CNS
Lecture 13

Network defenses

IPsec

Virtual Private Networks (VPNs)

Wireless security

Kerberos

Trusted systems

Secure OS

In the news

Microsoft workstation server buffer overflow

Microsoft XML core services remote code execution

Microsoft agent buffer overflow

WinZip remote code execution

You are here …

Attacks & Defenses
- Risk assessment✓
- Viruses✓
- Unix security✓
- authentication✓
- Network security
  Firewalls,vpn,IPsec,IDS
- Forensics✓

Cryptography
- •Random numbers✓
- •Hash functions✓
  MD5, SHA,RIPEMD
- •Classical + stego✓
- •Number theory✓
- •Symmetric key✓
  DES, Rijndael, RC5
- •Public key✓
  RSA, DSA, D-H,ECC

Applied crypto
- •SSH ✓
- •PGP ✓
- •S/Mime ✓
- •SSL ✓
- •Kerberos
- •IPsec
- •Crypto APIs
- •Secure coding

Network security

VULNERABILITIES
- denial of service
  - ICMP smurf, redirects, unreachable
  - SYN flooding
  - frag, teardrop
- impersonation
  - host rename (LAN)
  - DNS
  - source routing
- Session capture/modification
  - TCP seq number guessing
  - TCP hijacking
  - sniffing
- Server/application attacks
  - application flooding (ftp,mail,echo)
  - buffer overflows
  - Software bugs

COUNTERMEASURES
- disable
- configure properly
- xinetd, tcpwrappers
  - filters (allow, deny)
  - audit and alarm
- filtering portmap
- application filtering (securelib)
- patches
- scanners (Nessus, ISS)
- firewalls
- intusion detection & response
- encryption, virtual private networks (VPNs)

Where to encrypt?

link layer
- •encrypting modem, NIC (wireless)
- •transparent,fast
- •suitable for private net
- •protects only one link (pt-to-pt)
- •info may be exposed in OS

network/transport layer
- •swIPe, IPv6(IPsec)
- •transparent
- •selectable (policy)
- •appl./host/net keying
- •works over public net
- •virtual private network (VPN)
- •system layer: encrypting file systems (EFS/CFS)

application layer
- •end-to-end over public net
- •custom applications (PGP, ssh, ssl)
- •intrusive, but flexible
- •API for application development

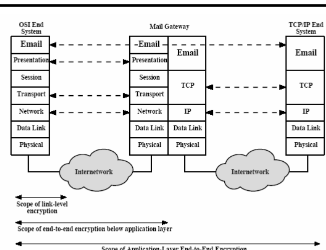


Figure 7.4 Encryption Coverage Implications of Store-and-Forward Communications

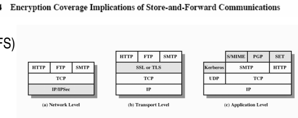


Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

Internet protocol (IP)

Is IP secure?

- integrity -- checksums?
- can you trust the source address?
- privacy?
- IP security option (RFC 760)
  - –military security model: subject/object labels
  - –label each IP datagram (secret, top secret, unclassified)
  - –IP stack and routers enforce access controls
  - –only effective in very controlled environment

IP header

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  OPTIONAL          Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv4 Header Format

```
Protocols:   1 ICMP       6 TCP     17 UDP       50 ESP   51 AH
```


IP security option

RFC 760

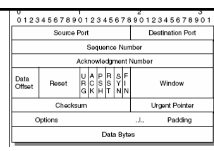
This option provides a way for DoD hosts to send security and TCC (closed user groups) parameters through networks whose transport leader does not contain fields for this information.

```
+--------+--------+---------+--------+
|00000010|00000100|000000SS |  TCC   |
+--------+--------+---------+--------+
Type=2  Length=4

Security:  2 bits

Specifies one of 4 levels of security
11-top secret
10-secret
01-confidential
00-unclassified

Transmission Control Code:  8 bits
```

Provides a means to compartmentalize traffic and define controlled communities of interest among subscribers.

But the IP header is easliy forged!

TCP keyed-MD5 option



- RFC 2385 (BGP, LDP, MSDP)
  - Authenticate routing protocols
- Include a keyed-MD5 checksum in TCP option field
  - Each TCP segment carries keyed checksum
  - Implemented in kernel
- Limited deployment: Cisco/Juniper routers, FreeBSD/OpenBSD
- setsockopt() to enable and set key TCP_MD5SIG
- Probably better to use HMAC as part of "application packet" or use secure transport (SSL, IPsec)

IPsec – new IP security headers

- RFC's for IPsec (v4 and v6)
- specifies implementation
- authenticated packets
  - prevents spoofed source addresses
- encrypted packets (transport or tunnel)
  - prevents sniffing
- does not specify policy
- now includes key management
- could use on a host
- could use on a router (tunnels)

IETF IPsec RFC's

RFC2406 ESP
RFC2402 AH
RFC2104 HMAC
RFC2412 Oakley
RFC2408 ISAKMP
RFC2409 IKE
RFC3715 IPsec and NAT

IPsec

- IP protos 50 and 51 for IPv4
- authenticated (51) and/or encrypted (50) datagrams
- system manager sets "policy"
- no changes to application (or optional)
- key management (IKE)
- can tunnel IP datagrams (VPN)
- implementations available
- US export controls limiting

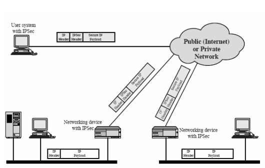


Figure 16.1 An IP Security Scenario

IPsec services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality (encryption)

Table 16.1 IPSec Services

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

Encryption without authentication is useless

IPsec protocol

- Establish a security association in each direction
  - negotiate parameters/algorithms
  - establish a secret (session key)
  - authenticate
  - not unlike ssh/ssl
- keyed hashes (md5/sha/tiger) HMAC
- public keys (RSA/DSA) or pre-shared secret
- block encryption
  - AES/DES/3DES/blowfish/CAST/IDEA/RC5
- Diffie-Hellman (mod p or ECC)
- tunnel and transport mode
- Requires modifications to OS!

Security Association (SA)

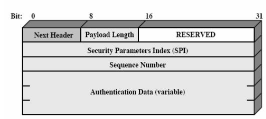
- sender/receiver security info
- SA for each direction
- maintained by kernel
- identify by SPI (handle) and destination
- specifies
  - encryption key, IV, algorithm (DES, 3DES,CAST, Blowfish,AES)
  - authentication algorithm (MD5, SHA)
  - key lifetimes
  - SA lifetime
  - security labels

SA

```
/* Security association data for IP Security */
struct key_secassoc {
u_int8  len;                /* Length of the data (for radix) */
u_int8  type;               /* Type of association */
u_int8  state;              /* State of the association */
u_int8  label;              /* Sensitivity label (unused) */
u_int32 spi;                /* SPI */
u_int8  keylen;             /* Key length */
u_int8  ivlen;              /* Initialization vector length */
u_int8  algorithm;          /* Algorithm switch index */
u_int8  lifetype;           /* Type of lifetime */
caddr_t iv;                 /* Initialization vector */
caddr_t key;                /* Key */
u_int32 lifetime1;          /* Lifetime value 1 */
u_int32 lifetime2;          /* Lifetime value 2 */
SOCKADDR *src;              /* Source host address */
SOCKADDR *dst;              /* Destination host address */
SOCKADDR *from;             /* Originator of association */
u_int32 tp_len;             /* Transform private data: length */
void *tp_data;              /* Transform private data: data   */
};
```


authentication

- negotiate
- MD5/SHA HMAC (keyed)
- calculated over non-changing fields of IP packet
  - headers (IP and TCP/ UDP) and user data
  - NAT causes problems (IP source address changes)
    dilemma: using source IP address as security token, but hosts have multiple and changing IP addresses today
- IP proto 51 RFC 2402 AH packet



