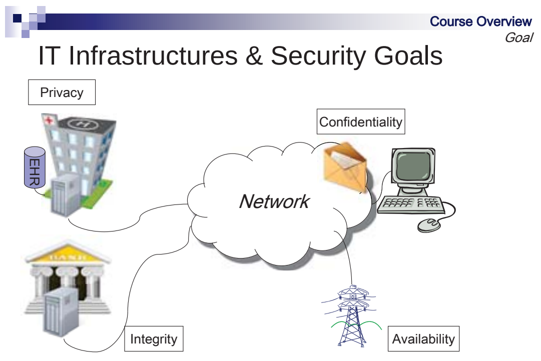
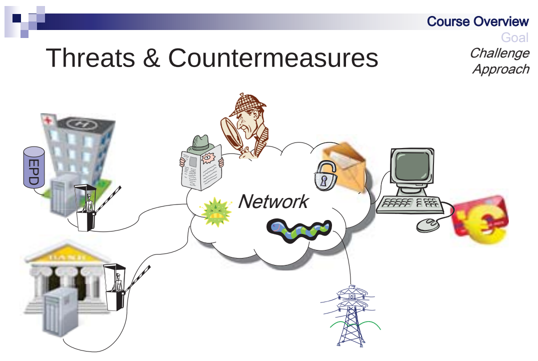
1 Introduction

1.1 Course structure and timing





IT infrastructures are faced with threats to its key security goals. This course aims to give an overview and general understanding of security goals, threats and countermeasures. After the course you should be able to (1) recognize and describe security requirements of scenarios (2) identify security technologies that can help alleviate security issues (3) combine these technologies in a basic security design.

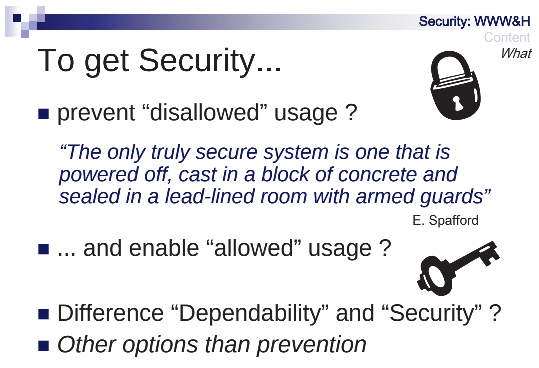
Week	Session	Date	Topic
1	1A	06-02-2013	Introduction
	1B	08-02-2013	Cryptography (1)
	Lab 1	08-02-2013	Web of Trust creation
2	2A	20-02-2013	Cryptography (2)
	2B	22-02-2013	Network security Basics
	Lab 2	22-02-2013	HTTP Basics, Sniffing and Tampering
3	3A	27-02-2013	Malware, Web security - SQL, XSS
	3B	01-03-2013	Hashes, Certificates, etc.
	Lab 3	01-03-2013	SQL and XSS
4	4A	06-03-2013	Access Control
	4B	08-03-2013	Digital Rights Management
	Lab 4	08-03-2013	Access Control and session information stealing
5	5A	13-03-2013	Authentication (1)
	5B	15-03-2013	Authentication (2)
	Lab 5	15-03-2013	Authentication Flaws, Password cracking
6	6A	20-03-2013	Security Protocols
	6B	22-03-2013	Side channel attack, security protocol exercises
	Lab 6	22-03-2013	Session Fixation/Stealing - Phishing
7	7A	27-03-2013	Privacy and Anonymity
	7B	03-04-2013	Discussion Exercises and Exam preparation.
			(Full lecture - no lab session this week).

1.2 The What-When-Why and How of Security

1.2.1 The What (and When) of Security

“Is your system secure?” What does this question actually mean; does it mean that nobody but you can use it; can throw it out the window; can keep you from using it...?

Security attributes capture goals that one may want to achieve to call a system ‘secure’. The most commonly used and widely accepted security attributes are *Confidentiality*, i.e. ‘my information stays secret’, *Integrity*, i.e. ‘my information stays correct’, and *Availability*, i.e. ‘I can get at my information’. Of course these concepts can also refer to resources or system aspects other than ‘information’.

