Mandatory Access Control

Computer Science Innovations, LLC

To Do Today

How am I doing?

Mandatory Access Control

Provenance

SQL Injection

Mandatory Access Control (Terms)

- Labeling
- Security Labels
- Continuous Protection
- Assurance
- Provenance

Why Do We Need MAC

As it turns out, we may use DAC to protect data on a need to know basis.

That means.... if all users are Top Secret and Top Secret is the highest level, so we have no Secret users, then we run the system at System High... So data differs by Need to Know.

What does this (DAC) look like?

System and all users are Secret – for example.

That being said, there is a group looking at Anthrax attacks.

There is another group looking at Olympic Security.

These are both Secret Issues, but do they need to know each other's data?

Security levels are the same, but need to knowing differs – DAC is your answer.

Knowing What You Know How for DAC?

We know, umasks, private groups, directory permissions. So what can we do.

We can formulate groups by categories of need to know. They use group and user level permissions to control.

Example

Redskin Fans – Scott, Cheri, Ernest, Ricardo

Haters – Elijah, Shawn, Al

Redskin Injury Report -

Pierre Garcon

Rams Injury Report

Sam Bradford

Belechik Principle – Hide Our Injuries

1) Users have private groups.
2) Supplemental Groups – Redskins, Haters
3) Redskin Injury Report in a Directory owned by Redskins and only they can read or write.
4) Rams Injury Report in a Directory owned by Rams and only they (Haters) can read or write.
5) Correct people in Each Group.

Security Adjudicators Say

1) If the Classification level is the same for all users (System High) and we vary only by need to know, they we may run at level C2 – which is discretionary access control.

2) People without need to know, but know are a risk.

From a Practical Standpoint – How?

Show you logging in to a web server and where this comes about.

Look at where we get these groups and how they are used.

For all security specifications, Groups and Roles are synonymous.

Setup a Session

Let's use the Google Example

Construct a Session

. Do an identity assertion which is what?
Username and password

Identity is Asserted

We have asserted the Identity then we do what?
Gather Roles.

We gather the roles. Is he an admin, what is he?

Roles are in Memory

User is valid..

Asserted Identity and gathered roles then we send a session id back to the Browser.

It stores this session id with the url

So we have www.google.com <id>

Roles ? Groups ?

A User (subject) may have many Roles and a Role may belong to many Users (subject)

Now we use this to implement need to know.

During Session Creation time the authorization module, reads all the Roles associated with a given user and loads them in memory. Now when you reference a User... you get their roles. Method Call... isInRole.

In Unix, What are our Roles?

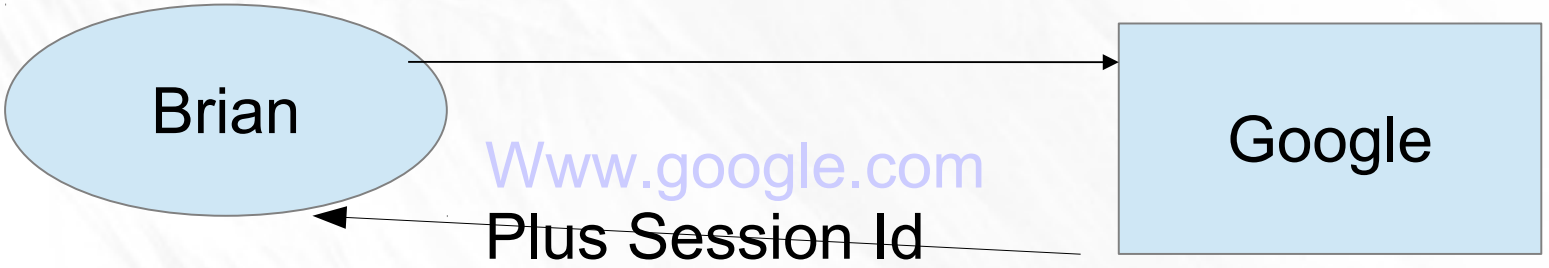
Groups... The same groups in /etc/group..

The same group in

Command chmod 2775

Next Request

It passes the session id <cookie> with the request



Provenance

Provenance, from the French provenir, "to come from", refers to the chronology of the ownership or location of a historical object

- Who, what, when, where, confidence and original source, security labels
- Weapons of Mass Destruction... not being in Iraq.

What Would Provenance Look Like

Make an Assertion....

Barack Obama is the 44th President of the United States....

Confidence = 100%

When – September 12, 2012

Security Label = Unclassified

Source = http://www.whitehouse.gov

Another Example

Assertion – Mitt Romney will be the 45th President of the United States on November 6, 2012

Confidence: .47

When: September 12, 2012

Security Label: Unclassfied

Source: rasmussenreports.com

Case Study

- In the intelligence community in early 2000's we could not differentiate conjecture from fact.
- So what happened, an analyst puts in something they believe with 20% certainty, and someone adds to it with 10% certainty. It is listed as a fact... What is the probability of it …. 2% accurate.
- If you do this enough, you get analysis that is bogus.