AH header

```
+------------+-------------------+------------+-------+--------------+
| IPv6 Header| Hop-by-Hop/Routing| Auth Header| Others| Upper Protocol|
+------------+-------------------+------------+-------+--------------+

+-------------+--------------+-----------------------------+
| IPv4 Header |  Auth Header | Upper Protocol (e.g. TCP, UDP)|
+-------------+--------------+-----------------------------+

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Payload Len  |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number Field                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                Authentication Data (variable)                 |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next header: e.g. TCP

SPI – index into SA table

Sequence # -- prevent replay

Authentication -- HMAC

Transport and tunnel modes

Tunnel mode has additional header that can specify different target (e.g. firewall or VPN)

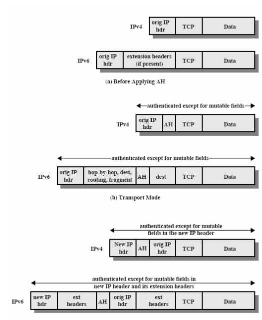
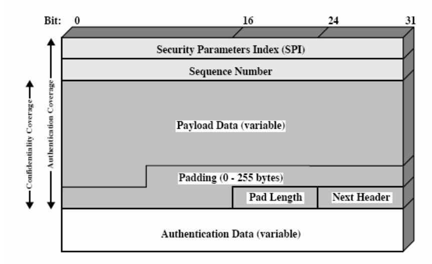


Figure 16.6 Scope of AH Authentication

IPsec encryption

- IP proto 50   RFC 2406   ESP



ESP header

```
|<--        Unencrypted             -->|<----    Encrypted   ------>|
+-------------+--------------------+------------+---------------------+
| IP Header   | Other IP Headers   | ESP Header | encrypted data      |
+-------------+--------------------+------------+---------------------+

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ----
|               Security Parameters Index (SPI)                 | ^Auth.
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Coverage
|                      Sequence Number                          | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | ----
|                    Payload Data* (variable)                   | |   ^
~                                                               ~ |   |
|                                                               | |Conf.
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Coverage
|               |     Padding (0-255 bytes)                     | |   |
+-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
|                               |  Pad Length   | Next Header   | |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
|                 Authentication Data (variable)                | |   |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


ESP transport and tunnel modes

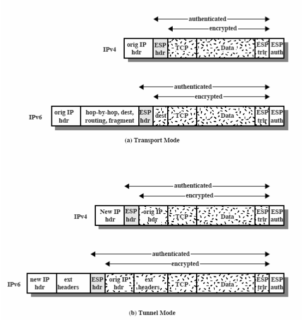
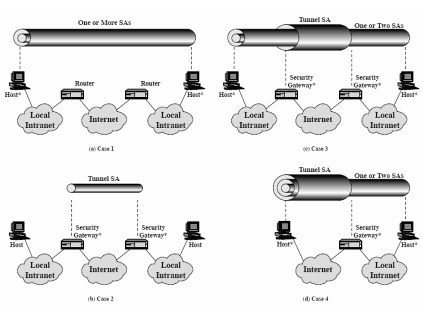


Figure 16.9 Scope of ESP Encryption and Authentication

IPsec processing

- use SPI from packet to look up SA
- authenticate with info from SA
- decrypt with info from SA
- transport mode protects payload (host based)
- tunnel mode protects entire packet
  - thwart some traffic analysis
  - used by VPNs (router/firewall)

Tunnel options



(a) End systems
(b) Routers/VPN
(c) a + b
(d) VPN +

IPsec implementations

- many OS vendors have implementations
- also shrink-wrapped VPN solutions
- export is problem
- NAT may be a problem
- freeware OS patches
- policy: all (transparent to applications) or application request
- block connect() til SA established

NRL's API (standard ?)

```
after socket()

setsockopt(fd, SOL_SOCKET, SO_SECURITY_AUTHENTICATION, &auth,
  len = sizeof(int))

setsockopt(fd, SOL_SOCKET, SO_SECURITY_ENCRYPTION_TRANSPORT,
  &esptrans, len = sizeof(int))
setsockopt(fd, SOL_SOCKET, SO_SECURITY_ENCRYPTION_NETWORK,
  &esptrans, len = sizeof(int))
```

IPsec key management

Internet Key Exchange (IKE)
- manual keying or automatic key establishment
- proposals (SKIP, ISAKMP, Photuris)
- SKIP
  - –light-weight
  - –in-band
  - –Diffie-Hellman (signed public keys)
- IKE (ISAKMP/Oakley) RFC2409
  - –out-of band, daemon (UDP port 500)
  - –negotiate (Oakley)
  - –Diffie-Hellman, public keys

IKE

Oakley RFC2412

- **key exchange protocol**
  - –speed vs more secure
  - –id vs anonymity
  - –new vs re-key
- **Diffie-Hellman with authentication (5 groups: 3 mod exp, 2 ECC) authenticate with signature, or public key encryption, or pre-shared secret**
- **cookies to thwart DoS -- MD5(IP src, IP dst, ports, mysecret)**
- **nonces to thwart replay (pseudo-random number)**
- **Simple encoding of BIG integers (32- bit length, and n 32-bit words, MSWF)**
- **provides perfect forward secrecy (PFS)**
  - –compromise of a single key will permit access to only
  - –data protected by a single key
  - – key used to protect transmission of data MUST NOT be used to derive any additional keys

Aggressive Oakley key exchange

I → R: CKY_I, OK_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, $S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

R → I: CKY_R, CKY_I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, $S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

I → R: CKY_I, CKY_R, OK_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, $S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

Notation:

I	=	Initiator
R	=	Responder
CKY_I, CKY_R	=	Initiator, responder cookies
OK_KEYX	=	Key exchange message type
GRP	=	Name of Diffie-Hellman group for this exchange
g^x, g^y	=	Public key of initiator, responder; g^{xy} = session key from this exchange
EHAO, EHAS	=	Encryption, hash, authentication functions, offered and selected
NIDP	=	Indicates encryption is not used for remainder of this message
ID_I, ID_R	=	Identifier for initiator, responder
N_I, N_R	=	Random nonce supplied by initiator, responder for this exchange
$S_{KI}[X]$, $S_{KR}[X]$	=	Indicates the signature over X using the private key (signing key) of initiator, responder

The D-H calculation is CPU-intensive. The cookies provide anti-clogging, so Mallory can't send random D-H values and make Bob do D-H calculations. Above, I does D-H after step 2, R does D-H after final step.