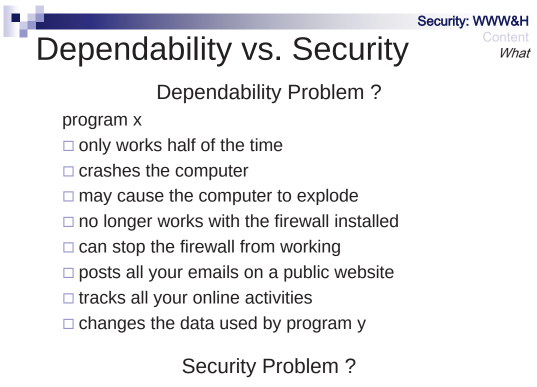


Security: WWW&H
Content
What

To get Security...

- prevent “disallowed” usage ?

  “The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards”
  E. Spafford
- ... and enable “allowed” usage ?
- Difference “Dependability” and “Security” ?
- *Other options than prevention*



Security: WWW&H
Content
What

Dependability vs. Security

Dependability Problem ?

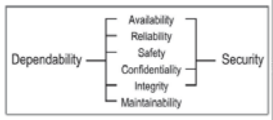
program x
- ☐ only works half of the time
- ☐ crashes the computer
- ☐ may cause the computer to explode
- ☐ no longer works with the firewall installed
- ☐ can stop the firewall from working
- ☐ posts all your emails on a public website
- ☐ tracks all your online activities
- ☐ changes the data used by program y

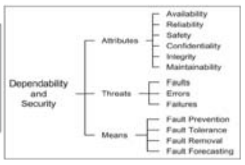
Security Problem ?

The security attributes of the system may be at *risk* from several types of *threats*. Besides the usual problem such as program errors and system failures, security also needs to address malicious entities which are specifically trying to break the system. This leads to the security means, i.e. the security tools applied to achieve the security attributes in face of these risks; we will come back to this in the ‘How of Security’ section (Section 1.2.3) below.

The `What’ of security - Security Attributes
Confidentiality
Integrity
Availability

Dependability vs. Security (2)





Basic Concepts and Taxonomy of Dependable and Secure Computing
IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING

The `What’ of security - Security Attributes
C-I-A
Privacy

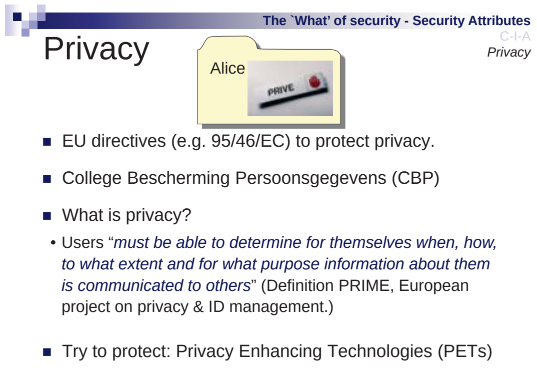
Privacy Online

Peter Steiner 1993

"ON THE INTERNET NOBODY KNOWS YOU'RE A DOG..."
HI ! WELCOME TO BUTCHER NET
TO DOWNLOAD A LEAN MEAT RECIPE CLICK HERE...
TO GO TO THE MARROW BONE PAGE CLICK HERE...
YOU'RE A DOG, AREN'T YOU ?

Nik Scott 2008

In addition to Confidentiality, Integrity and Availability (‘C-I-A’) other security attributes are sometimes formulated. Closely related but not usually called a security attribute is *Privacy*, i.e. ‘is information about me not misused’. Note the difference between Confidentiality and Privacy: where confidentiality requires data that you possess to remain secret, privacy deals with data about you that may be in the hands of others. Where ‘who gets the data’ is a key question in confidentiality, the purpose for which data is used is a key ingredient for privacy.

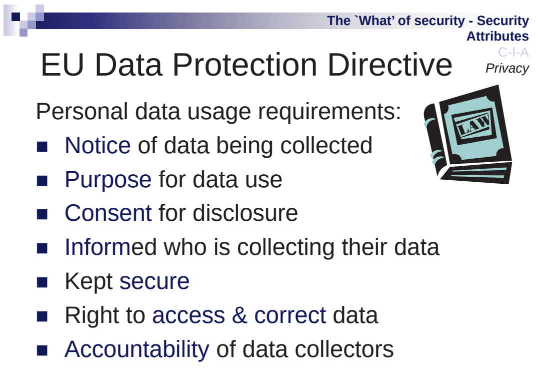


The `What' of security - Security Attributes
C-I-A
Privacy

Privacy

Alice
PRIVE

- EU directives (e.g. 95/46/EC) to protect privacy.
- College Bescherming Persoonsgegevens (CBP)
- What is privacy?
  - Users "*must be able to determine for themselves when, how, to what extent and for what purpose information about them is communicated to others*" (Definition PRIME, European project on privacy & ID management.)
- Try to protect: Privacy Enhancing Technologies (PETs)



