Agenda

- Social Engineering
- Whaling
- New access control paradigms
- DRM
- The Cloud
Social Engineering

- Social life is ruled by customs, culture and tradition
- By manipulating these rules we perform Social Engineering
- We attack the human behind the machine
https://www.youtube.com/watch?v=lc7scxvKQ0o
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
2. Imagined benefit
3. Trust
4. Moral responsibility
5. Guilt
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
   - “This is already approved, go ahead and let me in.”
2. Imagined benefit
3. Trust
4. Moral responsibility
5. Guilt
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
2. Imagined benefit
   - The victim believes the action will benefit him or her
   - “The supervisor wants you to do it”
3. Trust
4. Moral responsibility
5. Guilt
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
2. Imagined benefit
3. Trust
   - The victim innocently wants to be helpful.
4. Moral responsibility
5. Guilt
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
2. Imagined benefit
3. Trust
4. Moral responsibility
   - The victim is led to believe that it is the “right thing to do”
   - Perhaps the action will help fix some imagined problem.
5. Guilt
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
2. Imagined benefit
3. Trust
4. Moral responsibility
5. Guilt
   - The victim is led to believe that he or she has done something wrong
   - The attacker then says that taking this action will correct the problem
What social vulnerabilities do we exploit?

There are several reasons why social engineering works

1. Unclear responsibility
2. Imagined benefit
3. Trust
4. Moral responsibility
5. Guilt

People want to help!
Social Engineering: Going to the top!

- Tuesday-Thursday: Conference in Gothenburg
- At 11:10, an email is received by our administrator:

  Från: Jan-Åke Larsson [mailto:jan-ake.larsson@liu.se]
  Ämne: SV: omedelbar betalning(13:12:16)

  Hej,

  Kan du göra en banköverföring till Storbritannien idag?

  Best Regards...
  Jan-Åke Larsson

- Body of e-mail includes a picture of Jan-Åke
- At 12:32, our administrator *forwards* the email to the economy department and Jan-Åke
Whaling

- This e-mail was a whaling scam (also called CEO fraud)
- Carefully researched and well-planned, it’s a “spear-phishing” scam taken to a new height
- Often impersonating executives and management (JÅ is the head of ISY), it attempts to scam money off the administrative department
- According to the FBI, over 2.3 billion USD have been lost since 2013.
- Yes, this is now also done against LiU...
It did end well, though

- There was a Reply-to: field set, so replies would go to the scammer
- The email was forwarded, not replied to
- Jan-Åke discovered the scam within 20 minutes and immediately alerted the staff
- Nothing happened. But it could have!
Mitigation

- Policy
- Awareness
- Technology
Mitigation

- Policy
  - Personal example: “Never send passport number by email”
  - Corporate example: “Never give a password to anybody else”
  - But no policy is good without infrastructure

- Awareness

- Technology
Mitigation

- Policy
- Awareness
  - One must be aware that social engineering exists
  - Knowing the common signs of attack
  - Needs to cover all employees
- Technology
Mitigation

- Policy
- Awareness
- Technology
  - Email signatures
  - Authentication
New access control paradigms

- The standard access control setting is that a user wants access to resources on a server
- On the web, this has changed
- There aren’t any simple requests like read or write, but rather entire programs are sent from one end to the other to be executed there
- Clients execute scripts from servers
- Servers execute scripts from clients
The old paradigm suits closed organizations

- The old paradigm had principals, actions, and objects
- Principals need to be authenticated
- The security policy is used to authorize access of a principal to an object
- The underlying implicit assumption is that principals are well-known people

- This fits well in a closed organization
- The organization has authority over the users
- Users can be held accountable
- Security policy refers naturally to users, and user identities
The old paradigm suits closed organizations

- Rules are stored locally, in ACLs
- Enforcement is centralized
- Permissions are simple (rwx)
- Security policy is determined by one organization
- System managers are in charge
- Users do not participate in systems administration
The web is an open organization

- The web has lots of executable content
- Computation moves to the client, freeing resources at the server
- The client is now asked to make security policy decisions
- The browser becomes part of the trusted system
- There aren’t any fundamentally new problems, but the usual access control paradigm seems unsuitable
Revised terminology

