Lecture 12: Computer Security.

1. Chubby follow-up
2. Security in the real world
    a. Why this paper?
        i. Came from a sequence of papers about reasoning about trust in distributed systems
        ii. Picked this one because it was shortest, easiest to read -- should I include full versions in future?
        iii. Abstracts out specifics of authentication protocols (SSL, Kerberos, DAV, etc.)
        iv. Covers high-level issues
    b. Comments from reviews:
        i. Need to start with trust?
        ii. non-impact in the world, not an academic paper
        iii. What is novelty?
            1. The idea of using "speaking for" to organize and prove things and reason about authentication/authorization
            2. E.g. Windows didn't limit what IDs domains could speak about
            3. E.g. on-line shopping basket APIs don't reason about what a request can speak about, and hence can reuse secure channel to do the wrong thing
            4. E.g. Car jacking case from yesterday – couldn't reason about anything
    c. Outline:
        i. Security in the non-virtual world
        ii. What are the components of security
        iii. What are the differences between single-system security and distributed system security
        iv. Framework for understanding and assuring authentication/authorization
3. Non-virtual security
    a. How does it work?
        i. Perfect locks? aka strong security
        ii. Constant surveillance? aka constant logging/intrusion detection
    b. How it really works?
        i. Locks on our houses
        ii. Police to track down crimes
        iii. Value of the crime
        iv. Together:
            1. Locks have to be strong enough to deter criminals

            2. Risk of punishment must be high enough to deter criminals
            3. So:
                a. High value items (e.g. jewels): strong locks, high surveillance, high punishment
                b. Low value items (e.g. shovel in my garage): no locks, little punishment
                c. Credit cards: easy to steal/forge, but high effort to track down and limit damage (little loss, lots of surveillance)
4. Computer security
    a. From paper:
        i. Attacks come from anywhere
            1. Not just physically local
        ii. People want to share with anyone
            1. Not just known friends
        iii. Automated infection
            1. Grows faster
        iv. Hostile code
            1. Running others code on your system
        v. Hostile environment (taking a laptop to China)
        vi. Hostile hosts
            1. Sending your data to other people's system or running your code on their system
    b. What is different?
        i. Economy of scale: one attack can be automated at many nodes
            1. Effort to break in may be low (weak locks)
        ii. Difficult to prosecute: criminals may be overseas
        iii. Lots of low value items
            1. A few dollars from a bank account
    c. Is this always true?
        i. NO: stuxnet, military computers
            1. High value, strong locks, strong surveillance
            2. High cost of prosecution
5. Basic principles of computer security:
    a. Core components:
        i. Policy: what you want to allow/disallow, and by whom/when/where
        ii. Mechanism: things that enforce policy
        iii. Assurance: how you know your policy is being enforced by the mechanism
    b. Policies:

        i. Secrecy: controlling who reads information (e.g. Coke formula).
        ii. Integrity: controlling who modifies information (e.g. your bank balance or grades)
        iii. Availability: provide prompt access (e.g. Denial of Service attacks)
        iv. Accountability: logging who had access to what for post-attack diagnosis
        v. Components:
            1. Who is allowed, when, where, under what circumstances
                a. E.g. two tellers together during working hours can withdraw money
            2. Who is not allowed

c. Mechanisms
    i. Who are we defending against?
        1. Bad (buggy or hostile) programs
            a. e.g. downloading malicious app
        2. Bad agents giving instructions to good but gullible programs
            a. e.g. stack smashing attack on Apache
        3. Agents tapping/spoofing communications
            a. e.g. sniff packets, inject packets, modify packets
    ii. Defensive strategies: levels of control/isolation
        1. Complete isolation – air gap physical security
            a. Used for very-high value. E.g. private keys used to sign Microsoft Windows binaries
        2. Keep bad agents out
            a. Firewall: block bad things from getting in
            b. Code signing: only run code from trusted sources
            c. Trust everything on the inside with full access
        3. Limit damage of bad agents
            a. Sandboxing to limit operations available
            b. Limit programs invokable from sandbox, system calls made, etc.
        4. Catch the bad agents
            a. Run intrusion detection/audit logs and figure out who they are
            b. Difficult, but used
    iii. Basic mechanism: access control
        1. A "principle" requests to do an operation