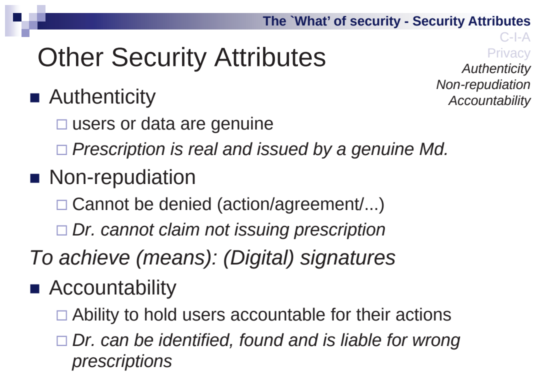
The `What' of security - Security Attributes
C-I-A
Privacy

EU Data Protection Directive

Personal data usage requirements:

- Notice of data being collected
- Purpose for data use
- Consent for disclosure
- Informed who is collecting their data
- Kept secure
- Right to access & correct data
- Accountability of data collectors

Other examples of security attributes are: *Authenticity*, i.e. 'is this information authentic', *Non-repudiation*, i.e. 'is this information undeniable' and *Accountability*. Authenticity is different from integrity in that, it does not focus on alteration of data but on the correctness of the data; is the original source of the information 'correct' (what ever that may mean in the setting where the notion is used). A signature on a contract would be an example of a way to achieve non-repudiation; you cannot later deny agreeing to the conditions in the contract. The relation between accountability and non-repudidation is similar to that between integrity and authenticity; non-repudiation can be an important part of achieving accountability but is by itself not yet sufficient.



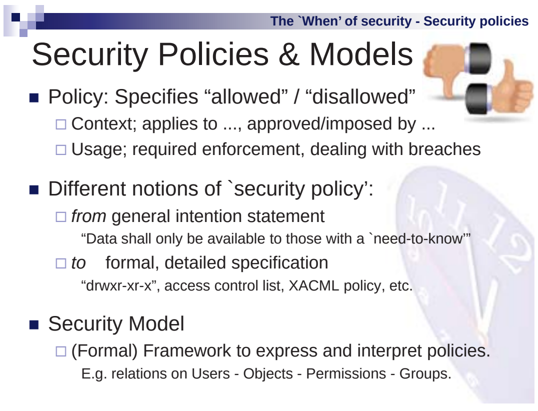
The `What' of security - Security Attributes
C-I-A
Privacy
Authenticity
Non-repudiation
Accountability

Other Security Attributes

- Authenticity
  - users or data are genuine
  - *Prescription is real and issued by a genuine Md.*
- Non-repudiation
  - Cannot be denied (action/agreement/...)
  - *Dr. cannot claim not issuing prescription*

To achieve (means): (Digital) signatures

- Accountability
  - Ability to hold users accountable for their actions
  - *Dr. can be identified, found and is liable for wrong prescriptions*



The `When' of security - Security policies

Security Policies & Models

- Policy: Specifies "allowed" / "disallowed"
  - Context; applies to ..., approved/imposed by ...
  - Usage; required enforcement, dealing with breaches
- Different notions of `security policy':
  - *from* general intention statement
    "Data shall only be available to those with a `need-to-know'"
  - *to* formal, detailed specification
    "drwxr-xr-x", access control list, XACML policy, etc.
- Security Model
  - (Formal) Framework to express and interpret policies.
    E.g. relations on Users - Objects - Permissions - Groups.

The *security policy* describes when what security attribute needs to be achieved. Note that the term security policy is used differently in different settings. It may be anything from a high level textual description meant to be understood and applied by human beings, e.g. "all personal identifiable information must only be read when needed to provide a service" to low level computer readable information e.g. "drwxr-xr-x"[1]. Translating high level policies into a systems design along with low level policies is an important step of creating a secure system.

