9. Authorization Logic
ENEE 757 | CMSC 818V

Prof. Tudor Dumitraș
Assistant Professor, ECE
University of Maryland, College Park

http://ter.ps/757
https://www.facebook.com/SDSAtUMD

Today’s Lecture

• Where we’ve been
  – Passwords
  – Biometrics
  – Kerberos

• Where we’re going today
  – Authorization logic

• Where we’re going next
  – Usability issues in access control
Access Control

• Principal makes a request for an object
• Reference monitor grants or denies the request

Ex: File editor → Send file → File server
Ex: Host → Forward packet → Firewall

• Authentication: Determining who made request
• Authorization: Determining whether requestor is trusted to access an object
  • This is the decision that the reference monitor must make

Two Implementation Concepts

• Access control list (ACL)
  – Store column of matrix with resource
  – Relies on authentication: need to know user
  – Delegation: let other process act under current user
    • UNIX su/sudo, Windows UAC

<table>
<thead>
<tr>
<th></th>
<th>File 1</th>
<th>File 2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>read</td>
<td>write</td>
<td>-</td>
</tr>
<tr>
<td>User 2</td>
<td>write</td>
<td>write</td>
<td>-</td>
</tr>
<tr>
<td>User 3</td>
<td>-</td>
<td>-</td>
<td>read</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User m</td>
<td>Read</td>
<td>write</td>
<td>write</td>
</tr>
</tbody>
</table>

Access Control Matrix [Lampson]

• Capabilities
  – User holds a ticket for each resource
  – Two variations
    • Store row of matrix with user, under OS control
    • Unforgeable ticket in user space
  – Reference monitor checks ticket: does not need to know identity of user/process
  – Delegation: Process can pass capability at run time
Role-Based Access Control

Users  Roles (also known as Groups)  Resources

- engineering
- marketing
- human res

Server 1  Server 2  Server 3

- Role examples: Administrator, PowerUser, User, Guest
  - Assign permissions to roles; each user gets permission
  - Advantage: users change more frequently than roles

A Logic of Authorization

- Inference rules

\[
\begin{align*}
\text{keystr signed } F \\
\text{key(keystr) says } F
\end{align*}
\]

(says-I)

\[
\begin{align*}
A \text{ says } (A.S \text{ says } F) \\
A.S \text{ says } F
\end{align*}
\]

(says-LN)

Translate digital signatures into says statements

Note the difference from:
- B.S says F
- S says F
A Logic of Authorization – cont’d

• Inference rules

If $F$ is true, then we can consider that any principal says $F$

\[
\frac{F}{A \text{ says } F}
\]  
(says-I2)

Modus ponens

\[
\frac{A \text{ says } (F \rightarrow G) \quad A \text{ says } F}{A \text{ says } G}
\]  
(impl-E)

A Logic of Authorization – cont’d

• Inference rules

A delegates authorization to B

\[
\frac{A \text{ says } (B \text{ speaksfor } A) \quad B \text{ says } F}{A \text{ says } F}
\]  
(speaksfor-E)

Limited delegation: A authorizes B to approve action $U$

\[
\frac{A \text{ says } (B \text{ speaksfor } A,S) \quad B \text{ says } F}{A,S \text{ says } F}
\]  
(speaksfor-E2)

\[
\frac{A \text{ says } \text{delegates}(A, B, U) \quad B \text{ says } \text{action}(U)}{A \text{ says } \text{action}(U)}
\]  
(delegate-E)
Authenticating a Channel

- Reference monitor receives a request $C$ says action(s)
- An access-control list usually specifies named principals
- Thus, reference monitor must collect credentials to prove that $A$ says action(s) for some $A$ on the access control list
- Credentials typically correspond to statements of the form
  \[ K_{CA} \text{ signed } (key(K_A) \text{ speaks for } key(K_{CA}).A) \]

- Two general methods
  - **Push:** The sender on the channel $C$ collects $A$’s credentials and presents them to authenticate the channel to the receiver.
  - **Pull:** The receiver looks up $A$ in some database to get credentials for $A$ when it needs to authenticate the sender.

How to Implement an Authenticated Channel

- Credentials come from **certification authorities** (CA)
  \[ K_{CA} \text{ signed } (key(K_A) \text{ speaks for } key(K_{CA}).A) \]
- Two implementation approaches: $K_{CA}$ can be a **public key** or a **symmetric key**

- Symmetric cryptography
  - Shared secret between sender and receiver: authentication messages must be sent and received in real time
  - Need an **online relay**
  - Example: Kerberos Key Distribution Center (KDC) issues statements of the form:
    \[ key(K_{TGS}) \text{ says } (key(K_{C,TGS}) \text{ speaks for } key(K_{TGS}).C) \]
    - Certifies to TicketGranting Service that session key $K_{C,TGS}$ speaks for the client $C$

- Public key cryptography
  - Broadcast channel: you can send a signed message without knowing who will receive it
  - Can store **signed certificates** locally
  - Example: Web certification authorities
Groups

- A group is a principal whose members speak for it

- Simplest way to define a group $G$ is for a defining $CA$ to issue certificates