• Don’t ask “who made this request?”
• Rather ask “what evidence do I have that this is a legitimate request?”
• Here, we either have authorization without authentication, or authentication must gain a new meaning
• *Authentication* would now mean verification of any evidence provided
  • From the external environment, like signatures, origin, …
  • From associated local data like session data, or even date and time
Revised terminology II

• Don’t ask “is this principal allowed to access the object?”
• Rather ask “what evidence is needed to be allowed access?”
• The above question is what is referred to as *authorization* in this context
• But to determine whether access is granted or not, you need to ask “is this evidence enough to be allowed access?”
• This is called *approval*, in our new terminology
Revised terminology III

- The singular Reference monitor used to be in charge of authentication and authorization

- A Policy Administration Point (PAP) creates the policy

- A Policy Decision Point (PDP) interprets the policy and makes authorization decisions

- A Policy Enforcement Point (PEP) performs access control, sending its decisions to a PDP and enforces received decisions

- A Policy Information Point (PIP) is a source of evidence
Example: XACML policy evaluation

The eXtensible Access Control Markup Language is intended to describe security policy in a large heterogenous environment.
Example: the Simple Public Key Infrastructure (SPKI)

- This is intended to work without user identities
- User identities only have a localized meaning anyway
- Access rights are likely connected to other attributes than identity
- Principals in SPKI are instead public keys (or their hashes)
- These are unique, with very high probability
- In localized name spaces, there may still be a need to connect a specific key to an identity
- This is done by creating a “Name certificate” signed with the private key
Example: the Simple Public Key Infrastructure (SPKI)

- Access rights are bound to a key, also by creating an “Authorization certificate”
- These contain the issuer, the subject, authorizations, delegation rights, and validity conditions
- Creating such a certificate is called “empowerment” in SPKI terminology
- Delegation is possible, but only for rights that you already hold

- The trust chain originates in a root ACL that must be set up locally
Trust management

- In heterogenous environments, even the process of finding granted and required credentials is a problem of its own
- Finding the appropriate third parties (PIPs) can also be a problem
- This is handled through trust management
- Here, a trust relationship may not consider trust in the usual sense, but rather a business relation
- In a federated environment, each organization may have their own security policy
- There must be a general agreement in the federation on how to resolve conflicts between different policies
Example: XACML policy evaluation

The eXtensible Access Control Markup Language is intended to describe security policy in a large heterogenous environment.
Code-based access control

- On the internet you are dealing with people who are essentially unknown to you
- Their “identity” cannot be a basis for access decisions
- Their physical location is not available or useful information
- You have no authority over them, nor can you hold them accountable
- Even if you did know the identity, a legal process would be slow, cumbersome, expensive, or even impossible

- User identity cannot be used as basis for access control
- Similarly, the source of a request is not useful, on its own
Code-based access control, or Evidence-based access control

Evidence in access control decisions can be

- **Code origin**  Local or remote code? Which URL? Which zone?
- **Code signature**  Is the code signed? By who? Here, “trust” may only need a decision to run code by a given author, not that you “trust” him or her
- **Code identity**  Has the code been approved by you? Is its hash value (=ID) correct?
- **Code proof**  Code can come with a “proof” of certain security properties, but this is more a research problem, than used in real code
Code-based access control, sub-component call

- In code-based access control, different components have different associated trust.
- Here “trust” should be read as “needs these permission in its task” rather than being trustworthy.
- When one software component calls another, the effective permissions of the current process will change.
- We need to resolve differences in trust level between the two components.
Code-based access control, stack inspection

- In the Java Virtual Machine, and in .NET Common Language Runtime, an extended call stack is used to keep track of permissions.
- Each frame on the stack holds the local state, including the permissions of that particular software component.
- When a permission is needed by a component, the entire stack is taken into account through a stack walk.
- Lazy evaluation is the simplest possible combination, the intersection of the permissions of all components in the stack.
Code-based access control, assertion

