Vorlesung 13

Security

Peter B. Ladkin
ladkin@rvs.uni-bielefeld.de

Sicherheit, Security und Safety

- Sicherheit ist ein deutsches Wort
- Aber dazu korrespondieren zwei Begriffe
  - Safety
  - Security
- Beide beziehen sich auf den Schutz vor einem unerwünschten Ereignis

Safety und Security

- Safety
  - Schutz vor einer *unabsichtlichen*, ungewünschten Wirkung eines Systems
- Security
  - Schutz vor einer *absichtlichen*, ungewünschten Wirkung eines Systems

Ursachen

- Man versucht, die Ursachen herauszufinden
- Die Ursachen könnten in einer Kausalitäts-"Kette" dargestellt werden
  - Es ist keine Kette. Es ist ein Graph von kausalen Faktoren und ihrer Zusammenwirkung
  - Es gibt mehrere notwendige Ursachen, die zusammen eine hinreichende Ursachenmenge bilden

Ursachen

- "Notwendig" heißt: Wenn man eine Einzelursache "herausnimmt", wäre das Ereignis nicht eingetreten.
- "Hinreichend" heißt: Wenn alle Ursachen zusammen anwesend sind, tritt unvermeidlich das Ereignis ein.
- Also hat man eine hinreichende Menge von notwendigen Ursachen, sogenannte "INUS" Konditionen (Mackie, The Cement of the Universe, Oxford U. Press, 1974)

Der Unterschied in der Prophylaxe

- Safety
  - Analysiert man die Ursachen für einen Fall und vermeidet man eine der Ursachen, dann kann das Ereignis auf dieser Art nicht mehr passieren.
- Security
  - Problemverhalten sind Ziel-orientiert. Wenn man eine von den Ursachen vermeidet, wird ein Ersatz schnell gefunden. Also müssen passende Maßnahmen übergreifender sein.

Beispiel

- Der Unix-Befehl "`rm *`" ist gefährlich. Er löscht alle Dateien im aktuellen Verzeichnis
- Safetymaßnahme
  - `rm` so ändern, dass eine Bestätigungsfrage auf stdout ausgegeben wird, womit nach Eingabe von "yes" gelöscht bzw. von "no" abgebrochen wird
- Securitymaßnahme
  - `rm` so ändern, dass erst nach Eingabe des richtigen Passworts gelöscht wird

Umfang eines Securityproblems

- Einzeloperationen/Befehle
  - Semantisch korrekt aber Lücken in der Übersetzung in einer Maschine
    - Buffer Overflow
    - Höhere Benutzererlaubnis außerhalb einer Critical Section
  - Die Semantik fordert Vertrauen in den Benutzer
    - Allgemeine Erlaubnis: "Root Permission"
    - Änderung der Internet-Parameter (z.B. IP-Adresse) des Rechners
    - Ermöglicht das Lesen vertraulicher Dateien

Umfang eines Securityproblems

- Umgebung
  - Was darf Benutzer "Fred" ausführen?
  - Multi-level Security
    - "Admin" darf alles
    - "Fred" darf wenig
    - "Lucy" darf weniger als Fred
  - In den sichersten MLS-Betriebssystemen ist es mathematisch bewiesen worden, dass Fred nur das kann was er darf
    - Meistens für die US National Security Agency (NSA) entwickelt
    - Sehr aufwendig

Erlaubnisse

- Was heißt "darf"? Darf was?
  - Eine Operation ausführen
  - Eine Datei lesen
  - Eine Datei schreiben oder löschen

MLS Beispiele

- Multics ist nach einem MLS-Modell designt.
- Unix ist von Bell Labs (Kernighan und Richie) zum Spaß entwickelt worden, als Bell Labs aus dem Multics-Consortium ausgestiegen ist.

Unix

- Drei Klassen von Permissions
  - Admin ("root") darf alles machen, alles lesen, alles schreiben
  - Gruppen-Rechte für Mitglieder einer Benutzergruppe
  - Einzelrechte für Benutzer
  - "Welt"rechte

Sicherheitsprobleme: Ein Beispiel

- Root darf alles
  - Also darf root die Passwort-Datei schreiben
  - Andere Benutzer dürfen die Passwort-Datei nicht schreiben
  - Alle Benutzer dürfen die Passwort-Datei lesen
    - Benutzer identifizieren sich bei der Ausführung des "login"-Programms. Login muss die Passwort-Datei lesen können, um den Benutzer identifizieren zu können
    - Die Passwörter in der Datei müssen verschlüsselt werden; login verschlüsselt das eingegebene Passwort zuerst; danach vergleicht es die beiden Verschlüsselungen
    - So far so good

Ein Beispiel

- Benutzer darf die Passwort-Datei nicht schreiben
- Aber man muss die Möglichkeit haben, das Passwort ändern zu können (vielleicht hat ein Anderer das Passwort beim Eintippen gesehen)
- Darf man aber selbst nicht
- Das Kommando muss seine Rechte "höher" setzen, das neue (verschlüsselte) Passwort in die Passwort-Datei hineinschreiben, und die Rechte wieder "niedriger" setzen