IKE – ISAKMP RFC 2408

- **protocols and packet formats to establish, modfiy, and delete security associations**
  - –application on Alice wants to communicate with application on Bob, kernel policy says an SA is needed, Alice's ISAKMP daemon is notified to get an SA with Bob
  - –Negotiate algorithms, key sizes, type of association
- **phase I -- two ISAKMP peers (daemons) establish a security association (SA)**
- **phase II -- negotiate and establish SA for requesting application (IPsec)**
- **packet formats are chain of payloads**
- IKE is Oakley plus ISAKMP

ISAKMP formats

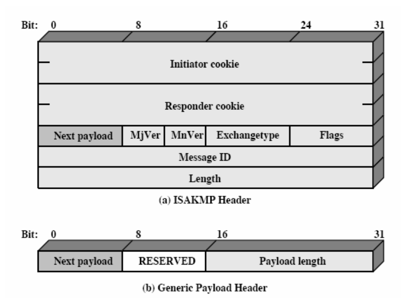


Figure 16.12 ISAKMP Formats

ISAKMP payloads

Type	Parameters	Description
Security Association (SA)	Domain of Interpretation, Situation	Used to negotiate security attributes and indicate the DOI and Situation under which negotiation is taking place.
Proposal (P)	Proposal #, Protocol-ID, SPI Size, # of Transforms, SPI	Used during SA negotiation; indicates protocol to be used and number of transforms.
Transform (T)	Transform #, Transform-ID, SA Attributes	Used during SA negotiation; indicates transform and related SA attributes.
Key Exchange (KE)	Key Exchange Data	Supports a variety of key exchange techniques.
Identification (ID)	ID Type, ID Data	Used to exchange identification information.
Certificate (CERT)	Cert Encoding, Certificate Data	Used to transport certificates and other certificate-related information.
Certificate Request (CR)	# Cert Types, Certificate Types, # Cert Auths, Certificate Authorities	Used to request certificates; indicates the types of certificates requested and the acceptable certificate authorities.
Hash (HASH)	Hash Data	Contains data generated by a hash function.
Signature (SIG)	Signature Data	Contains data generated by a digital signa wture function.
Nonce (NONCE)	Nonce Data	Contains a nonce.
Notification (N)	DOI, Protocol-ID, SPI Size, Notify Message Type, SPI, Notification Data	Used to transmit notification data, such as an error condition.
Delete (D)	DOI, Protocol-ID, SPI Size, # of SPIs, SPI (one or more)	Indicates an SA that is no longer valid.

ISAKMP exchanges

Table 16.4 ISAKMP Exchange Types

Exchange	Note
(a) Base Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE	Basic SA agreed upon
(3) **I → R:** KE; ID_I; AUTH	Key generated; Initiator identity verified by responder
(4) **R → I:** KE; ID_R; AUTH	Responder identity verified by initiator; Key generated; SA established
(b) Identity Protection Exchange	
(1) **I → R:** SA	Begin ISAKMP-SA negotiation
(2) **R → I:** SA	Basic SA agreed upon
(3) **I → R:** KE; NONCE	Key generated
(4) **R → I:** KE; NONCE	Key generated
(5)* **I → R:** ID_I; AUTH	Initiator identity verified by responder
(6)* **R → I:** ID_R; AUTH	Responder identity verified by initiator; SA established
(c) Authentication Only Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE; ID_R; AUTH	Basic SA agreed upon; Responder identity verified by initiator
(3) **I → R:** ID_I; AUTH	Initiator identity verified by responder; SA established
(d) Aggressive Exchange	
(1) **I → R:** SA; KE; NONCE; ID_I	Begin ISAKMP-SA negotiation and key exchange
(2) **R → I:** SA; KE; NONCE; ID_R; AUTH	Initiator identity verified by responder; Key generated; Basic SA agreed upon
(3)* **I → R:** AUTH	Responder identity verified by initiator; SA established
(e) Informational Exchange	
(1)* **I → R:** N/D	Error or status notification, or deletion

Notation:
I = initiator
R = responder
* = signifies payload encryption after the ISAKMP header

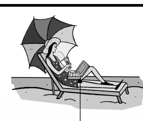
The IPsec dance on the wire (tcpdump)

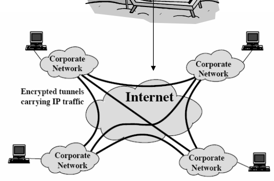
- VPN client requesting secure tunnel (w/ shared secret)
- IKE on UDP port 500
- Encrypted IPsec, IP proto 50

```
06:52:00.852251 205.138.57.223.1147 > 192.31.96.188.500: udp 300
06:52:02.791434 192.31.96.188.500 > 205.138.57.223.1147: udp 244
06:52:02.791513 192.31.96.188.500 > 205.138.57.223.1147: udp 244
06:52:05.081347 205.138.57.223.1147 > 192.31.96.188.500: udp 48
06:52:05.083159 192.31.96.188.500 > 205.138.57.223.1147: udp 404
06:52:05.083250 192.31.96.188.500 > 205.138.57.223.1147: udp 404
06:52:05.226289 205.138.57.223.1147 > 192.31.96.188.500: udp 180
06:52:05.237861 192.31.96.188.500 > 205.138.57.223.1147: udp 164
06:52:05.237941 192.31.96.188.500 > 205.138.57.223.1147: udp 164
06:53:40.553207 205.138.57.223 > 192.31.96.188: ip-proto-50 100
06:53:40.555380 192.31.96.188 > 205.138.57.223: ip-proto-50 84 (DF)
06:53:40.555452 192.31.96.188 > 205.138.57.223: ip-proto-50 84 (DF)
06:53:40.648533 205.138.57.223 > 192.31.96.188: ip-proto-50 76
06:53:40.679228 205.138.57.223 > 192.31.96.188: ip-proto-50 148
06:53:40.713540 192.31.96.188 > 205.138.57.223: ip-proto-50 76 (DF)
06:53:40.713605 192.31.96.188 > 205.138.57.223: ip-proto-50 76 (DF)
```

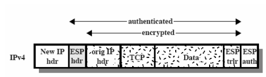

Virtual Private Networks

- Tunneling traffic over the Internet
  - Initially adhoc solutions (PPTP)
  - Most now based on IPsec and may be interoperable
- Alternative: leased lines (plus encryption) or remote dial-in
- construct encrypted tunnels over the Internet
- router-tunnels and standalone clients (roaming)
- use for internal privacy too
- clients for linux, mac, win*
- no changes to applications
- network address translation(NAT) makes client appear like its on local net





(b) A virtual private network via Tunnel Mode



VPNs

Components

- encrypting boxes/software
  - VPN enterprise routers
  - VPN client software
- maybe VPN server hardware encryption
- total isolation from public net, or firewalls too
- key server

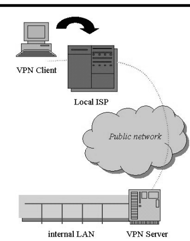
What about security of remote client?
home machine compromised and now its tunneled in ☹
you've bypassed enterprise IDS and firewall – need another IDS for VPN

Criteria

- platforms, interoperability
- proprietary or open (IPsec)
- ease of use
- strength of security
- performance (server bottleneck?)
- mobile user support
- ease of management (key mgt)
- network address translation (NAT)
  - Both client and host NAT support
- Scalability (active sessions)
- exportable
- Cost (free clients?)