a. A principle is any identifiable agent, e.g. a user, a system, a program

2. A "reference monitor" decides based on the reques and the principle whether to grant access to a resource
3. Example:
    a. Access a file: have write & write w/o quota requests
    b. Access control list says all users can write, some can bypass quotas (separate requests)

iv. Authentication vs Authorization
1. Authentication: identifying who a principle is
2. Authorization: deciding what they can do – are they allowed to do an operation
3. Auditing: record the decisions of the reference monitor

d. Assurance ideas:

i. Core concept: **trusted computing base**
1. All the software/hardware that has to work correctly to enforce policy
2. Examples:
    a. firewall to keep bad guys out at perimeter
    b. **QUESTION:** what is TCB for file system?
        i. File system, kernel, configuration data, password file, setuid root files for file system security

ii. Use TCB idea:
1. Make it small – more likely to be secure
2. Defense in depth – multiple layers of defenses
    a. firewall
    b. authentication
    c. ACLs in application

6. Reasoning about security

a. Local access control:

i. Assume channel for user requests (system calls) is secure

ii. System has local database of names, passwords, and IDs

iii. User authenticates by logging in to compare password against database and lookup ID

iv. Programs can have identities (**setuid)**

v. Resources (files) have ACLs with principles & permissions

b. Extending this to an organization

i. Move database to another machine (e.g. Kerberos key distribution center, Windows domain controller/Active Directory server)

ii. Machine sets up a **secure channel** to check password against server
  1. server provides a **token** that vouches for user identity & provides IDs
     a. e.g. Kerberos ticket granting ticket

c. Extending to multiple organizations
  i. Key requirement: want multiple domains of users
    1. Allows multiple entities to vouch for password -> username/ID mapping
       a. Assign names within a domain
  ii. How:
    1. Secure channel between domains
       a. User logon information sent to home domain, which vouches for user
       b. Home domain uses secure channel to resource domain vouching for user identity
       c. Resource domain communicates to resource host user identity
       d. Resource host uses user identity to decide access control
  iii. Requirements:
    1. Limit scope of what a domain can vouch for
       a. Don't want servers at Michigan to vouch for swift@cs.wisc.edu or serve IDs in our domain

7. Reasoning about distributed security
   a. Why can we do this? Consider SSL authenticated logon to a web server at Wisconsin
   b. Working backwards:
      i. Request comes over an SSL connection encrypted with a key K_{ssl}
      ii. SSL connection created by a smart card K_{alice} signing a challenge from a Michigan server
      iii. A Michigan server certifies that K_{alice} is for alice@umich.edu
      iv. A Wisconsin AFS server certifies that alice@umich.edu is in the group Architects
      v. the ACL on a file Affiliates in AFS says the group Architects has read/write access to the file
   c. So: lots of different components. Why should we believe Alice has access?
      i. Session keys
      ii. User passwords/public/private keys
      iii. Authentication across domains (wiscsonsin and Michigan)
      iv. Group memberships
      v. ACL entries