- Sometimes a component really needs a specific permission to do its job
- And the stack walk may prevent this, for example if a higher-level component lacks the permission
- In that case, there usually is a possibility to force a permission for the component, to assert the permission
- But take care, because this means that a call to the component gives access, even to an untrusted calling component
- Build in defenses when you assert permissions
Code-based access control, stack optimizations

- Using the stack for security evaluation changes the requirements on the stack
- For example, a common optimization is *tail call elimination*
- When a component is called at the end of the calling component, the frame of the calling component is usually replaced by the new frame
- But this now removes the permissions in that frame, alternatively, removes restrictions in that frame
- In *eager evaluation* (the counterpart of lazy), the callers’ rights are also tracked separately from the stack, as the *current permissions*
The Java 1 security model

- Simple policy: sandbox for unsigned applets, full access for local code and signed applets
- Inflexible, there is no intermediate security level
- Location and signature are the only sources of evidence of trust
The Java 2 security model

- Byte code verifier checks
  - Class file format
  - Stack does not overflow
  - Appropriate types in method calls
  - Type conversion
  - Class references
  - Static access restrictions, e.g., method private data

- Class loader
The Java 2 security model

- Byte code verifier
- Class loader
  - Loads the class
  - Assigns a protection domain
  - Link-time checks of type safety
- Security manager
Java security policy

- A *protection domain* is assigned to each class at load time
- This contains code source, principal, a reference to class loader, static permissions, and load-time dynamical permissions
- Protection domains are static

- Permissions is granted to code, and is required to access a resource
- There are only positive permissions
- There is a permission class hierarchy
- Entries that have not yet been needed are marked UnresolvedPermission
The Java 2 security model

- Byte code verifier
- Class loader
- Security manager
  - Performs access control
  - Uses the *execution context* including an execution stack walk
  - It uses lazy evaluation, extended with assertion of permissions (and some more extensions)
The .NET security model

- The .NET Common Language Runtime is the corresponding thing to JVM
- Several programming languages are supported (C#, VB, managed C++, VJ#, ...), all are translated into MicroSoft Intermediate Language, corresponding to Java byte code
- A logical unit of MSIL code is called an assembly
- There is also provisions for native code, compiled machine language for specific hardware
- But these are outside the access control of the CLR
Evidence in .NET

- Examples are
  - Site of origin
  - Security zone
  - Hash value
  - Authenticode (MS) signature
  - “Strong name” signature
  - Application-specific evidence

- The last *assembly-provided* evidence can never override the preceding *host-provided* evidence types

- Code is divided into *app domains* that is used to separate executing code
“Strong names” in .NET

• A “Strong name” is a public/private key pair
• But there is no trust chain, no certificate
• A strong name is not proof of identity, but rather evidence that the different parts of an assembly are from the same source

• In .NET, authenticode signatures more correspond to the traditional picture, as generated by a named entity
Permissions in .NET

• The usual set of permissions, but can be negative: “this code can never …”
• Assertion of permissions can be permitted
• An added permission is that of accessing unmanaged code
• It is also possible to skip verification altogether

• Instead of protection domains (as in Java), .NET assemblies uses code groups
• Each assembly can belong to several code groups, but exclusive code groups prohibits this
• Code groups are arranged in hierarchies
Stack walk in .NET

• Basically lazy evaluation, but with several extensions
• Has assertion of permissions
• But also can represent anti-assertion (Deny)
• There are also short-range extensions
  • Evidence in the form of a hash value gives one method access permission
  • If that method calls another, access is denied
  • The Link Demand action checks the calling method for access, but does not perform a full stack walk
Digital Rights Management

- DRM enforces access control on a client machine, set by an outside party
- Another departure from traditional access control
- The threat is no longer a third party, but the recipient
- The system owner may try to bypass the security policy
Computer security problem?