VPN example – remote user

- Client connects to Internet
- Click on VPN icon
- Login
- Laptop appears as a new IP address on enterprise network
  - Can access internal file shares
  - Access internal-only web services etc.
  - No changes to applications



- Connection is made to enterprise VPN server, e.g. port 443
- IPsec SA established (encrypted tunnel)
- User logs in
- Routing tables on client adjusted
  enterprise traffic goes thru the tunnel

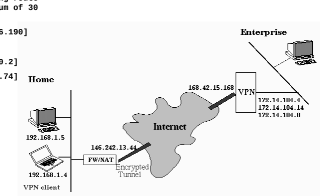
VPN example

Before

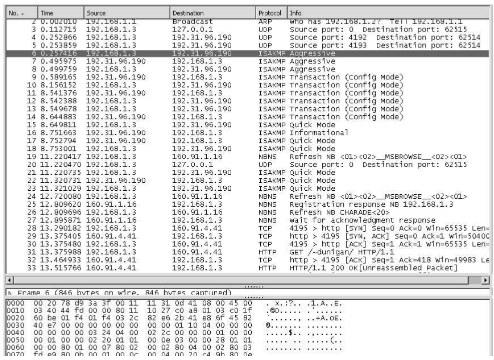
```
C:\WINNT\system32>tracert thistle.csm.ornl.gov
Tracing route to thistle.csm.ornl.gov [160.91.212.74] over a
  maximum of 30 hops:
 1 10 ms 20 ms 10 ms 10.118.32.1
 2 10 ms 20 ms 20 ms 172.30.34.17
 3 10 ms 20 ms 10 ms 172.30.34.58
 4 10 ms 10 ms 21 ms 68.47.160.50
 5 10 ms 20 ms 20 ms 12.118.120.101
 6 10 ms 31 ms 20 ms tbr2-p013901.attga.ip.att.net [12.123.21.142]
 7 30 ms 40 ms 40 ms 12.122.10.138
 8 40 ms 40 ms 50 ms tbr2-cl7.cgcil.ip.att.net [12.122.10.45]
 9 40 ms 40 ms 50 ms gbr2-p40.cgcil.ip.att.net [12.122.11.54]
10 40 ms 40 ms 40 ms gr1-p3110.cgcil.ip.att.net [12.123.5.5]
11 40 ms 41 ms 50 ms aads-att.es.net [198.124.216.21]
12 40 ms 40 ms 40 ms chicr1-ge0-chirt1.es.net [134.55.209.189]
13 90 ms 70 ms 60 ms aoacr1-oc192-chicr1.es.net [134.55.209.58]
14 70 ms 80 ms 70 ms dccr1-oc48-aoacr1.es.net [134.55.209.62]
15 80 ms 81 ms 80 ms atlcr1-oc48-dccr1.es.net [134.55.209.66]
16 90 ms 80 ms 91 ms ornl-atlcr1.es.net [134.55.208.62]
17 100 ms 90 ms 90 ms orgwy.ornl.gov [192.31.96.225]
```

After

```
C:\WINNT\system32>tracert thistle.csm.ornl.gov Tracing route
to thistle.csm.ornl.gov [160.91.212.74] over a maximum of 30
hops:
1 120 ms 90 ms 100 ms ornlvpn.ens.ornl.gov [192.31.96.190]
2 * * * Request timed out.
3 100 ms 90 ms 90 ms swgecsb-1.ens.ornl.gov [160.91.0.2]
4 91 ms 90 ms 90 ms thistle.csm.ornl.gov [160.91.212.74]
```



VPN establishment on the wire (ethereal)



No.	Time	Source	Destination	Protocol	Info
2	0.002010	192.168.1.1	Broadcast	ARP	Who has 192.168.1.2? Tell 192.168.1.1
3	0.112715	192.168.1.3	127.0.0.1	UDP	Source port: 0 Destination port: 62515
4	0.252866	192.168.1.3	192.31.96.190	UDP	Source port: 4192 Destination port: 62514
5	0.253859	192.168.1.3	192.31.96.190	UDP	Source port: 4193 Destination port: 62514
6	0.257416	192.168.1.3	192.31.96.190	ISAKMP	Aggressive
7	0.495975	192.31.96.190	192.168.1.3	ISAKMP	Aggressive
8	0.499759	192.168.1.3	192.31.96.190	ISAKMP	Aggressive
9	0.589165	192.31.96.190	192.168.1.3	ISAKMP	Transaction (Config Mode)
10	8.156152	192.168.1.3	192.31.96.190	ISAKMP	Transaction (Config Mode)
11	8.541376	192.31.96.190	192.168.1.3	ISAKMP	Transaction (Config Mode)
12	8.542388	192.168.1.3	192.31.96.190	ISAKMP	Transaction (Config Mode)
13	8.549678	192.168.1.3	192.31.96.190	ISAKMP	Transaction (Config Mode)
14	8.644883	192.31.96.190	192.168.1.3	ISAKMP	Transaction (Config Mode)
15	8.649811	192.168.1.3	192.31.96.190	ISAKMP	Quick Mode
16	8.751663	192.31.96.190	192.168.1.3	ISAKMP	Informational
17	8.752794	192.31.96.190	192.168.1.3	ISAKMP	Quick Mode
18	8.753001	192.168.1.3	192.31.96.190	ISAKMP	Quick Mode
19	11.220417	192.168.1.3	160.91.1.16	NBNS	Refresh NB <01><02>__MSBROWSE__<02><01>
20	11.220470	192.168.1.3	127.0.0.1	UDP	Source port: 0 Destination port: 62515
21	11.220735	192.168.1.3	192.31.96.190	ISAKMP	Quick Mode
22	11.320731	192.31.96.190	192.168.1.3	ISAKMP	Quick Mode
23	11.321029	192.168.1.3	192.31.96.190	ISAKMP	Quick Mode
24	12.720080	192.168.1.3	160.91.1.16	NBNS	Refresh NB <01><02>__MSBROWSE__<02><01>
25	12.809620	160.91.1.16	192.168.1.3	NBNS	Registration response NB 192.168.1.3
26	12.809696	192.168.1.3	160.91.1.16	NBNS	Refresh NB CHARADE<20>
27	12.895871	160.91.1.16	192.168.1.3	NBNS	Wait for acknowledgment response
28	13.290182	192.168.1.3	160.91.4.41	TCP	4195 > http [SYN] Seq=0 Ack=0 Win=65535 Len=
29	13.375405	160.91.4.41	192.168.1.3	TCP	http > 4195 [SYN, ACK] Seq=0 Ack=1 Win=5040C
30	13.375480	192.168.1.3	160.91.4.41	TCP	4195 > http [ACK] Seq=1 Ack=1 Win=65535 Len=
31	13.375988	192.168.1.3	160.91.4.41	HTTP	GET /~dunigan/ HTTP/1.1
32	13.464933	160.91.4.41	192.168.1.3	TCP	http > 4195 [ACK] Seq=1 Ack=418 Win=49983 Le
33	13.515766	160.91.4.41	192.168.1.3	HTTP	HTTP/1.1 200 OK[Unreassembled Packet]

```
Frame 6 (846 bytes on wire, 846 bytes captured)
0000  00 20 78 d9 3a 3f 00 11  11 31 0d 41 08 00 45 00   . x.:?.. .1.A..E.
0010  03 40 44 fd 00 00 80 11  10 27 c0 a8 01 03 c0 1f   .@D..... .'......
0020  60 be 01 f4 01 f4 03 2c  82 e6 2b 41 e8 6f 45 82   `......, ..+A.oE.
0030  40 e7 00 00 00 00 00 00  00 00 01 10 04 00 00 00   @....... ........
0040  00 00 00 00 03 24 04 00  02 2c 00 00 00 01 00 00   .....$.. .,......
0050  00 01 00 00 02 20 01 01  00 0e 03 00 00 28 01 01   ..... .. .....(..
0060  00 00 80 01 00 07 80 02  00 02 80 04 00 02 80 03   ........ ........
```


OpenVPN

openvpn.net

- Open source solution to VPN (windows/linux/*bsd)
- Authenticate with shared secret or public key (openssl)
- Tunnel IP or ether frames over UDP or TCP
- Operates as user-space daemon (doesn't use IPsec, no OS mods)
- Establishing a tunnel (a little routing/device trickery)
  - –Client connects to server daemon and authenticates
  - –New subnet addresses (temp) negotiated for tunnel endpoints
  - –Tunnel provides secure (encrypted/authenticated) path
  - –Client/server can use tunnel for NFS/rlogin/print etc.(rlogin 10.0.0.3)

private IP addresses
10.0.0.0
172.16.0.0
192.168.0.0

OpenVPN vs IPsec VPNs

- **IPsec VPNs**
  - –Complex (SA's, IKE)
  - –Kernel support or mods to IP
  - –Costly
  - –Early problems with NAT
  - + enterprise implementations
    - higher performance
    - scales
    - SecurID
- **OpenVPN**
  - –/+ Can attach from any host
  - + operates at user level
  - +based on SSL
  - - doesn't scale

Both provide pre-shared key or PKI authentication.