8. Core concepts:
    a. **Speaking**: making a statement
        i. An ACL says "user X has access to the file"
        ii. A key can say "the holder of the key encrypted /sent this message"
        iii. A database can say "The user that encrypts with a key is named …"
        iv. Also called a token:
            1. X.509 certificate: Kca says Kserver → Server-name
            2. Group membership: Database says ID user → ID group
        v. How used:
            1. If signed by a public key, can be forwarded to anyone and used offline & verified by anyone
            2. If encrypted with a secret key, can ask sender to re-encrypt and send to anyone with whom sender shares a secret key
                a. e.g. ask KDC to re-encrypt a ticket for a specific server
            3. If sent on secure channel (see below), then cannot be forwarded
    b. **speaks for** (the happens-before of computer security?)
        i. Basic usage:
            1. "Principle P speaks for principle Q about subjects T"
                a. P→Q T
            2. Meanings:
                a. If P says something about T, then Q says (or would say) the same thing about T
                b. Q takes responsibility for what Q says about T
                    i. Q trusts P to speak on its behalf
                c. P is more powerful than Q – P's statements are taken as seriously as Q's
            3. Subjects:
                a. Allows scoping of "speaks for"
                b. Example:
                    i. Michigan servers can only speak about users at Michigan
                    ii. File ACLs can only speak about file permissions
        ii. Longer example: the chain above
            1. Kssl → Kalice → Alice@umich.edu → architects → affiliates
                a. The SSL key can speak on behalf of Alice's key (by the protocol)

            b. Alice's key speaks on behalf of the user alice@umich.edu (because certified by the server at Michigan)
            c. alice@umich.edu speaks for architects (by AFS group database)
            d. architects speak for the file Affiliates (by the ACL)
        iii. Uses:
            1. Key speaks for a username
            2. Program hash speaks for the program
            3. Certifying authority speaks for a group of names
                a. Verisign can speak for mapping of keys to DNS top-level domains
                b. A principles speaks for any name below it:
                    i. wisc.edu speaks for cs.wisc.edu
                    ii. So: if wisc says a key speaks for cs.wisc.edu, then it means cs.wisc.edu would say the key speaks for cs.wisc.edu
            4. User can speak for a group it belongs to
            5. Group can speak for a resource if in ACL
    c. **Secure Channels**
        i. Messages encrypted with a key
        ii. Assume the channel speaks for the key
9. How do we establish the links exist?
    a. Why do we trust the principle
        i. If Q says P→Q about T, we believe P→Q about T
            1. Principles can delegate their own authority
    b. How do we know who says the delegation
        i. The only ways to know **directly** who says something
            1. Receive it over a secure channel (know the key that encrypted it)
                a. If Q is a key, then Q says
        ii. Access it from a local database
            1. If Q is a guard (verifier), then Q can consult a local database (e.g. user database for a key or ACL for a file)
        iii. Otherwise: user inference to build a chain
            1. A →B and B → then A → C
            2. For mapping of a process to a file:
                a. Process → user → group membership → file
                b. Capability = direct mapping process → file
                    i. bypasses inference chains (e.g. login, key verification, ACL lookup)
    c. Why willing?

        i. Some facts installed manually
            1. Root certificates for SSL
            2. Name/password for domain controller in Windows
        ii. Some from protocol:
            1. If I run SSL, then I authorize the SSL channel to speak for me

10. How do we do inference
    a. **Push**: client generates information and sends to server
        i. e.g. Kerberos login to get ticket, user name, group memberships, sent to server which decrypts ticket
    b. **Pull**: Client requests, object queries database
        i. **e**.g. access control lsits

11. So: Authentication
    a. Provide to a server that a message speaks for a user
        i. Convince that the key encrypting the message speaks for the user
        ii. Key encrypting message speaks for users key
        iii. Users key speaks for user name
        iv. domain server speaks for user name
        v. e.g.
    b. Notice: need local root of trust
        i. Need to trust that domain server speaks for user name

12. Example: Kerberos login
    a. AS-req = username -- -- > AS-rep = {TGT }Kuser
    b. TGS-req = TGT, host -- -- > TGS-req = {host, user,Ksess}Khost
    c. AP-req = {host,user,Ksess}Khost, {login}Ksess
    d. Khost speaks for TGS says Ksess speaks for User
    e.

13. QUESTIONS:
    a. How handling naming without hierarchy?
        i. Basically have to give up on human-readable names being unique, because collisions are unavoidable
        ii. Use keys instead (See SPKI, SDSI)
    b.