Ein Beispiel

- Es gibt einen Befehl, der die Rechte umsetzt
- Er heißt "`setuid()`"
- Also "`setuid(root)`“ wird ausgeführt, die PW-Datei geschrieben und "`setuid(Benutzer)`" gemacht
- Problem: Was passiert, wenn es der ausführende Benutzer schafft, dass das Programm nach dem "`setuid(root)`" Einzelkommandos von der Tastatur annimmt? Er hat "root permission". Er kann alles (meistens ist es ein “ER”!)

Ein Beispiel

- Also kann die Interaktion zwischen Umgebung und Einzelkommandos zu solchen Sicherheitslücken führen
- Unix ist nicht Multi-level Secure!
- Für die Geschichte und die Entwicklung der Ideen von Sicherheit (u.a. MLS), siehe
  - Donald MacKenzie, Mechanizing Proof, MIT Press 2001

Umgebungsprobleme

- Außerdem gibt es "Covert Channels"
  - Ein Benutzer hat Zugriff auf Dinge, von denen er auf Informationen rückschliessen kann, die er nicht bekommen darf
  - Beispiel:
    - Es sollte vertraulich bleiben, dass bestimmte Benutzerklassen überhaupt existieren
    - Fred glaubt, er ist in der höhsten Klasse außer Admin
    - Er bemerkt, dass Admin nicht eingelogt ist
    - Er bemerkt Aktivitäten, die er nicht weiter bestimmen kann
    - Diese liegen in einer höheren Klasse als er; daher weiß er jetzt, dass es Benutzer gibt, die in einer höheren Klasse als er, aber nicht Admin sind

Probleme so weit

- Kommandos
  - Probleme mit
    - Ausführung
    - Design
  - Falsches Vertrauen in Benutzer
- Umgebung
  - MLS
  - Ungünstige Interaktion zwischen MLS und Einzelkommandos
  - Covert Channels

Weitere Probleme

- Probleme liegen auch bei den Netzprotokollen
  - Bestätigungen bzw. ausgeführte Services auf Vertrauensbasis behandelt
- Zwei "Exploit"-Arten
  - "Remote Exploits" benutzen die Protokollsschwäche, um an den Rechner heranzukommen (Benutzerrechte auszuüben)
  - "Local Exploits" sind identisch zu den für einen berechtigten Benutzer beschriebenen

Local Exploits

- Man bekommt zuerst Benutzername+Passwort
- Man logt sich mit diesen beiden ein
- Man macht Mist (schreibt Files, löscht Files, versucht root-Rechte zu bekommen)

Local Exploits

- Wie bekommt man BN+PW?
  - Man untersucht Protokollverkehr auf dem Kabel bzw. in einem Rechner, der die Pakete überträgt
  - Bei bestimmten Services werden BN+PN in Klartext übertragen (telnet, ftp, pop, imap)
  - z.B. schließt man seinen Laptop an ein Ethernet an und ließt die IP-Pakete, die hin und her geschickt werden
    - Gegenmaßnahme 1: Man kann es nicht., z.B. Switched Verkehr, der nur zwischen Quelle und Ziel läuft
    - Gegenmaßnahme 2: Man erlaubt nur vertraute Benutzer an dem physikalischen Netz
    - Gegenmaßnahme 3: Man verschlüsselt den Verkehr

Local Exploits

- Probleme
  - Gegenmaßnahme 1: Wenn man einen Switch mit vielen Paketen gleichzeitig überlastet, macht der Switch Broadcasting, bis der Verkehr wieder vernünftig ist
  - Gegenmaßnahme 2: die Putzleute haben meistens auch Schlüssel, oder auch Freunde
  - Gegenmaßnahme 3: Die Verschlüsselung muss sicher sein, z.B. bei einem FunkLAN nach IEEE 802.11 ist die Verschlüsselung nicht hinreichend

Die beste Maßnahme

- Ständige oder frequente Überwachung
  - Logfiles von Service-Benutzungen
  - Unpassende Einträge bemerken und verfolgen
    - Beobachten, wenn verdächtige Operationen ausgeführt werden
    - Services beenden, wenn man weiß, was gemacht worden ist und deswegen weiß, was der Cracker am System geändert hat
    - Aufräumen oder das System wieder vom Backup restaurieren und Benutzer auffordern, BN+PW zu ändern
- Mit Passwort-Verfahren endlich aufhören!

Stichwörter

- Sicherheit ist relativ
  - Ziel: besser als die Nachbarn zu sein und daher weniger attraktiv für Angriffe
- Sicherheit ist nur relativ
  - Außerhalb des NSA gibt es keinen wirklich sicheren Rechner
- Überwachung ist wesentlich