Authentisierung und Security: Aktuelle Entwicklungen

Dipl.-Chem. Rainer Orth
Technische Fakultät
Universität Bielefeld
ro@TechFak.Uni-Bielefeld.DE

Übersicht

- PAM
- GSS-API und RPCSEC_GSS
- SASL
- Diverses: SACRED, KINK
- Nicht: IPsec, Kerberos, MSec, OpenPGP, PANA, PKIX, SecSH, S/MIME, TLS, ...
- Zusammenfassung

PAM: Pluggable Authentication Modules

- Probleme
  - verschiedene Verfahren zur Authentisierung lokaler Benutzer
  - mehrere Applikationen mit dieser Aufgabe
  - fehlende Unterstützung einzelner Verfahren inbes. ohne Sourcecode
  - unflexible/hardcodierte Policies
- Lösung: konfigurierbares Authentisierungsframework mit stackbaren Modulen

PAM: Spezifikation, Implementierungen und Nutzer

- erste Spezifikation: Samar, Schemers (Sun), OSF RFC 86.0, 1995
- jetzt auch: Open Group X/Open Single Sign-On Service (XSSO), 1997
- Implementierungen
  - Sun/Solaris (seit 2.3, dort noch anderes Interface, seit 2.6 public)
  - HP-UX 11, AIX 5.2
  - Linux-PAM: diverse Linux-Distributionen
  - OpenPAM: FreeBSD
- Nutzer in Solaris 9: login, passwd, su, rlogind, rshd, telnetd, ftpd, rpc.rexd, uucpd, init, sac, cron, ppp, dtsession, dtlogin, ssh, ttymon

PAM: Aufgaben und Interfaces

- Modul-Aufgaben
  - Authentication (`auth`)
  - Account Management (`account`)
  - Session Management (`session`)
  - Password Management (`password`)
- Modul-Stack: `required, optional, sufficient`
- Interfaces:
  - PAM-API für Applikationen
  - PAM-SPI (Service Provider Interface) für Module

PAM: Konfiguration mit pam.conf

```
# service_name module_type control_flag module_path     options
rlogin         auth        sufficient   pam_rhosts_auth.so.1
rlogin         auth        required     pam_unix.so.1
rlogin         auth        optional     pam_krb5.so.1   try_first_pass
#
other          account     required     pam_unix.so.1
other          account     optional     pam_krb5.so.1
#
other          session     required     pam_unix.so.1
other          session     optional     pam_krb5.so.1
#
other          password    required     pam_unix.so.1
other          password    optional     pam_krb5.so.1   try_first_pass
```

PAM-Module

- normalerweise in `/usr/lib/security/$ISA`
- Beispiele: `pam_krb5`, `pam_ldap`, `pam_unix`, `pam_smartcard`, `pam_opie`
- zahlreiche Module werden mit Linux-PAM verteilt oder setzen darauf auf: `http://www.kernel.org/pub/linux/libs/pam/modules.html`

GSS-API: nicht nur ein API

- Generic Security Service Application Program Interface
- allgemeines API für Security-Dienste in Netzwerk-Protokollen
- unabhängig von den zugrundeliegenden Mechanismen
- enthält keine Abhängigkeiten von Netzwerk-Protokollen
- GSS-API liefert dem Aufrufer Token, die er in protokoll-spezifischer Weise versenden muß
- erste Spezifikation: GSS-API V1, RFC 1508, John Linn, 1993
- aktuell: GSS-API V2 Update 1, RFC 2743, 2000

GSS-API: Bindings, Mechanismen und Implementierungen

- C- und Java-Bindings: RFC 2744, 2853 (2000)
- Mechanismen
  - Kerberos V5 (RFC 1964, 1996)
  - Simple Public-Key (SPKM) (RFC 2025, 1996)
  - Low Infrastructure Public Key (LIPKEY) (RFC 2847, 2000)
- Implementierungen
  - MIT Kerberos V5, Heimdal: Kerberos V5
  - Solaris 8/9: Kerberos V5, Diffie-Hellman
  - JDK 1.4: Kerberos V5
  - Globus Toolkit: SSL/TLS
  - GSS (GNU): Kerberos V5

GSS-API: Interfaces und Nutzer

- Context-Erzeugung: `GSS_Init_sec_context`, `GSS_Accept_sec_context`
- Per-Message-Funktionen: `GSS_GetMIC`, `GSS_VerifyMIC`, `GSS_Wrap`, `GSS_Unwrap`
- Hilfsfunktionen: Status-Abfragen, Namensbehandlung, ...
- Nutzer: EAP, SSH, SASL, TSIG (DNS), IKE (ISAKMP), RPCSEC_GSS, ...

