

Security (Part 1)

Security

- **Framework**
 - authentication
 - access control
- **Breaking and entering**
 - prevention

Concerns

- **Authentication**
 - who are you?
- **Access control**
 - what are you allowed to do?
- **Availability**
 - can others keep you out?

Logging In ...

- **Username/password**
 - who knows the passwords?

One-Way Functions

- $f(x)$ is easy to compute
- $f^{-1}(x)$ is extremely difficult, if not impossible, to compute
 - Unix password file contains image of each password
 - » `/etc/passwd` contains `twd:y`
 - » `twd` logs in, supplies `x`
 - » if $f(x) = y$, then ok
 - » `/etc/passwd` is readable by all

Unix, and many other systems, authenticate users by having them supply their passwords. Rather than keep the plaintext of the passwords a file where they might be seen by others, Unix stores encrypted passwords, as described in the slide. Much of our discussion on cryptology-related concerns comes from *Applied Cryptography*, 2nd Edition, by Bruce Schneier, John Wiley and Sons, 1996.

Dictionary Attack

- For all words in dictionary, compute $f(\text{word})$
- Find word such that $f(\text{word}) = y$

Systems that employ just one-way functions to protect their passwords are vulnerable to dictionary attacks.

Counterattack

- **Salt**
 - for each password, create random “salt” value
 - `/etc/passwd` contains $(f(\text{append}(\text{word}, \text{salt})), \text{salt})$
 - 12-bit salt values in Unix
 - attacker must do dictionary attack 4096 times, for each salt value
 - » done ...
 - » **Feldmeier and Karn produced list of 732,000 most common passwords concatenated with each of 4096 salt values**
 - covers ~30% of all passwords

Unix uses “salt” as a means to foil dictionary attacks, though it’s probably not of tremendous use anymore.

Counter Counter Attacks

- **Don't allow common access to password images**
 - /etc/passwd contains everything but password images and is readable by all
 - /etc/shadow contains password images
- **Use better passwords**
 - “w7%3ngibwy6” rather than “fido”
- **Use strong cryptography and smart cards**
 - combined with PINs
- **Use biometrics**

Defeating Authentication

- What are the prime factors of

5325138870287932192846843055513588820529482732761
0407403175727513859436883214523893737052953027480
7754890798107434809613388354335732832883202827204
2055572159979867180328891700281777291005819624495
2509309592137003269247211376423318797402174094174
3851002617777645320194597739213700326924721137642
3318797402174094174385100261777764532019459773388
0145388493887041421512320698181588962921353458454
9713993496308859388014538849388704142151232069818
15889629213534584549713993496308859?

Hint: one of them is:

6438080068035544392301298549614926991513861075340134329180734395241382648
4237063006136971539473913409092293733259038472039713333596954925632262097
9036686633213903952966175107096769180017646161851573147596390153.

The point is that defeating a decent authentication technique is probably too tough to bother, particularly when there are probably other, much easier ways of breaking in.

Defeat Authentication, Sneakily ...



CS 167

XXI-10

Copyright © 2006 Thomas W. Doepfner. All rights reserved.

Since defeating a decent crypto scheme is far too difficult, we might try stealing someone's password. If you walked up to a PC with the contents of the this slide on the screen, you might be tempted to type in your password. However, there's the risk that this screen was not put up by the system, but by some evil user who's trying to trick you into yielding your password. Recent versions of Windows provide a means for protection against this sort of attack: if you type ctrl-alt-delete, the response is guaranteed to be directly from the operating system and not from any application program. If there is no user logged on, the response will be a guaranteed legitimate login screen. If a user is logged on, the response will be a window to the system's task-manager application. This notion of an input that's guaranteed to be handled by a trusted component (the operating system in this case) is called *trusted path*.

More Terminology

- **Principals/subjects**
 - the active parties
 - users
 - processes
 - etc.
- **Objects**
 - the passive parties: the things being protected
 - files
 - processes
 - etc.

Access Matrix

	<i>/a/b/c</i>	<i>/x/y/z</i>	Process 112	twd's xterm
Aaron	rw		rwc	protection domain
Dan	r			
Adam		rw		
Dave	r			
Joel				rw
Process 117	rw	r	rwc	

ACL

CS 167

XXI-12

Copyright © 2006 Thomas W. Doepfner. All rights reserved.

The access matrix represents the intended authorizations in the system. Rather than representing it directly, the information is usually stored with the objects (labeling the columns) or sometime with the subjects (labeling the rows). The former approach is often called access control lists (ACLs), the latter is often called capabilities. The former is much more common.