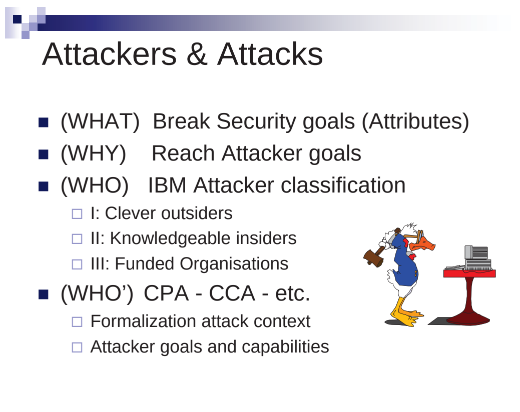
The exact meaning of a security policies can be given within a *security model*; a (formal) framework to express and interpret policies. For example the unix file permission given above can be interpreted as a relation between Users, Groups, Objects and Permissions; An object has an owner and group (an additional part of the security policy) and the owner of the object (a directory) has read,write and execute permission, members of the group as well as other users have read and execute permission.

1.2.2 The Why of Security

Achieving reliability of a system is already a difficult task. To achieve security one does not only have to deal with unintentional errors but also account for intentional attacks on the system. Every

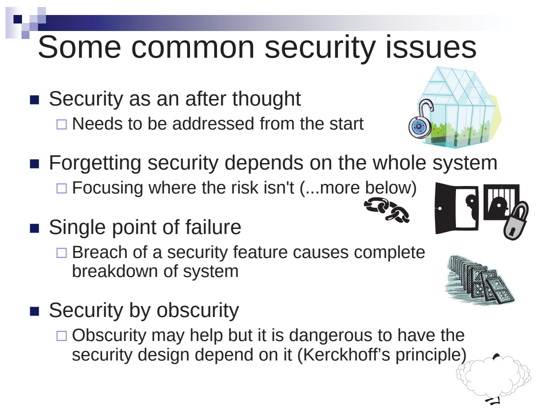
[1]A unix style file permission setting

day seems to bring new security incidents where attackers are able to exploit security weaknesses. Although this may give an skewed perspective; a system remaining secure yet another day will not make the news, it does show the importance of considering who may wish to attack the system and why; we need an *attacker model*. This attacker model captures the capabilities and intentions of an attacker (see also Section **??** on requirement engineering below) similar to the security model describes the meaning of a security policy, i.e. the intentions of the designers/users of the system.



Attackers & Attacks

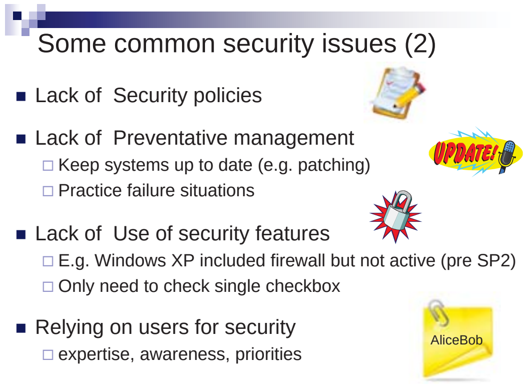
- (WHAT) Break Security goals (Attributes)
- (WHY) Reach Attacker goals
- (WHO) IBM Attacker classification
  - I: Clever outsiders
  - II: Knowledgeable insiders
  - III: Funded Organisations
- (WHO') CPA - CCA - etc.
  - Formalization attack context
  - Attacker goals and capabilities



Some common security issues

- Security as an after thought
  - Needs to be addressed from the start
- Forgetting security depends on the whole system
  - Focusing where the risk isn't (...more below)
- Single point of failure
  - Breach of a security feature causes complete breakdown of system
- Security by obscurity
  - Obscurity may help but it is dangerous to have the security design depend on it (Kerckhoff's principle)

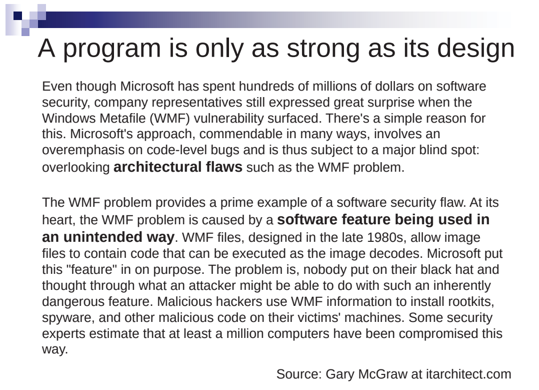
Attacker models can be general; such as IBMs classification of attackers into three categories (Clever outsiders, Knowledgeable insiders and Funded Organisations) or formal such as those used in analysis of cryptographic algorihtms (e.g. Chosen-Plaintext-Attack (CPA) where the attacker is able to get encryptions of plain text she has chosen.) Any security analysis will need both the security goals (attributes/policy) and an attacker model. Sometimes these are left implicit but they remain key ingredients; the question 'is this system secure?' has no meaning without them. Not properly considering them is a common cause of security problems. Several other common causes of security problem are often related to this.