ONC-RPC-Security mit RPCSEC_GSS

- RPC-Security bisher: keine: `AUTH_NONE, AUTH_SYS`
- . . . oder kaum implementiert: `AUTH_DH, AUTH_KERB`
- bieten höchstens Authentisierung
- jetzt: `RPCSEC_GSS`, ein Security-Flavor für alle GSS-API-Mechanismen (RFC 2203, 1997)
- bietet auch Message Integrity und Verschlüsselung
- z.B. zwingend vorgeschrieben für NFS v4

SASL: Simple Authentication and Security Layer

- Authentisierungs- und (optional) Security-Framework für verbindungsorientierte (textbasierte) Protokolle
- statt jedes Authentisierungsverfahren mit jedem Protokoll zu verbinden: je ein SASL-Profil pro Protokoll und eine SASL-Spezifikation pro Mechanismus
- Unterscheidung zw. Authentication ID und Authorization ID für Proxy-Authentisierung

SASL: Spezifikationen und Nutzer

- Spezifikation: RFC 2222 (J. Myers, Netscape, 1997)
- Mechanismen
  - ANONYMOUS: RFC 2245 (1997)
  - CRAM-MD5: RFC 2195 (1997)
  - DIGEST-MD5: RFC 2831 (2000)
  - EXTERNAL: RFC 2222 (1997)
  - GSSAPI: RFC 2222 (1997)
  - OTP: RFC 2444 (1998)
  - PLAIN: RFC 2595 (1999)
- Nutzer: BEEP, IMAP, ACAP, LDAP, POP, SMTP, ggfs. HTTP

SASL: Beispiel mit IMAP

```
S: * OK imap.TechFak.Uni-Bielefeld.DE Cyrus IMAP4 v2.1.12 server ready
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA ...
   STARTTLS LOGINDISABLED AUTH=GSSAPI
S: C01 OK Completed
C: A01 AUTHENTICATE GSSAPI
S: +
C: YIICHgYJKoZIhvcSAQICAQBuggINMIICCaADAgEFoQMCAQ6iBwMFACAAAACjggFBYYIBP...
S: + YGgGCSqGSIb3EgECAgIAb1kwV6ADAgEFoQMCAQ+iSzBJoAMCAQGiQgRAiKeAys2uhqv...
C:
S: + YDMGCSqGSIb3EgECAgIBAAD/////vGH6MSnsyvVE+vpJI+gJw0d7nZw4ftFJBAAQAAQEBAQ=
C: YDMGCSqGSIb3EgECAgIBAAD/////YPcESBF05yRn5yqed1hg56BbgHErzFVGBAAEAAQEBAQ=
S: A01 OK Success (privacy protection)
```

SASL: Beispiel 2 mit IMAP und STARTTLS

```
S: * OK imap.TechFak.Uni-Bielefeld.DE Cyrus IMAP4 v2.1.12 server ready
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA ...
   STARTTLS LOGINDISABLED AUTH=GSSAPI
S: C01 OK Completed
C: S01 STARTTLS
S: S01 OK Begin TLS negotiation now
TLS connection established: TLSv1 with cipher DES-CBC3-SHA (168/168 bits)
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA ...
   STARTTLS AUTH=PLAIN AUTH=GSSAPI
S: C01 OK Completed
C: A01 AUTHENTICATE PLAIN
S: +
Please enter your password:
C: cm8Acm8AR2xkVyt3c1M=
S: A01 OK Success (tls protection)
```

SASL-Implementierungen

- C- und Java-APIs als Internet-Drafts
- Implementierungen
  - Cyrus SASL 2.1.13
  - GNU SASL 0.0.7
  - Cryptix 0.8.8 (Java)

KINK: Kerberized Internet Negotiation of Keys

- Alternative zu IKE (ISAKMP) bei Authentisierung und Key Exchange für IPsec
- Probleme mit IKE (Internet Key Exchange)
  - erfordert manuelle Key-Verteilung oder PKI
  - Public-Key-Kryptographie und DH-Key Exchange CPU-intensiv
  - verteilte Policy-Kontrolle auf den Endsystemen

KINK: Lösung und Status

- daher: KINK: Kerberos V5 für Authentisierung (entweder secret key oder PKinit) und Key-Verteilung
- zentralisiertes Management, Secret-Key-Kryptographie weniger CPU-intensiv
- Nachteil: nicht global einsetzbar (d.h. ohne trusted third party (KDC))
- bisher: Requirements-RFC (RFC 3129, 2001) und Draft-Spezifikation (basiert auf ISAKMP-Packetformat)

SACRED: Securely Available Credentials

- Problem: Nutzung von Credentials von verschiedenen Systemen aus: Workstation, Laptop, PDA, Handy, ...
- werden benötig für z.B. S/MIME, PGP, TLS, ...
- Schwierigkeiten mit Transport per Floppy, PCMCIA-Karten, ...
- Alternative zu Smart Cards
- zwei Modelle: Client—Credential Server, Peer-to-Peer
- bisher: Requirements-RFC (RFC 3157, 2001), Internet-Drafts für Server-Framework, Protokoll

Zusammenfassung

- PAM schon heute nutzbar, aber nicht allgegenwärtig und Modul-Qualität sehr unterschiedlich, betriebssystem-abhängig
- GSS-API direkt nur für Spezialanwendungen (z.B. UDP-basiert)
- SASL für die meisten Netzwerk-Protokolle