- CIA?
- Confidentiality: Content should be available to buyers only, but buyers won’t take on extra trouble to protect the interests of others
- Integrity: Your creation should not be distorted and misused
- Availability: Precautions to limit availability for non-buyers should not hamper legal availability
Tools

- Trusted hardware + Cryptographic protection
- Some tools need online checking
- Use a trusted platform module that runs an attestation protocol
- Some use smartcards containing cryptographic keys
- Software protection will never work for attacks from computer enthusiasts, period
Availability

- Limitations against copying should not limit the availability for paying customers
- Availability only on-line gives problems with network access requirements
- Copy protection gives problems with backups
- Encryption gives problems with key management
Distribution, encrypted products

- Use one “session key” per film, TV programme, TV channel, tune or album etc.
- Distribute the session key encrypted with the personal key of each customer
- How does one revoke keys?
- There is an efficiency problem in large customer groups
- The keys could possibly be copied
- There are also privacy issues if customer need two-way access
- Customers are sometimes locked to one “player”
DRM hasn’t really been successful

- Does anybody recognize 09 F9 11 02 9D 74 E3 5B D8 41 56 C5 63 56 88 C0?
DRM hasn’t really been successful

• Does anybody recognize 09 F9 11 02 9D 74 E3 5B D8 41 56 C5 63 56 88 C0?

• This is the master key for the now-defunct HD-DVD video system

• The AACS tried to remove mentions of this key under the DMCA

• On May 1, 2007, Digg started removing posts and closing accounts mentioning the key
The cat was already out of the bag

• This backfired horribly, with the Digg front page being flooded with references to “09 F9”
The cat was already out of the bag

- This backfired horribly, with the Digg front page being flooded with references to “09 F9”

This flag contains the first fifteen bytes of the key as RGB-encoded colors, with C0 added to the end.
Detection, how?

- **Watermarking**: Embed code that proves origin, unapproved copies are illegal
- **Fingerprinting**: Embed a *unique* code in each sold copy, and register who bought that copy, unapproved copies can be traced to original buyer
The future of DRM

- There is an ongoing debate about the usefulness of DRM
- Some argue that pirates steal
- Others argue that copying actually increase sales
- Content providers also use the legal system to prohibit piracy
- It is now illegal in some countries to reverse-engineer DRM tech
The Cloud

- Location-independent resource pooling
- Rapid elasticity
- Ubiquitous net access
- On-demand self-service
- Measured services

- Avoid start-up costs
- Reduce operating costs
- Increase agility
The Cloud

- **Infrastructure as a service (IaaS)**
  - Provider supplies virtualized machines and storage
- **Platform as a service (PaaS)**
  - Provider supplies programming environments and building blocks
- **Software as a service (SaaS)**
  - Provider supplies applications

- Public clouds
- Private clouds
- Community clouds
- Hybrid clouds
Shared responsibility

• In SaaS, provided services have a large number of integrated features and are typically not extended by the client. Providers are responsible for the security of the services, more so in public than in private clouds.

• In PaaS, the client develops applications on top of the provided platform. The client is responsible for protecting the application, but lower layers are up to the provider, like isolating the clients’ applications from one another.

• In IaaS, the client also supplies the OS. The provider must still provide basic, low-level data protection.
Service-level agreements

• In cloud computing Service Level Agreements are necessary, they record a common understanding about services, priorities, responsibilities, guarantees, and warranties

• A new layer is needed to support contract negotiation, and to monitor contract enforcement

• The dynamic nature of the cloud makes continuous monitoring necessary

• Consumers might not completely trust measurements from a single provider; perhaps an agreed-upon third party is needed
Compliance and Regulations

- The client should monitor that the service complies with the SLA, their own security policy, external regulations, and also with legal requirements.
- Cloud computing is global in its nature, so multiple jurisdictions can come into effect.
- Unfortunately, legal requirements vary, and sometimes even are in conflict with each other.
Heterogeneity

- Cloud providers may use different hardware and software platforms to build their cloud
- A tenant may use SaaS from one provider, PaaS from another, and IaaS from a third
- The protection requirement for each tenant of a single provider may differ
- Each tenant may have different trust relations with the provider
Authentication and identity management I

- So far, password-based authentication is the standard
- This is seen as one of the most significant risks of cloud computing
- Especially, if the provider has a simple mechanism to export all data (which is often sought by tenants)
- This is now a single point of failure.
Authentication and identity management II