Some common security issues (2)

- Lack of Security policies
- Lack of Preventative management
  - Keep systems up to date (e.g. patching)
  - Practice failure situations
- Lack of Use of security features
  - E.g. Windows XP included firewall but not active (pre SP2)
  - Only need to check single checkbox
- Relying on users for security
  - expertise, awareness, priorities

UPDATE!
AliceBob



A program is only as strong as its design

Even though Microsoft has spent hundreds of millions of dollars on software security, company representatives still expressed great surprise when the Windows Metafile (WMF) vulnerability surfaced. There's a simple reason for this. Microsoft's approach, commendable in many ways, involves an overemphasis on code-level bugs and is thus subject to a major blind spot: overlooking **architectural flaws** such as the WMF problem.

The WMF problem provides a prime example of a software security flaw. At its heart, the WMF problem is caused by a **software feature being used in an unintended way**. WMF files, designed in the late 1980s, allow image files to contain code that can be executed as the image decodes. Microsoft put this "feature" in on purpose. The problem is, nobody put on their black hat and thought through what an attacker might be able to do with such an inherently dangerous feature. Malicious hackers use WMF information to install rootkits, spyware, and other malicious code on their victims' machines. Some security experts estimate that at least a million computers have been compromised this way.

Source: Gary McGraw at itarchitect.com

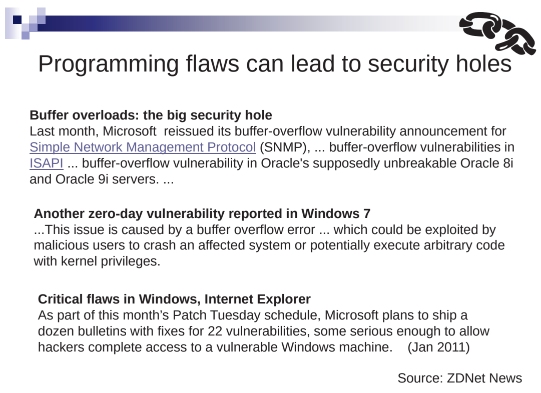
A chain is no stronger than its weakest link. This is also the case for the security of a system. Consider for example the following aspect of a systems and some potential issues.

Design There is no hope of having a secure system if the system design does not address security goals or worse has inherent features/goals with imply security problems. As examples consider the Windows Meta File (WMF) where arbitrary code execution, a clear security risk, is a design feature.

As another example one can consider the internet; initially the internet linked a group of trusted systems and now very important security goals were not consideration in its design, e.g. any computer can claim to have an IP, no authentication of DNS, etc. (See also lecture on network security). No protection of content. Of course the are currently security mechanisms (IPsec, HTTPS, etc.) that try to remedy this but 'add on security' is always problematic - security needs to be considered from the start.

Software quality A perfect design does not help if the implementation is flawed. Often security issues are caused by software bugs with buffer overflow vulnerabilities being one of the major issues. In a buffer overflow attacks input from an untrusted source is written into a buffer without the bounds of the buffer being checked. This causes the untrusted data to be written to places it is not supposed to go; it may overwrite a return address on the stack, causing a jump to an attacker selected location at the end of the current routine. See e.g. [7] for details on buffer overflows, and an analysis of tools to prevent them.

The problem of software bugs is not solved easily; e.g. recently an unsolved buffer overflows vulnerabilities was reported in Windows 7 and in January 2011 Microsoft ships fixes for 22 vulnerabilities. Note that software and systems evolves; it is not the case that each round of patching brings us closer to a final secure and bug free system.



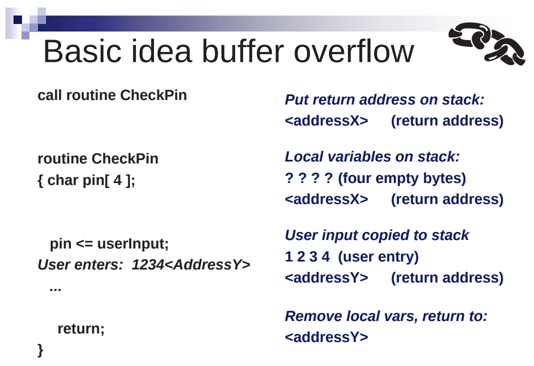
Programming flaws can lead to security holes

Buffer overloads: the big security hole
Last month, Microsoft reissued its buffer-overflow vulnerability announcement for Simple Network Management Protocol (SNMP), ... buffer-overflow vulnerabilities in ISAPI ... buffer-overflow vulnerability in Oracle's supposedly unbreakable Oracle 8i and Oracle 9i servers. ...