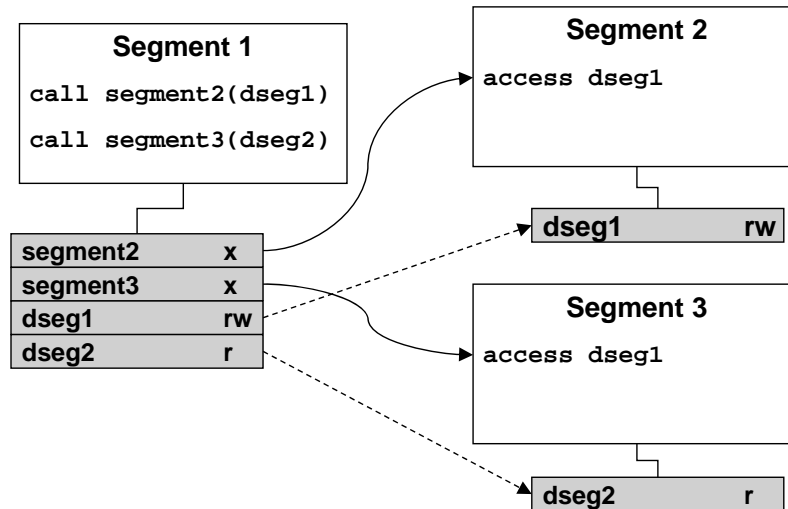
To be precise, a row of the matrix is a collection of capabilities, where each one allows some particular sort of access to some object. The set of capabilities given in one row defines the subject's *protection domain*.

Principle of Least Privilege

- **Make each protection domain as small as possible**
 - makes most sense when applied to processes

This rather obvious idea was first enunciated in the 60s and has been more or less ignored ever since.

Capability-Based Systems



CS 167

XXI-14

Copyright © 2006 Thomas W. Doepfner. All rights reserved.

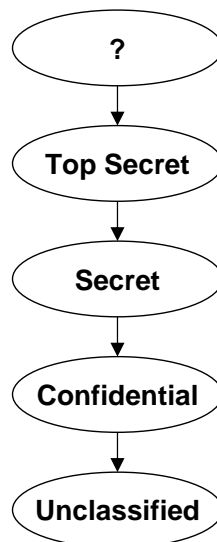
A popular idea in the 70s and early 80s was the notion of systems whose security models were based solely on the use of capabilities. Think of a program as consisting of a number of segments. (Segments here correspond to logical pieces of a program, such as procedures, files, etc.) To execute the code in a segment or to access data in it, you need a capability for the segment. Such capabilities behaved both as a set of permissions and an address. They were implemented in such a way so as not to be forgeable: possession of a capability necessarily implies that you have access to the indicated segment. When executing code in a segment, a process would have a segment table referring to the complete set of other segments that are accessible. In the slide, the process executing segment 1 has a segment table giving it capabilities to segments 2 and 3 (containing code) as well as dseg1 and dseg2 (containing data). When, for example, the process, while executing in segment 1, calls segment 2, the latter's segment table is set up to contain the segments passed as arguments.

Capability-based systems, though interesting, were never a commercial success (and, in fact, were occasionally spectacular failures).

Mandatory vs. Discretionary Access Control

- **Discretionary**
 - ACLs, capabilities, etc.
 - access is at the discretion of the owner
- **Mandatory**
 - government/corporate security, etc.
 - access is governed by strict policies

Mandatory Access Control (1)



CS 167

XXI-16

Copyright © 2006 Thomas W. Doepfner. All rights reserved.

Upper levels can read information of lower levels, but not vice versa. Upper levels may not write into lower levels. Anything modified at a level gets the classification of that level. Each level forms the basis of a protection domain.

Mandatory Access Control (2)

- Privacy/confidentiality policies
 - compartmentalization



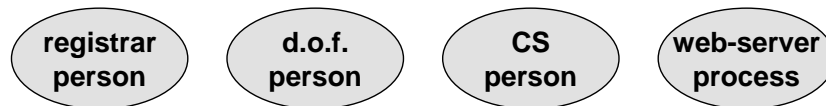
Another use of MAC is to enforce compartmentalization. For example, it might be Brown's policy that, for example, people working in the registrar's office have access to student records, but do not have access to faculty salaries. People working in the dean of the faculty's office do have access to faculty salaries, but do not have access to student records. This should continue to be the case even if someone switches jobs (but not computer IDs), moving from the registrar's office to the dean of the faculty's office. Note that this requires a notion of "role": one's role might change from being a registrar person to being a dean-of-the-faculty person.

