als erzeugte Schlüssel
    - – Änderung des Benutzerpasswords durch Admin führt dazu, die privaten Schlüssel für den Benutzer nicht mehr lesbar sind (→ private Schlüssel außerhalb sichern!)

7 Verschlüsselnde Dateisysteme — EFS (3)

▲ Kritische Punkte (2)

- Schlüssel-Recovery
  - ➤ Recovery-Agent (z. B. Administrator) kann verschlüsselte Dateien restaurieren
  - ➤ FEKs werden auch mit Public Key des Recovery Agenten verschlüsselt
    - – aus X509-Zertifikat des Recovery Agenten
    - – wenn Zertifikat abgelaufen ist, muss der Recovery-Schlüssel bei erneutem Speichern einer Datei erneuert werden
  - ➤ Private Recovery Key kann außerhalb des Systems verwahrt werden (eigentlich MUSS! - sonst reicht es, Administrator-Rechte zu erlangen, um alle verschlüsselten Dateien entschlüsseln zu können)
    - – alte Schlüssel müssen aufbewahrt werden (für alte Dateien)

7 Verschlüsselnde Dateisysteme — EFS (4)

▲ Kritische Punkte (3)

- Verschlüsselung auf NTFS-Dateisysteme beschränkt
  - ➤ vom Benutzer für ein Verzeichnis und ggf. alle Unterverzeichnisse einstellbar
  - ➤ Dateien werden beim Kopieren auf andere Dateisysteme entschlüsselt
  - ➤ Transfer von Dateien über Netzverbindungen unverschlüsselt
- Verschlüsselung der privaten Schlüssel mit dem Benutzer-Passwort
  - ➤ schwaches Glied in der Kette
  - ➤ Problem wenn Administrator Benutzer-PW ändert
    ➥ privaten Schlüssel extrahieren und separat speichern