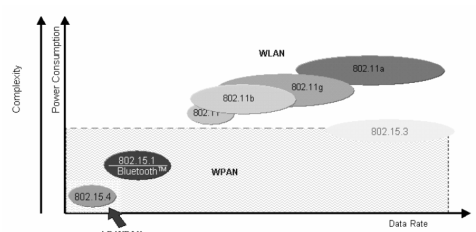
Both suffer from allowing a foreign host/net into the "inside"
probably need IDS/firewalls at VPN border

PPTP

- Microsoft's Point-to-Point Tunneling Protocol for VPN ('94)
  - –Windows 95, 98, NT (Nice Try)
- Don't need no standards or industry review! Invented their own:
  - –Authentication protocol (broken)
  - –Hash function (weak)
  - –Key generation algorithm
  - –Used a known encryption algorithm, but effective key length reduced by user-chosen ASCII passwords
  - –Plus other implementation bugs
- The implementation was badly flawed, some later patches

IPsec uses standard crypto and widely reviewed …

wireless



- •cell phones
- •pda's
- •sensor nets (industrial, military)
- •IEEE 802.11 802.15 (WPAN), 802.16 (WiMAX)

Wireless security

•wireless devices: VCR remote, PDA, cell phone, wireless ether

•wireless threats
- sniffing content
- user location
- traffic analysis
- becoming a node (new/impersonate)
- becoming a base station
- replay/inject
- jamming (DoS)

•wireless defenses
- spread spectrum
- authentication
- encryption at link level (LFSR, ECC, RC4)
- access control list (allowable MAC addresses)

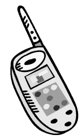
• wireless protocols: WPA, 802.11i/WEP, GSM, CDPD, bluetooth (weak security?), WiMAX

•Use IPsec or application security (ssh/ssl) for end-to-end security

Wireless ethernet security (802.11i)

- Wired equivalent privacy (WEP) has some known flaws
  - Short IV, CRC, no key mgt.
- WPA (WiFi protected access)
  - Builds on WEP
    - Still uses RC4
    - temporal key integrity protocol TKIP
  - Longer IV (48 instead of 24 bits)
  - Better message authentication (
  - Key hierarchy instead of single master key
  - Key refresh/update protocol
  - Support for Extensible Authentication Protocols (EAP)

Cellular security



GSM (invented their cypto ☹)
- compress, encrypt
- subscriber id module (SIM) provides authentication
- challenge/response (A3)
- A8 generates A5 key
- encryption (A5) (broken?)
  - A5 is stream cipher (3 LFSRs) (2^{40})
    A5/2 only 2^{16}
- most cell phone security is weak

Link layer encryption applies only between cell tower and your cell phone, landline transmission is NOT encrypted

Network security

VULNERABILITIES
- denial of service
  - ICMP smurf, redirects, unreachable
  - SYN flooding
  - frag, teardrop
- impersonation
  - host rename (LAN)
  - DNS
  - source routing
- Session capture
  - TCP seq number guessing
  - TCP hijacking
- Server/application attacks
  - application flooding (ftp,mail,echo)
  - buffer overflows
  - Software bugs
  - Improper configuration

COUNTERMEASURES
- disable
- configure properly
- xinetd, tcpwrappers
  - filters (allow, deny)
  - audit and alarm
- filtering portmap
- application filtering (securelib)
- patches
- scanners (Nessus, ISS)
- firewalls
- intusion detection & response
- Encryption
  - Link layer (WEP)
  - Transport layer (IPsec)
    - virtual private networks (VPNs)
  - Application layer (ssl, ssh, pgp)

kerberos

- trusted, third-party authentication service
- based on secret key, single login/signon
- part of MIT's project Athena (public domain), '85
- components: library, data base, authentication daemon, ticket-granting service, applications
- uses authenticators (for users and servers) and tickets
- provides:
  - authenticated messages
  - safe messages (encrypted checksum)
  - fully encrypted messages (encrypted telnet)
  - Single sign-on
- needs network time
- uses symmetric encryption (DES)
- applications must be "kerberized" (security "added on")
- does not trust hosts

- available for most UNIX's, DCE, ONC RPC, AFS/DFS, Windows 2000/XP
  - Just daemons, data files, and applications on top of the OS

Setting up kerberos

- get source from MIT (cygnus)
- designate secure authentication server machine (KDC)
  - maybe slave authentication servers for redundancy
- build applications (r-utilities, login, ftp, pop, klogin,kinit, klist, kadmin)
  - PAM (plugable authentication modules) may make it easier
- register principals (user, servers)
- data base (principal/password) is encrypted with master key
- install each server's key on server
  /etc/servtab– prevents host/IP impersonation
- client-only easy, (PC/MAC versions, linux)
  - Can't do RSA-ssh 'cause host needs your password to do kinit
  - Now there is a kerberized ssh
  - also a public-key version of Kerberos login, also smart card support

Kerberos components

- Key Distribution Center
  - Hardened host
  - Data base of user and server keys encrypted with master secret
  - Performs Authentication Service (AS)
  - Performs Ticket Granting Service (TGS)
  - Alternate/secondary KDC's
- Servers
  - Server software (print spooler, login, ftp) on various machines must be registered and have *server keys* securely stored on the machine
- Users
  - Account and Kerberos password

kerberos

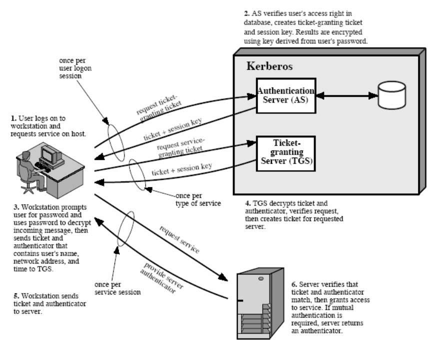


Figure 14.1 Overview of Kerberos

Kerberos exchanges

goals

- authentication of client to server
- optional authentication of server to client
- secure exchange of random session key

- Avoids plain-text passwords, permits single-signon
- Needham-Schroeder authentication

Table 14.1 Summary of Kerberos Version 4 Message Exchanges

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) **C → AS:** $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) **AS → C:** $E_{K_c}[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) **C → TGS:** $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) **TGS → C:** $E_{K_{c,tgs}}[K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$$

$$Ticket_v = E_{K_v}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$$

$$Authenticator_c = E_{K_{tgs}}[ID_C \parallel AD_C \parallel TS_3]$$

(c) Client/Server Authentication Exchange: to obtain service

(5) **C → V:** $Ticket_v \parallel Authenticator_c$

(6) **V → C:** $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$$Ticket_v = E_{K_v}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$$

$$Authenticator_c = E_{K_{c,v}}[ID_C \parallel AD_C \parallel TS_5]$$

Ticket	Authentication Service (AS)
Authenticator	Ticket Granting Service (TGS)
IP address (AD)	Client (C)
Time stamp (TS)	Server (V)
lifetime	
Key (K) K_{ctgs} and K_{cv} are random	