Another zero-day vulnerability reported in Windows 7
...This issue is caused by a buffer overflow error ... which could be exploited by malicious users to crash an affected system or potentially execute arbitrary code with kernel privileges.

Critical flaws in Windows, Internet Explorer
As part of this month's Patch Tuesday schedule, Microsoft plans to ship a dozen bulletins with fixes for 22 vulnerabilities, some serious enough to allow hackers complete access to a vulnerable Windows machine. (Jan 2011)

Source: ZDNet News



Basic idea buffer overflow

```
call routine CheckPin


routine CheckPin
{ char pin[ 4 ];


  pin <= userInput;
User enters:  1234<AddressY>
  ...

   return;
}
```

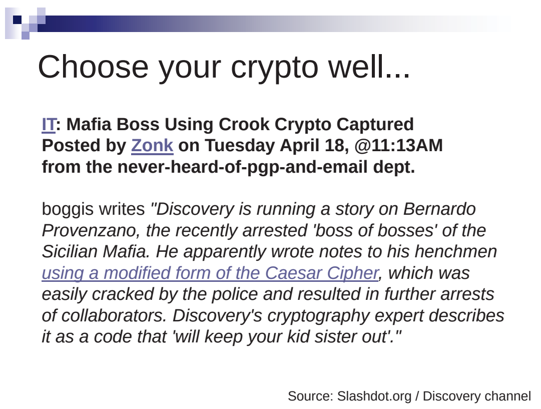
Put return address on stack:
<addressX> (return address)

Local variables on stack:
? ? ? ? (four empty bytes)
<addressX> (return address)

User input copied to stack
1 2 3 4 (user entry)
<addressY> (return address)

Remove local vars, return to:
<addressY>

Security Tool Selection Choose your crypto well, especially if you are a mafia boss. "...wrote notes to his henchmen using a modified form of the Caesar Cipher, ... a code that 'will keep your kid sister out'." Clearly here the selected security tool was grossly insufficient to reach the security goal. This is an extreme example but often inappropriate security tools are used of tools are used well past their 'best before/replace by' date such as the hash function MD5 (see also the lecture on hash functions) which has been known to be vulnerable for a long time but is only slowly being phased out. Using 'home-made' crypto solutions instead of tried and proven standard algorithms would also fit in this category. (e.g. leave design of crypto to the experts; obscurity of a design is not a good replacement their experience and expertise.)

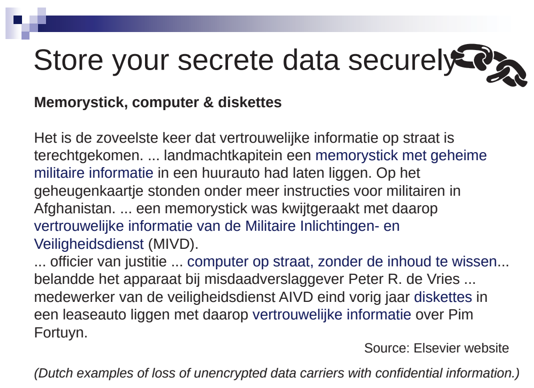


Choose your crypto well...

IT: Mafia Boss Using Crook Crypto Captured
Posted by Zonk on Tuesday April 18, @11:13AM
from the never-heard-of-pgp-and-email dept.

boggis writes *"Discovery is running a story on Bernardo Provenzano, the recently arrested 'boss of bosses' of the Sicilian Mafia. He apparently wrote notes to his henchmen using a modified form of the Caesar Cipher, which was easily cracked by the police and resulted in further arrests of collaborators. Discovery's cryptography expert describes it as a code that 'will keep your kid sister out'."*

Source: Slashdot.org / Discovery channel



Store your secrete data securely

Memorystick, computer & diskettes

Het is de zoveelste keer dat vertrouwelijke informatie op straat is terechtgekomen. ... landmachtkapitein een memorystick met geheime militaire informatie in een huurauto had laten liggen. Op het geheugenkaartje stonden onder meer instructies voor militairen in Afghanistan. ... een memorystick was kwijtgeraakt met daarop vertrouwelijke informatie van de Militaire Inlichtingen- en Veiligheidsdienst (MIVD).
... officier van justitie ... computer op straat, zonder de inhoud te wissen... belandde het apparaat bij misdaadverslaggever Peter R. de Vries ... medewerker van de veiligheidsdienst AIVD eind vorig jaar diskettes in een leaseauto liggen met daarop vertrouwelijke informatie over Pim Fortuyn.