Solution.....

- We can no longer put any data in a computer system without Provenance.
- Marked with information including source and confidence.
- If I made the assertion that there is tyranny in Syria today... how would I mark this?
- Assertion, there is tyranny in Syria on June 26, 2012? Yes, believe this with 99% certainty, well known, source... mp4 file loaded on your web server.

MAC

Labeling – All data must be labeled.

Security Labels – You must show an adjudicator that the label is stored with the data. Therefore you cannot get to the data without getting to the label.

Tagging every piece of data.

Still some Issues with Labeling.

Issue for Labeling

So we have an employee record.... Let's say its Mo Walters.

His name, his address and his phone number (SECRET) but his salary (TOP SECRET). So at what label granularity do we use?

One choice is to label every piece of data at the column level … clearly to cumbersome. So we label the row at the highest level. This clearly is over-classifying. But this is how we deal with it.

Continuous Protection

Can I get to the data, without using the Access Control or Adjudication? You need to prove, that the only mechanism to get to the data is via the protection mechanism. Therefore we have Continuous Protection.

How?

Stored Procedure in a data base and the stored procedure (Adjudication) is the only way to get to the data. No Vendor Independence.

Aspect Oriented Programming (AOP). You may intercept calls. They are guaranteed to happen. What will this allow us to do. If we do it at the beginning of a call, we can audit. If we do it at the end, we can remove all data the user should not see.

Assurance

There is much written on Assurance in the Orange Book.

Mathematically prove Code.

This was a disaster... Why?

You are more likely to make a mistake in the proof than you are in the code. Useless.

Assurance

But the way we do this now is By Observation.

State your case, and prove it. So why does it work this way. At the end of the day it is a management decision.

MAC

Algorithm for MAC is:

- Flatten hierarchies
- If data is a subset of users roles (groups), you can see it.
- Cannot write below. No read up, no write down. Bell-Lapadula model.
- Implement it. We recognize MLS (MAC) in a RDMBS is difficult because we typically label at the row.
- Semantic Web... Google, Google Marketplace, Whole E-Commerce Industry. Provenance at the triple level.

Flatten Hierarchies

Orange Book says (Hierarchical and Non-Hierarchical). It means you have something like Mo is Top Secret (Hierarchy) and he is a US Citizen (Non-Hierarchical).

Mo's Roles Top Secret, Secret, Confidential, Unclassified, US Citizen.

We flatten the hierarchy, so we run one algorithm.

Bell-Lapadula Model

No Read Up.... No Write Down....

So If Mo is Top Secret, can he read Secret Data.... Read Down is ok.

As a Top Secret user can he take a Top Secret Document and write it down as Unclassified – No. Why No Write Down.

Let's Adjudicate

Mo - Top Secret, US Citizen.

Roles Top Secret, Secret, Confidential, Unclassified and US Citizen

Jason – Confidential, Unclassified, US Citizen

Data – One set of data. Troop Locations (Top Secret)

More Data – Budget for DOD Department

What Can They Read

Mo - Top Secret, Secret, Confidential, Unclassified and US Citizen

Troop Locations (Top Secret)

Budget for DOD Department

Jason – Confidential, Unclassified, US Citizen

Data – One set of data.

Budget for DOD Department

What Can They Write

Mo – He has to, as well as Jason, they have to work at one security level at a time. To Prevent Write Down. What we want to avoid is Mo – re characterizing troop locations as Unclassified.

Formally State It

Take All the Users Roles as a Set R.

Take all the data's Roles as a Set D.

If D is a Subset of R, you can read.

Best Practice?

- Never store data without Provenance.
- Keep a copy of the original source or a reference to it. The reference could always be found on archive.org.
- Store the Security Labels.
- Store who, what, when.
- Store the confidence.

Interpreted Languages

- Php you build the statement dynamically.
- Nothing other than check prohibits changing a value to more SQL
- SELECT * from Person where name = “<SOMETHING>”
- Can easily become – SELECT * from Person where name = 'abc' or '1' = '1'.
- *This would be injected!!!!*
- *Allowing us to see everything.*

Prepared Statements

PreparedStmt stmt = new PreparedStatement ("SELECT * from Person where name = ?");

- The value placed for ? Must be a literal.
- Later in the software you do a stmt.setValue(0, "abc").
- The value is interpreted as a Literal.
- Therefore no SQL Injection.

SQL Injection

- Cannot do SQL Injection on Compiled Statements that place literals.
- One Statement at a time.
- Cannot drop tables with it.
- Can see more that you are supposed to.
- Can update more than you are supposed to.

