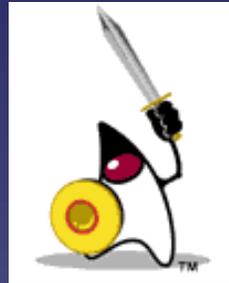


Java-Security

Teil II



Vorlesung Firewalls und andere IT-Sicherheitsmechanismen

cand.Ing André Große Bley

7. Februar 2003

Übersicht

- Signierte Klassen
- Policies
- Sicherheitsprobleme
- Anwendungsgebiete
- Ausblick

Signierte Klassen

- JAR-Dateien können signiert werden
- JAR-Dateien enthalten Klassen und Anwendungsdaten
- Signierung nach ITU-T X.509 Empfehlung
- Als de facto Standard von SSL/TLS bekannt
- Einfache Handhabung für den Endnutzer durch Einsatz einer CA
- Nutzer kann prüfen, ob das Archiv nach dem Signieren manipuliert wurde
- Tools zur Zertifikat- und Signaturverwaltung werden mitgeliefert

Probleme signierter Klassen

- Signatur recht sicher auf verwendetes Zertifikat zurückverfolgbar
- CA soll Identität des Inhabers des Zertifikates prüfen
- Mögliche Probleme:
 - ★ Gestohlene Zertifikate
 - ★ Nicht vertrauenswürdige CA
 - ★ Zweifel an der Integrität der CA
 - ★ Kosten für Zertifizierung durch ROOT-CA
- Klasse wird von niemanden auf Sicherheit überprüft! \Rightarrow „böse“ Klassen können signiert sein
- (Java 1.1) Signierte Klassen bekommen meist ohne Nachfrage volle Zugriffsrechte (wie lokale Klassen)

Policies

- Java2 (ab Java 1.2) erlaubt feinere Einstellung der Zugriffsrechte durch Policies:
 - ★ Wahl der Rechte anhand der Codebase oder des Signierers
 - ★ Beschränkung der Dateizugriffe auf bestimmte Pfade
- *AccessController* kann auch für lokale Klassen geladen werden
- Eigener *SecurityManager* nur noch selten nötig
- Policies werden aus Konfigurationsdateien gelesen, *policytool* hilft beim Erstellen dieser
- ABER: Browser mit integriertem Java benutzen eigene Sicherheitsklassen ⇒ andere Einstellmöglichkeiten

Konfiguration

- *JAVAROOT/jre/lib/security/java.security* ist *master security properties file*
 - ★ Definiert *SecurityManager*- und *AccessController*-Klassen
 - ★ Erlaubt mehrere SM und AC
 - ★ Legt den Suchpfad für Policies fest
- *JAVAROOT/jre/lib/security/java.policy* ist *standard policy file*
 - ★ Festlegung der grundlegenden Rechte
 - ★ Gewähren höherer Rechte für bestimmte Klassen
 - ★ Allerdings komplizierte Konfiguration durch viele verschiedene Rechte

Sicherheitsprobleme

- Bistlang keine Designfehler im Sicherheitskonzept bekannt
- Verschiedene Fehler in der Implementierung aufgetreten, die schnell behoben wurden
- Sämtliche Lücken werden veröffentlicht, zum Teil mit Test-Applets (Sicherheit durch Offenheit)
- Quellcode ist verfügbar \Rightarrow Möglichkeit der eigenen Sicherheitsprüfung
- Denial-of-Service durch Applets über Ressourcenverbrauch möglich
- Sicherheit als ein primäres Entwicklungsziel sorgt für bedeutend weniger Sicherheitsprobleme verglichen mit anderen Produkten

Viren

Es gibt zwei bekannte Java-Viren:

- StrangeBrew
 - ★ Versucht andere Java-Klassen (nur .class) zu infizieren
 - ★ In Browsern ohne vom Nutzer geänderte Sicherheitseinstellungen nicht gefährlich
 - ★ Keine nennenswerte Verbreitung (weniger als 49 bekannte Infektionen laut Symantec)
- BeanHive
 - ★ Fordert höhere Zugriffsrechte an, kann nur aktiv werden, wenn Nutzer zustimmt
 - ★ Demonstrationvirus mit ebenfalls nicht nennenswerter Verbreitung

Anwendungen

- Klassische Anwendungen, die Sun sah
 - ★ Interaktive Webseiten auf *Client*basis (Applets)
 - ★ Portable Geräte
 - ★ Embedded Systems
 - ★ Ersatz des Arbeitsplatz-PCs durch Java-Stations
 - ★ Plattformunabhängige Anwendungen
- Aktuelle Anwendungen
 - ★ Mobile Geräte (PDAs, Mobiltelefone)
 - ★ Native Applikationen
 - ★ Interaktive Webseiten auf *Server*basis (Servlets)

Ausblick

- Java ist nicht absolut sicher, allerdings viel sicherer als Konkurrenzprodukte
- Sicherheit und Portabilität sorgt für Beliebtheit auf Serverseite
- Mobile javataugliche Geräte gewinnen an Verbreitung
- Sicherheit z.B. bei Mobiltelefonen wichtig

Vielen Dank für Ihre Aufmerksamkeit.

Fragen?

`grossagk@rub.de`

PGP-Key ID 0xDFB7CB00

<http://scherz.keks.net/uni/>