Source: Elsevier website

(Dutch examples of loss of unencrypted data carriers with confidential information.)

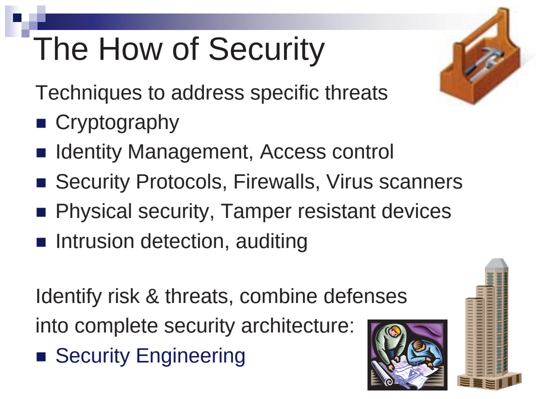
System usage Even a perfectly designed and implemented system (should one ever by created) is of no use if it is not used correctly. USB data sticks that offer encryption of their content are readily available and company policy may state that such sticks should be used. However, if the user does not enable this feature this is all for naught.

Users have different priorities; e.g. ease of use; and many not use security features or will even try to work around them if they interfere with what they are trying to do.

Of course these are only examples and there are many more aspects of a system where a weak link in the security chain may occur. Key points are that one needs to consider the system as a whole and consider security from the start.

1.2.3 The How of Security

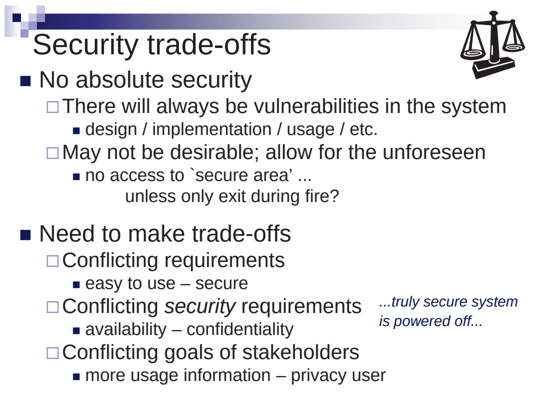
We have already seen some security tools (means) above and during the course we will try to add key tools to this toolbox for the security engineer. Crytography is an important part of this toolbox. However recall that security tools by them selfs do not make the system secure. A common claim 'the data is secure because it is encrypted' is by itself meaningless and may even indicate that the security goals and attacker model have not been sufficiently considered. For instance encryption offers no protection against inside attackers who have access to the key. A good security design is needed which addressed to know what security tools need to be employed where and when. We need to do *Security Engineering*. For the tools we will treat in the remainder of the course always try to consider how they fit in a bigger design.



The How of Security

Techniques to address specific threats
- Cryptography
- Identity Management, Access control
- Security Protocols, Firewalls, Virus scanners
- Physical security, Tamper resistant devices
- Intrusion detection, auditing

Identify risk & threats, combine defenses into complete security architecture:
- Security Engineering



Security trade-offs

- No absolute security
  - There will always be vulnerabilities in the system
    - design / implementation / usage / etc.
  - May not be desirable; allow for the unforeseen
    - no access to `secure area' ...
      unless only exit during fire?
- Need to make trade-offs
  - Conflicting requirements
    - easy to use – secure
  - Conflicting *security* requirements
    - availability – confidentiality
  - Conflicting goals of stakeholders
    - more usage information – privacy user

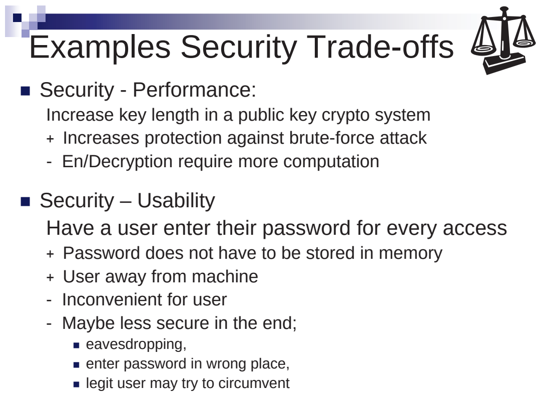
...truly secure system is powered off...