- Heterogeneity is another problem; different identity tokens and protocols are used simultaneously
- A front-end service may need to keep identity info protected with some of the services it interacts with
- How multi-tenant multi-provider cloud environments affect privacy of identity information isn’t yet well understood
Access control and accounting

- Services are heterogeneous and diverse
- Dynamic context-, attribute-, or credential-based authorization
- There are also privacy-protection requirements

- The cloud needs proper user and service accounting
- Tenants do not want a provider to maintain detailed accounting records other than for billing purposes
- The out-sourcing and multi-tenancy aspects may be an additional factor
Security in a dynamic environment

• For access control, (extensions of) Role-Based Access Control are used, but this is work in progress
• In a cloud, roles are dynamic, and not always well defined, so *role mining* is sometimes used
• Security policy is most efficient when a centralized approach is possible
• But the dynamic and decentralized nature of the cloud requires more flexible solutions
• Cross-domain frameworks are specified, verified, and enforced using Security Assertion ML (SAML) and eXtensible Access Control ML (XACML)
Privacy and data protection

- Many organizations are not comfortable storing their data off-site
- Private information does face increased risk in a shared infrastructure
- Cryptographic tools are being considered as solutions to this problem but privacy protection and outsorced computation need more attention
- A related issue is accountability, as it relates to changes in data
- The information needed to track changes, audit, and perhaps perform history-based access control is now a privacy issue
- Balancing between this and privacy is a challenge
Example attack: Deduplication

- A common method to save bandwidth is *deduplication*
- Enterprise clients often store multiple copies of identical or similar data
- Private customers do so too; common software manuals or media files are often already present at the provider
- If the provider can identify these before upload, upload can be avoided, saving bandwidth
- Most vendors do not try to hide deduplication
- You can find this either in the upload logs, the upload status message, the upload time, the upload speed, or network traffic
- Examples: DropBox, MozyHome, Memopal, ...
Example attack I: Identifying files through deduplication

• Assume an attacker wants to find out if a client has stored a file at a cloud provider
• All the attacker needs to do is to attempt storing the file, if deduplication occurs, then yes: some client has stored the file
• If the file is unique for some user, then that user has stored the file
• If some law enforcement authority attempts to upload some file with illegal content and deduplication occurs, then there are users that have uploaded it
• If deduplication does not occur, then the upload should be aborted
• The law enforcement authority can now obtain a court order for the cloud provider to identify the users
Example attack II: Learning file content through deduplication

- Assume an attacker wants to find the salary figure on some client’s employee’s contract
- The attacker can generate contracts in the client’s name with different salary figures; the one deduplicated contains the actual salary
- If a bank sends a document with customer name and PIN and this gets stored in a cloud service, the same approach can be used
- 10000 PINs can be tried, without alerting the bank
- This can be used to find passwords, results of medical tests, auction bidding status, ...
Example attack III: A covert channel through deduplication

- Assume an attacker succeeds in installing malicious software on a client’s computer but cannot communicate anymore with the computer, say, because of a firewall
- The malware is such that it generates a file at the client in one of two versions, $F_0$ and $F_1$
- The files should be sufficiently random so that they are unique
- When the target user runs a backup, the file gets uploaded to the cloud provider
- The attacker can now check for presence, and the result is one bit of communication from the target to the attacker, despite all the security measures
- This can be reversed to communicate to the target system
The Cloud

- **Infrastructure as a service**
  - Provider supplies virtualized machines and storage
- **Platform as a service**
  - Provider supplies programming environments and building blocks
- **Software as a service**
  - Provider supplies applications

- **Public clouds**
- **Private clouds**
- **Community clouds**
- **Hybrid clouds**
Where to go from here?

- TSIT03 Cryptology (HT1, ISY)
- TDDC90 Software Security (HT2)
- TDDD17 Information Security, Second Course (VT1-VT2)
Course reviews

- After the course, you will be asked to give your opinion via Studentportalen.
- We very much value your feedback!
- ...especially free-text answers. This will help improve next year’s course.
Jonathan Jogenfors

www.liu.se