Mandatory Access Control (3)

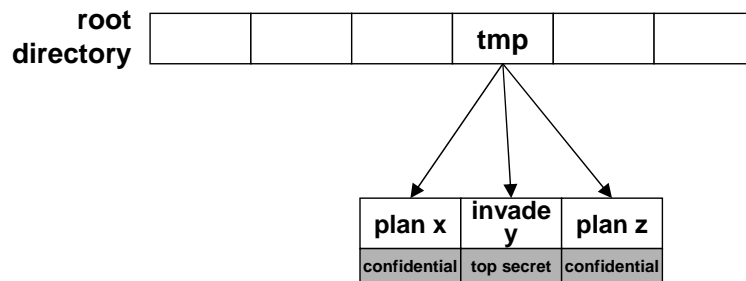
- **Local computer policy**
 - **web-server**
 - **may access only designated web-server data**
 - **administrators**
 - **may execute only administrative programs**
 - **(may not execute code supplied by ordinary users)**

Implementing MAC

- Label subjects and objects
- Security policy makes decisions based on labels and context

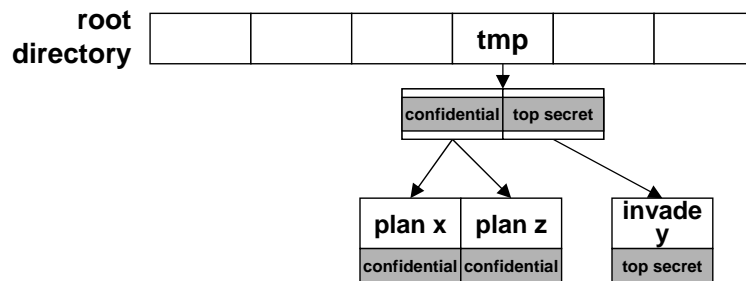


Multi-Level Directories (1)



That there is a file named “invade y” might be considered to be information that shouldn’t be made available to just anyone. However, it’s in a directory that accessible to just anyone. We might come up with an access-permission type that prohibits those without the necessary clearance from seeing the name of a directory entry, but what if someone cleared only for confidential tries to create the file “/tmp/invade y”?

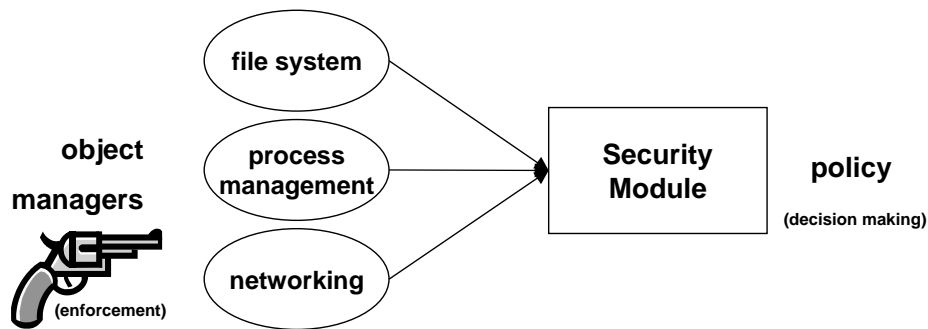
Multi-Level Directories (2)



The solution is to create an implicit subdirectory of `/tmp` with an entry for each classification. Thus if one is in the *top secret* domain, references to `/tmp` are actually to `/tmp/top secret`.

SELinux

- **Security-Enhanced Linux**
 - MAC-based security
 - labels on all subjects and objects
 - policy-specification language



CS 167

XXI-22

Copyright © 2006 Thomas W. Doepfner. All rights reserved.

A description of SELinux is given in the paper "Integrating Flexible Support for Security Policies into the Linux Operating System" by Peter Loscocco and Stephen Smalley in the Proceedings of the 2001 USENIX Annual Technical Conference (FREENIX '01), June 2001.

A Mandatory Security Policy

- “All programs and homeworks in CS167 must not be readable by anyone other than the owner and the group cs167ta”
 - implementable?
 - usably?

Orange Book

- **Evaluation criteria for secure systems**
 - **D: minimal protection**
 - **C: discretionary protection**
 - **C1: discretionary security protection**
 - **C2: controlled access protection**
 - **B: mandatory protection**
 - **B1: labeled security protection**
 - **B2: structured protection**
 - **B3: security domains**
 - **A: verified protection**
 - **A1: verified design**

The “Orange Book” (so-called because of the color of its cover) was released in 1985. Its actual title is “Department of Defense Trusted Computer System Evaluation Criteria” and is effectively a government standard on the security for standalone computer systems. Standard Unix and Windows systems, if properly set-up and administered, can achieve C2. SELinux might be able to achieve B1 or better (it must be officially evaluated first).