Authentication protocols

- See lecture 3, need confidentiality and timeliness
- Authenticators: shared secret and/or public keys
- Worry about replay attacks
  - Mallory copies a message and replays it later
  - Replay a time-stamped message within the time "window"
  - Suppress original message, send it later
  - Reflect message back to sender
- Replay defenses:
  - sequence numbers (bookkeeping problem)
  - timestamp (hard to keep distributed clocks sync'd, NTP)
  - challenge/response (nonce) (hard for connectionless transactions)
- Kerberos (Needham/Schroeder) uses timestamps

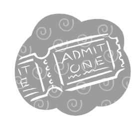
Kerberos credentials

authenticator

name/instance/realm of the client
timestamp

- used only once
- generated each time client wants to use a service
- encrypted with server's session key
- inhibits replay
- shows that the sender of the ticket is the same party to whom the ticket was issued

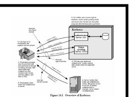
ticket



server
client
client workstation address
timestamp
lifetime
session key

- encrypted with server's key
- generated by TGS
- good for a single client and server
- TGS's voucher for the identity of the requestor of the service

Kerberos session



- user logs in, kerberized login sends <client name, TGS server name > to Kerberos AS
- Kerberos AS generates random session key (SK) and replies
  {< $SK_{\{TGS\}}$, {client name,WS addr, TGS-name, $SK_{\{TGS\}}$}$K_{\{TGS\}}$ >}K_c
- On client, user's password K_c is used to decrypt message
- To get a ticket for another service, client sends a message to TGS, with authenticator (encrypted with $SK_{\{TGS\}}$), the sealed TGS ticket, and the server name
- TGS generates a random session key $SK_{\{server\}}$ and replies with
  {< $SK_{\{server\}}$, {client name,WS addr,sever, $SK_{\{server\}}$} $_{\{server\}}$ }all encrypted with $SK_{\{TGS\}}$
- the client can send a request to the server consisting of the server's encrypted ticket, and an authenticator encrypted with $SK_{\{server\}}$
- the server can decode the ticket and get the session key $SK_{\{server\}}$ and decode and verify the client (check for replay)
- server adds 1 to timestamp and sends to client encrypted with $SK_{\{server\}}$ (mutual authentication)

kerberizing

- you can add Kerberos calls to your own client/servers
- need Kerberos data base, authenticator, ticket-granting server, and administrative programs
- can use klogin, but better if you have kerberized BSD utilities
- Kerberos calls added to login, r-utilities, NFS
- "rlogin -x " sets up encrypted session, every packet is encrypted
- Kerberos API (later)

Kerberos v4

- typical client/server application
- library requests, just UDP packets
- Kerberos servers listening on well-known ports (88)
- encryption: modified DES CBC
  – PCBC has weaknesses
- MAC: Juneman checksum on (key, msg)
- Random numbers (session keys)

```
srandom(time.tv_usec ^ time.tv_sec ^
  getpid() ^ gethostid() ^ counter++ )
key = random()
```

☹

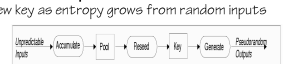
```
  SERVICES
kerberos        88/udp          kdc
# Kerberos authentication--udp
kerberos        88/tcp          kdc
# Kerberos authentication--tcp
klogin          543/tcp
# Kerberos authenticated rlogin
kshell          544/tcp         cmd
# and remote shell
kerberos-adm    749/tcp
# Kerberos 5 admin/changepw
kerberos-adm    749/udp
# Kerberos 5 admin/changepw
kerberos-sec    750/udp
# Kerberos authentication--udp
kerberos-sec    750/tcp
# Kerberos authentication--tcp
kerberos_master 751/udp
# Kerberos authentication
kerberos_master 751/tcp
# Kerberos authentication
krb5_prop       754/tcp
# Kerberos slave propagation
kpop            1109/tcp
# Pop with Kerberos
eklogin         2105/tcp
# Kerberos encrypted rlogin
krb524          4444/tcp
# Kerberos 5 to 4 ticket xlator
```


Kerberos v5

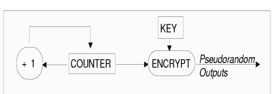
- support of DCE
- more functionality
- new encodings
  – ASN.1 data representation (v4: byte-order bit)
  – address encoding (v4: IPv4 only)
  – stronger random numbers (yarrow with /dev/random)
  – AS and TGS exchanges include a nonce instead of timestamp
  – selectable encryption/MAC
    - MAC: DES of md5/md4/DES-CBC
    - encryption+MAC: DES + md4/md5/crc
- principal names multicomponent
  – v4 was name/instance/realm (40 max)
  – v5 name/realm
- new ticket flags (delegation) and longer lifetimes
- v5 will handle v4 requests
- More recently:
  – public key for initial authentication (DoD using smart card to hold public key, need PIN)
  – one-time password support, kerberized ssh
  – Kerberos v5 RFC1510

kerberos v5 random numbers

- KDC generates random session keys
- yarrow using /dev/random and packet interarrival times for random input
  – Initial seed from master key (and other realm keys if available)
- yarrow keeps a fast and slow pool of random bits mixed with SHA-1
  – Reseed and new key as entropy grows from random inputs



- Output bits are generated from 3DES using a key from the fast pool



- Wipes memory and saves pool to file on exit

v5 tickets

- proxiable TGT -- can be used to request tickets for a different net address (Alice can let Bob use her printer)
- forwardable TGT -- can be presented to a remote TGS
- lifetimes
  – longer lifetimes (v4: 21 hrs) (v5: start/end)
  – renewable (by KDC)
  – postdated (good a week from now for 2 hrs, KDC clears INVALID flag)

Kerberos cross-realm authentication

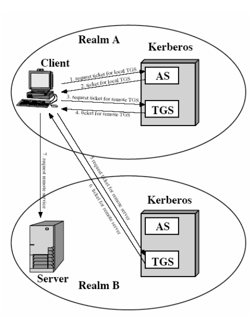


Figure 14.2 Request for Service in Another Realm

Why not Kerberos?

- every network service must be modified
- Kerberos server must be physically secure with hardened OS
- export restrictions
- off-line password attack on message from KDC to client
- if password is disclosed, eavesdropper can decrypt other tickets and spoof servers and users
- (need kerberos-able client)

Still, better than anything else.

- also used in DCE and Windows XP
- part of DoD single-signon/common access card public key (on smart card) login to kerberos

Other variations

SESAME

- European project
- based on Kerberos
- uses public key
  - ticket encrypted with user's public key
  - AS stores only public keys, not as vulnerable

CORBA -- technology for distributed applications

- set of specs
- object request broker (ORB)
- security spec released '95
- authentication, access control, audit, message protection

Secure operating systems

- design issues
  - hardware features
  - OS features
  - design principles
- Examples
  - Multics, TMACH, SELinux, OpenBSD
- evaluation methods
  - Orange book, common criteria, FIPS
- Why secure applications are not enough. NSA -- really need secure OS (required reading)

features for security

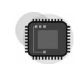
Hardware

- privileged state
- privileged instructions
- memory protection (RO,NX)
- timer interrupt
- system call (trap/SVC)
- hardware RNG ?
- TPM?

