Lecture 4 - 09/17/2019

Identity

Example: File in Unix
- UID
- Login Name

When a file must be known to several systems:
- “A url”

When a computer must be known to several systems:
- Hostname or IP

When an Identity must be known to several systems:
- Hard problem to solve

Identity Mapping

- A way to map identities across different systems.

Globally Unique Identifiers

Hierarchical name space
- Want a scememe that is globally unique but manageable in practice
  - Do what we do naturally: divide and conquer
- The organization can manage all of their sub-identities

Flat name space
- Single authority manages a table of identities
- Responsible for all names of all people
  - E.G. usernames, student numbers

ISO X.500 - DistinguishedName

The DistinguishedName was adopted by X.509 as the naming scheme used in identifying certificates (although now it’s a little more open). Not Bulletproof: a tree can be duplicated and a person can be spoofed in a second tree. However public and private keys can verify that a tree is the original and valid tree

The amount of information needed depends on the scale of the environment you want to work on.

Final Points

Naming authority and Authentication Authority are different. One controls the name, one assigns the name to an ID.

Authentication

How to claim an identity.

Provision of information so that system can confirm identity
Bind a principal to a representation of identity internal of the computer

Information provided by the principal:

- Falls into classes
    - What you know
    - What you have
    - What you are
    - [Where you are]

Five components of authentication systems:

A: Authentication Information (Password, fingerprint)
C: Complementary information (Information that's stored by the system)
F: Function that maps one to the other (Hash fn, feature extraction algorithm)
L: Login functions (Compares what you give with what it has, gives T/F)
S: Selection functions (Enrollment procedures and PW reset)

Local vs Remote Authentication:

- Distinction is whether authentication information needs to travel over the internet.

Security Tools

Examples: Security Policies, Access Control, Cryptography, DB Security, OS Security.

Security Policy

Says whats good/bad or allowed/not allowed. This is built from our threat model.We build our system starting from our security policy. A secure system starts in an authorized state and cannot enter an unauthorized mode.Model/define system as a finite state machine (FSM).

Security policies can be written with respect to confidentiality, integrity, or availability. (There are also hybrid policies; some mixture).

Confidentiality:

- Identifying information that flows or leaks to opele that are not allowed to receive it.
- “I have data that you shouldn't see, if you do that is getting into an insecure state”
- We have to account for dynamic changes:
    - E.g. People used to have access and don't anymore.

Integrity:

- Not concerned with the flow of information
- Concerned with changes to the information in travel.
- Separation of duties: you must have two separate entities for verification

Availability:

- Defining what services need to be available and to what parameters.
- “We have to be able to accept this many logins per second”
- “People can request web pages but not script files”

Policies can be written formally or informally:
When policies are written ‘informally’ its written in natural language. Easier to read and write, but harder to implement.

When policies are written ‘formally’ it’s written in a more symbolic or math based language. Math language are much easier to write proofs for.

Security policies rest on assumptions and we have to understand what those assumptions are.

E.g. Procedure: *Install a security patch as soon as it's available*.

System administrator receives a security patch for the OS and installs it. Has security been improved?

Assumptions:

- Did it come from the official vendor
- Did the vendor test it properly.
- Did they test is on MY environment or a similar one to mine?
- Did the sysadmin install it properly?

Example Policies:

Bell-LaPadula Model (“Secure computer systems; Mathematical Functions”. MiTRE Tech. Report, March 1973)

- Most famous and influential confidentiality policy
- Built around military-style security labels.
    - Subjects have security “Clearances”
    - Objects have security classifications.
- Realized there could be read write errors
- *-property:
    - A subject can write an object iff the subject's level is less than or equal to the objects level. (Read down; write up)
- Can constrain read/writes based on conditions (Time of day, Location, e.t.c)
- Let (L,C) = Classification label + set of codewords
  = Security category
- (L,C) dominates (L’,C’) iff L > L’ and C > C’
- S can write to O if S dominates O and S has write access to O

Let Σ be a system with secure initial state ε and let T be a set of state transitions. If every element of T preserves the security condition and the *-property, thenevrye state &, i > 0 is secure.