To Stop SQL Injection

- Use Compiled statements that handle literals.
- *Check for SQL Syntax in Interpreted languages.*
- *Can affect Update or Delete.*

Let's Do It

Have a lab for you. That we will do together.

Review SQL Syntax.

Play with the site.

Then, I am going to print out each statement you are going to run. So you can SQL INJECT (not for REAL).

Then do it for Real.

SQL

Structure Query Language – ANSI 1989.

DML – Data Modification Language (Includes Reads)

SELECT, INSERT, UPDATE and DELETE

SELECT <columns> FROM <table> WHERE col <boolean_operator> <value>

SELECT * from USERS WHERE id = 5

SELECT id from USERS where name = 'Mo'

Lab 3.1

1) go to http://10.10.10.243/admin

Just play the with application

2) Try to read something you should not see. So I will, not run the SQL, but print out the command it would run. Plan your SQL Injection Strategy.

3) Inject and see something you would not normally see.

Lab 3.2

1) go to http://10.10.10.243/admin

And go to
http://10.10.10.243/admin/searchMembers2.php

2) Use searchMember2.php to see statement and /admin to see the results of the query.

3) Inject and see something you would not normally see.

Buffer Overflow

- In the 70's and 60's for that matter Operating Systems were written in assembler and therefore ran on distinct hardware.
- In the Late 70's AT&T (Kernighan, Ritchie, Thompson) create as Operating System to run on any hardware (Unix).
- To do this, they need a computer programming language to run on any hardware. They create the 'C' programming language.
- Business users wanted their software to run on any hardware so they programmed in 'C'.

Buffer Overflow (Continued)

- C was designed to go low level. It was a replacement for Assembler..
- Using 'C' in a business context meant you could overwrite buffers or buffer overflow.
- This no longer occurs for what reason?
- Programming languages now are virtualized meaning you cannot go beyond memory bounds.
- Therefore no …. Buffer Overflow.

Avoiding Buffer Overflow

- Be born after 1970.
- Use fully virtualized languages Java, Php, C++, etc. etc.
- The only parts of the system written in 'C' are the operating system something you cannot touch in an application program. Period.

Get vs. Post

- Web page pushes information to the sever is a Post

<form>

<input type=”text”>

</form>

Get is http://host?val=a&val2=b

- Parameters passed on the url
- Gets and Posts are the same thing.

Access Control in Unix

We wish to limit access to a Unix system.

Obvious methods?

Username/password

Public key/private key

What else? TCP Wrappers!

Restrict by IP address

- 192.168.1.150
- Or even IP address range
- 192.
- 204.158.

TCP Wrappers

Where do we accomplish this?

The */etc/hosts.allow* and */etc/hosts.deny* files

Traditional safe setup

hosts.deny

ALL:ALL

hosts.allow

List all allowable IP addresses or IP address ranges.

This will deny access to all IP addresses that aren't explicitly granted access.

This access control is done at the kernel level.

There's no circumventing this

TCP Wrappers

/etc/hosts.allow

/etc/hosts.deny

Now you take hosts.deny

ALL:ALL

Then you take hosts.allow

ALL: 192.168.1.

You do not have to use all

sshd: 192.

Gotcha!!! Big One

TCP Wrappers does a reverse DNS lookup.

If you have in the hosts.allow file

ALL: www.scottstreit.com

This will not work. Why? Take www.scottstreit.com and tell me the IP address now take that IP Address and tell me the authoritative server name. FIOS,

Work Around

Use IP Address.

Or be the authoritative name.

You are mail.scottstreit.com www.scottstreit.com

Wiki.scottstreit.com

In hosts.allow .scottstreit.com

Syntax is SERVICE: WHO

Lab 3.3

Take your backtrack instance

In /etc/hosts.deny ALL: ALL

In /etc/hosts.allow ALL: an ip address

Restrict one of your neighbors and allow one in.

Use a regular account.

Lab - Scenario

Your employer decides to restrict computers by MAC address.

So, to exist on the network, you must fill out a form and wait for an Admin to put you in the MAC address table.

Security? Secure? Not Secure? Annoying. Useful?

Not secure. See MAC address – unencrypted. Spoof... You are on.

IDS – look for more than one DHCP or DNS server.

Lab – Scenario - Answer

Your employer decides to restrict computers by MAC address.

So, to exist on the network, you must fill out a form and wait for an Admin to put you in the MAC address table.

Security? Secure? Not Secure? Annoying. Useful?

Not secure. See MAC address – unencrypted. Spoof... You are on.

IDS – look for more than one DHCP or DNS server.

Review

- SQL Injection
- Fingerprinting
- SQL Injection – Cannot do it with pre-compiled statements. Such as Java Prepared Statements
- Occurs in interpreted programming languages such as Php, perl, Python
- Test to make sure Literals are no SQL Syntax