OS

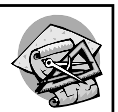
- user authentication
- memory protection
- file/device protection
- permit sharing but not hogging
- provide integrity and consistency
- scheduling
- interprocess communication

Trusted platform module (TPM)



- Window's Vista, Linux?, coming to MAC OS
- Attached processor for storing keys, doing crypto, RNG
- Tamper resistant
- Authenticate user/system (digital rights)
- Secure startup, trusted OS query, disk encryption (stolen laptop)
- Code signing
- Trusted computing group "standard"
- Number of vendors making TPMs

Secure OS design



- design security in from the beginning
- define access rules for subjects/objects and allowable info flows (security policy)
- Label objects (top secret, secret, unclassified, read-only)
- least privilege (fine grained)
- small kernel (TCB)
- separation (temporal, logical, cryptographic), layers or compartments
- complete mediation -- every access checked
  - Mandatory access control (MAC)
- default is deny
- ease of use
- apply software engineering principles

Access control

- access matrix (subjects/objects/access)
- by objects: access control list (ACL), who can do what to this object, UNIX file permissions
- by subjects: capability tickets --what this subject/user has access to
- maintained by OS
- role-based access control (RBAC)
  - Each process/user has a "role"
    - Doctor/nurse/administrator
  - A user may have several roles
  - Config files specify to which domains a role has access
  - Can associate a role with a "method"
  - Easy to change privileges of a role
  - Principal of least privilege

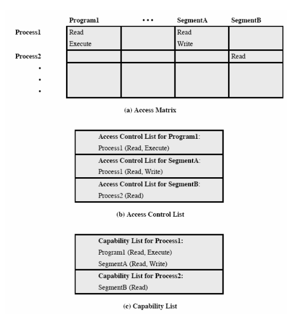


Figure 20.3 Access Control Structures

TCB

Trusted Computing Base

- reference monitor
- implements/enforces security
- authentication and access control
  - –No read-up or write-down
  - –Labeled objects, user clearance
  - –"need to know" role-based
- tamper resistant
- can't be bypassed
- small/simple, correctly implemented
- verifiable ?

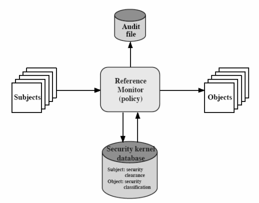


Figure 20.4 Reference Monitor Concept

Secure OS's

- Multics
- Scomp
- KSOS (3-layer, PDP-11)
- kernelized VM/370
- secure UNIX/Linux
  - –UCLA Secure UNIX
  - –TMACH/DTOS
  - –CMWs, NetTop
  - –bastille Linux, NSA secure Linux (SELinux), Grsecurity
- OpenBSD

multics

- B2 multilevel secure OS (mid 60's)
- multilayered (rings)
- extension of 2 layer systems
- upper rings implemented in software (processor: GE645)
- kernel, ring 0
- "traps" for requesting ring services (gate)
- uses capabilities
- segmented address space
- design review (PL/I)
- need more hardware assist (ring-crossing)

UCLA secure UNIX (1979)

- three layers
- user/supervisor/kernel (PDP-11)
- small secure kernel
- kernel interface layer
- application layer
- all security functions in one place
- capability list maintained by kernel
- objects: processes, pages, devices

TMACH

Trusted MACH

- based on CMU's MACH kernel
- with OSF UNIX personality
- sponsored by DARPA and TIS
- available for 486/586

TMACH goals

- B3 security -- data integrity
- portable to many platforms
- competitive performance
- extensible
- UNIX personality

TMACH architecture

kernel, small, secure

- layering, modularity, abstraction, data hiding
- TCB servers, multi-level secure servers
- non-TCB code, the OS user interface

TMACH security

- labeled objects (military model)
- user clearance
- mandatory access control
- access mediated by reference monitor
- auditing
- ACL's
- trusted path for user authentication
- subdivided privileges (least privilege)
- task communication (ports)
- covert channel analysis
- trusted startup and recovery
- security model (Bell and LaPadula)

Secure Linux

- NSA's Security-Enhanced linux
  - Mandatory access control (type-enforcement and RBAC)
  - Separation of information based on confidentiality and integrity
  - Based on Linux Security Modules (LSM)
  - Latest RedHat/Fedora has SELinux option, file ACL's, firewalling
- Hardened linux
  - Bastille linux (scripts to harden configuration)
  - Trustix
  - Engarde Secure Linux
  - Openwall Linux (Owl)
  - RSBAC
  - Grsecurity
- Support for TPM
- Use for bastion hosts and "external servers" (http, dns, login)

Securing the Linux kernel

- Need lots of "hooks" in kernel source to check permissions
  - File open/read/write, process create, sockets, …
  - LSM model (modules) SELinux
  - Grsecurity others require kernel patches
- Non-trivial configuration files to establish "policy"
- Grsecurity has a learning mode to establish what privileges a process needs
- User roles can be restricted
  - You can only read mail (can't compile)
  - Different administrative privileges

Example ACL

Printer daemon ACL (Grsecurity)

```
Subject /usr/sbin/cupsd o {
        /                      h
        /etc/cups/certs
        /etc/cubs/certs/0       wcd
        /etc/group              r
        -CAP_ALL +CAP_CHOWN  +CAP_DAC_OVERRIDE
        bind disabled
        connect disabled
}
```

- Configure services to have certain capabilities and access rights
- Configure users to be part of certain groups/roles
- Principle of least privilege – even if you buffer overflow cupsd …

Hardening linux

- Disable net services
- Reduce setuid programs
- Active password mgt/renewal
- Default mask
- Limit root login locations
- Set BIOS password
- Get rid of r-utilities
- Login banners
- Remote syslog
- More detailed event logging
- User-specific /tmp areas
- Firewall enabled
- Enable NTP
- Enable port scan detector
- Incorporate StackGuard/PaX
- Support no-exec data areas (MMU)
- Auto patch/update
- Immutable/append files (chattr)
- Encrypted file systems
- SELinux or Grsecurity

… read the books

OpenBSD

- Most secure of the open source UNIX
- Developed in Canada, so crypto software included
  - Kerberos v5
  - IPsec
  - openssh/openssl
  - Support for crypto hardware
- Immutable and append-only files, no writing to /dev/mem /dev/kmem
- Actively audit source code looking for vulnerabilities as well as timely patches for bugs discovered by "others"
- Default installation – non-esential services disabled
- PRNG (/dev/srandom) (/dev/random is for hardware RNG)

Security validation (assurance)

demonstrate the security of an OS or application or crypto device

- verification
- penetration analysis
  - test security services
  - try to bypass security
  - tiger team
  - demonstrate presence of errors, not absence
- informal validation
  - requirements checking
  - design and code reviews
  - module and system testing
  - certification (Orange Book or itsec or Common Criteria)

DoD certification (historical)

Requirements for a secure OS
- security policy
- authentication
- labeling of objects and subjects
- accountability (audit logs)
- assurance (enforce/evaluate)
- tamper resistant
- documentation

ORANGE BOOK

DoD Trusted Computer System Evaluation Criteria ('85)
- hierarchy of requirements (D,C1,C2,B1,B2,B3,A1)
- list of security features
- strength of OS design
- limit data access (privacy)
- control data flow
- certification service
- objective assessment

Orange book ratings

- D -- minimal (D for DOS)
- C1 -- discretionary
- C2 -- controlled access
- B1 -- labeled
- B2 -- structured
- B3 -- security domains
- A1 -- verified