  $key(K_{CA})$ says $P_1$ speaks for $key(K_{CA}).G$
  $key(K_{CA})$ says $P_2$ speaks for $key(K_{CA}).G$

  ... for group members $P_1, P_2, ...$

Example Proof

- Tudor says (BumJun speaks for Tudor.Students)
- BumJun says action(AVW 3425)
- Tudor says delegate(Tudor, Tudor.Students, AVW 3425)
- Tudor.Students says action(AW 3425)
- Tudor says action(AW 3425)

Stored in the reference monitor.
Part of the TCB.
**Traditional Access Control Lists**

- **Implicitly known to the reference monitor.**
  - BumJun speaks for Tudor.Students
- **Not signed (part of TCB)**
- **Received in the request.**
  - BumJun says action(AVW 3425)
  - Tudor.Students says action(AWV 3425)
- **Stored in the reference monitor.**
  - Part of the TCB.

**A “Pull” Approach**

- **Retrieved by reference monitor.**
  - Tudor says (BumJun speaks for Tudor.Students)
  - Tudor says delegate(Tudor, Tudor.Students, AVW 3425)
- **Received in the request.**
  - BumJun says action(AVW 3425)
  - Tudor.Students says action(AWV 3425)
- **Stored in the reference monitor.**
  - Part of the TCB.
- **Credentials stored on server that is not part of TCB**
A “Push” Approach

A “Proof Carrying” Approach
[Appel & Felten, CCS’99]
Roles

• Suppose a principal wants to limit its authority
  – Tudor “as” GamePlayer
  – Tudor “as” SysAdmin

• Intuition: A “as” R should be weaker than A
• A can accomplish this by enabling statements of the form

\[ A.R \text{ says } F \]

to be created

Programs as an Application of Roles

• Acting in a role is like acting according to some program
• If host H is running program with binary code I, then H can make

\[ H.I \text{ says } F \]
  – for a statement F made by the process running I

• Instead of using the whole program I, H can instead make

\[ H.D \text{ says } F \]
  – where D = h(I) for h a cryptographic hash function, and using

\[ D \text{ speaksfor } P \]
  – where P is the program name
Loading Programs Securely

• To load program named $P$, on host $H$
  – Creates a process $pr$
  – Reads text $I$ of file $P$ from the file system
  – Finds credentials for $D$ speaksfor $P$ and checks $h(I) = D$
  – Copies $I$ into $pr$
  – Gives $pr$ ability to write to channel $C$
  – Emit: $H$ says $(C$ speaksfor $H,P)$

• Now $pr$ can issue requests on channel $C$
  – Will be granted if $H.P$ is on ACL

Parasitic Virus Control

• Some viruses alter texts of programs in the file system
  – If $I'$ is the infected program text, then $D' = h(I')$ will be different from $D = h(I)$, and so $D$ speaksfor $P$ will not apply

• Certification authority $CA$ can issue certificates
  
$K_{CA}$ signed $P$ speaksfor key($K_{CA}$).trustedSW
$K_{CA}$ signed $N$ speaksfor key($K_{CA}$).trustedNodes
$K_{CA}$ signed $(P$ speaksfor key($K_{CA}$).trustedSW $\rightarrow$
  $(N$ speaksfor key($K_{CA}$).trustedNodes $\rightarrow$
  $N.P$ speaksfor key($K_{CA}$).trustedNodeAndSW))

– where trustedSW, trustedNodes, and trustedNodeAndSW are group names, $P$ is a program name, and $N$ is a node name

Configure ACLs with these groups
Secure Booting

- ‘trustedNodes’ should be computers that
  - run operating systems validated before booting
  - validate other software before loading it

- Validating O/S during boot is like validating other software
  - Machine W holds h(I) in boot ROM, where I is O/S image
    - i.e., h(I) speaks for P

- To create a channel C such that C speaks for W.P, W can
  - Generate a new signature key pair \( <K_{W,P}, K_{W,P}^{-1}> \), and
  - Give \( K_{W,P}^{-1} \) to P, along with \( K_W \) signed key(\( K_{W,P} \)) speaks for key(\( K_W \)).P

- Private key for \( K_W \) must be protected in secure hardware (e.g. TPM)
  - Otherwise, O/S can read it

TCG Integrity Challenge and Response

- Each TPM is equipped with a different private key \( K_{TPM}^{-1} \) and a certificate
  \[ K_{TPM} \text{ signed } \{ \text{key}(K_{TPM}) \text{ speaks for key}(K_{TPM}).\text{TrustedProcessingModules} \} \]
  - signed by a “trusted platform module entity” (TPME)
  - TrustedProcessingModules is a group

- Remote machine can query TPM for contents of PCRs

- TPM responds with signed PCR values
  - Think of it as signed with \( K_{TPM} \)
    \[ K_{TPM} \text{ signed } \text{PCRvals} = ... \]
  - (In reality, is not signed with \( K_{TPM} \) but another Attestation Identification Key (AIK) is used to enhance privacy)
Sources

• Various slides from Mike Reiter and Dan Boneh

Review of Lecture

• What did we learn?
  - Authorization logic

• Paper discussion: “Designing an Authentication System”
  - Discussion lead: Zikai
  - Scribe: BumJun

• What’s next?
  - Usability issues in access control