- Might be able to add features to an OS to qualify for C1-B1
- B2 requires security part of OS design.
- B3/A1 provable model of security

Ratings

C1 -- discretionary access
- memory protection (user vs OS)
- object access control (owner/group/world)
- user authentication (password)
- discretionary access controls
- penetration testing
- e.g., MVS with RACF

C2 -- controlled access
- single user access controls (ACL)
- audit logs (tamper resistant)
- object reuse, protect memory, files, swap
- e.g, MVS/ACF2, VMS, DEC UNIX

Ratings

B1 – labeled
- mandatory access controls (privacy)
- labeled objects (incl. devices)
- label printer output
- prevent read-up and write-down (Bell-LaPudula)
- analysis and testing of design and source code
- informal model of security policy e.g., CMW's (compartmentalized mode workstation)

B2 -- structured protection
- test and review of design
- principle of least privilege
- trusted path (user/tty/OS) (mutual authentication)
- security kernel (TCB)
- programs must report security level changes
- covert channels identified and bandwidth estimated, e.g., Multics

Covert channels

- ways to reveal info to lower level
- low-bandwidth
- difficult to eliminate
- use memory/file or timing
- existence of a file (bit 0 or 1)
- mode of a file (0 or 1)
- high CPU load (0 or 1)
- process name (0 or 1)
- network connection (0 or 1)

ratings

B3 -- security domains
- stronger design
- ACL supports user denial
- ACL supports read/write (integrity)
- full access control
- active audits (alerts)
- penetration resistant
- secure startup and crash, e.g., TMACH (applied for B3)

A1 -- formal verification
- formal, provable security model
- top-level specification
- demonstrate spec follows model
- implementation consistent with spec
- formal analysis of covert channels, e.g., Scomp

Orange book shortcomings

- miss one feature, lose rating
- local software invalidate ?
- OS patches invalidate ?
- mandatory access controls cumbersome
- write-up would allow virus

NT got a C2 rating ('96)
BUT
epoxy-shut floppy
no network
Compaq 386

UK's itsec

UK software certification ('91), fast-track assessment, list of certified products

Assurance levels:

- E0 – no assurance
- E1 – informal architecture design, some testing, secure dn.
- E2 – penetration tested, audit trail
- E3 – source code/drawings, acceptance procedure, re-test
- E4 – formal security model, configuration control
- E5 – independent configuration control
- E6 – formal architecture description and correlation with design and testing

Common Criteria (ISO 15408 '99)

Combine US and EU criteria
Functional requirements: desired security behavior
Assurance requirements: assure claimed security measures are effective
Assurance levels

- EAL1 – functionally tested
- EAL2 – structurally tested
- EAL3 – methodically tested and check
- EAL4 – methodically designed, tested, and reviewed
- EAL5 – semi-formally designed and test
- EAL6 – semi-formally verified design and tested
- EAL7 – formally verified design and tested

CUPERTINO, Calif. - May 21, 2002 - Symantec Corp. (NASDAQ: SYMC), the world leader in Internet security, today announced that Symantec Enterprise Firewall 7.0 has been awarded Common Criteria Evaluation Assurance Level 4 (EAL4) certification. This prestigious certification assures customers that Symantec Enterprise Firewall has gone through a long and rigorous testing process and conforms to standards sanctioned by the International Standards Organization.

CC functions requirements

Class	Description
Audit	Involves recognizing, recording, storing and analyzing information related to security activities. Audit records are produced by these activities, and can be examined to determine their security relevance.
Cryptographic support	Used when the TOE implements cryptographic functions. These may be used, for example, to support communications, identification and authentication, or data separation.
Communications	Provides two families concerned with non-repudiation by the originator and by the recipient of data.
User data protection	Specifying requirements relating to the protection of user data within the TOE during import, export and storage, in addition to security attributes related to user data.
Identification and authentication	Ensure the unambiguous identification of authorized users and the correct association of security attributes with users and subjects.
Security management	Specifies the management of security attributes, data and functions.
Privacy	Provides a user with protection against discovery and misuse of his or her identity by other users.
Protection of the TOE security functions	Focused on protection of TSF (TOE security functions) data, rather than of user data. The class relates to the integrity and management of the TSF mechanisms and data.
Resource utilization	Supports the availability of required resources, such as processing capability and storage capacity. Includes requirements for fault tolerance, priority of service and resource allocation.
TOE access	Specifies functional requirements, in addition to those specified for identification and authentication, for controlling the establishment of a user's session. The requirements for TOE access govern such things as limiting the number and scope of user sessions, displaying the access history and the modification of access parameters.
Trusted path/channels	Concerned with trusted communications paths between the users and the TSF, and between TSFs.

TOE – target of evaluation

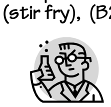
CC assurance requirements

Class	Description
Configuration management	Requires that the integrity of the TOE is adequately preserved. Specifically, configuration management provides confidence that the TOE and documentation used for evaluation are the ones prepared for distribution.
Delivery and operation	Concerned with the measures, procedures and standards for secure delivery, installation and operational use of the TOE, to ensure that the security protection offered by the TOE is not compromised during these events.
Development	Concerned with the refinement of the TSF from the specification defined in the ST to the implementation, and a mapping from the security requirements to the lowest level representation.
Guidance documents	Concerned with the secure operational use of the TOE, by the users and administrators.
Life cycle support	Concerned with the life-cycle of the TOE include lifecycle definition, tools and techniques, security of the development environment and the remediation of flaws found by TOE consumers.
Tests	Concerned with demonstrating that the TOE meets its functional requirements. The families address coverage and depth of developer testing, and requirements for independent testing.
Vulnerability assessment	Defines requirements directed at the identification of exploitable vulnerabilities, which could be introduced by construction, operation, misuse or incorrect configuration of the TOE. The families identified here are concerned with identifying vulnerabilities through covert channel analysis, analysis of the configuration of the TOE, examining the strength of mechanisms of the security functions, and identifying flaws introduced during development of the TOE. The second family covers the security categorization of TOE components. The third and fourth cover the analysis of changes for security impact, and the provision of evidence that procedures are being followed. This class provides building blocks for the establishment of assurance maintenance schemes.
Assurance maintenance	Provides requirements that are intended to be applied after a TOE has been certified against the CC. These requirements are aimed at assuring that the TOE will continue to meet its security target as changes are made to the TOE or its environment.

FIPS 140-1

Crypto module security (hardware, e.g. encryptors, crypto cards)

- Level 1 – uses FIPS approved algorithms
- Level 2 – tamper-evident seals, role-based authentication (C2)
- Level 3 – self-destruct (zero's itself) if tampered with, identity based authentication (B1)
- Level 4 – sophisticated tamper detection and zero'ing, resist environmental attacks (stir fry), (B2)



☹ The problem of establishing that a program, application, or OS meets any particular security requirement is known to be fundamentally unsolvable!

Next time …

Writing secure code

Crypto API's

Secure applications

Lectures

1. Risk, viruses
2. UNIX vulnerabilities
3. Authentication & hashing
4. Random #s classical crypto
5. Block ciphers DES, RC5
6. AES, stream ciphers RC4, LFSR
7. **MIDTERM** ☹
8. Public key crypto RSA, D-H
9. ECC, PKCS, ssh/pgp
10. PKI, SSL
11. Network vulnerabilities
12. Network defenses, IDS, firewalls
13. IPsec, VPN, Kerberos, secure OS
14. Secure coding, crypto APIs
15. review

CS594 project due: December 1