1.2.3.1 Trade-offs

Recall this quote we saw earlier:

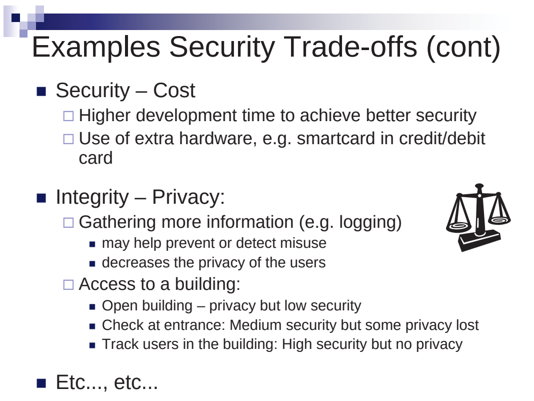
> *"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards"*
>
> E. Spafford

Such a system may be secure but not very useful. (Actually it may not be secure at all - Which security attribute is clearly not satisfied? - without the security goals we cannot answer this question...) There is often a clear trade-off between security and usability (why do I need to remember that password...), performance (e.g. using encryption adds computation time) and costs (e.g. replacing pin cards and readers by smartcard enabled versions) but also between different security attributes e.g. confidentiality and availability. Which trade-offs are worth while; e.g. how much security do we gain for the performance we give up?



Examples Security Trade-offs

- Security - Performance:
  Increase key length in a public key crypto system
  + Increases protection against brute-force attack
  - En/Decryption require more computation
- Security – Usability
  Have a user enter their password for every access
  + Password does not have to be stored in memory
  + User away from machine
  - Inconvenient for user
  - Maybe less secure in the end;
    - eavesdropping,
    - enter password in wrong place,
    - legit user may try to circumvent

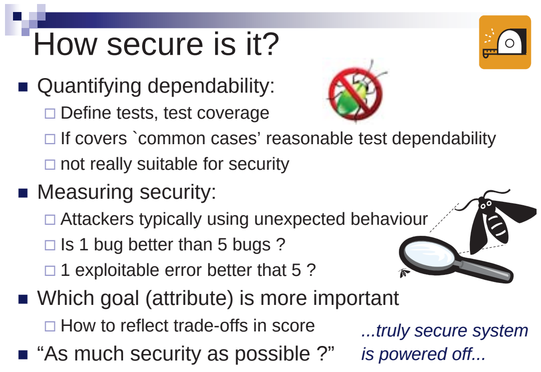


Examples Security Trade-offs (cont)

- Security – Cost
  - Higher development time to achieve better security
  - Use of extra hardware, e.g. smartcard in credit/debit card
- Integrity – Privacy:
  - Gathering more information (e.g. logging)
    - may help prevent or detect misuse
    - decreases the privacy of the users
  - Access to a building:
    - Open building – privacy but low security
    - Check at entrance: Medium security but some privacy lost
    - Track users in the building: High security but no privacy
- Etc..., etc...

1.2.3.2 Measuring and Selling security

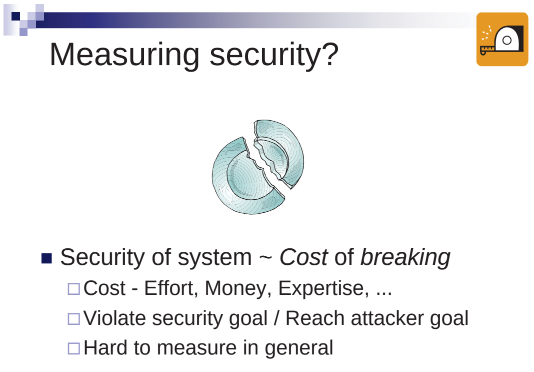
Why does security often not get the attention it needs? For one; if it's good you don't see it. Would you pay €50 more for say a television if it was more secure. (Does your answer depend on 'how much' more secure?).



How secure is it?

- Quantifying dependability:
  - Define tests, test coverage
  - If covers `common cases' reasonable test dependability
  - not really suitable for security
- Measuring security:
  - Attackers typically using unexpected behaviour
  - Is 1 bug better than 5 bugs ?
  - 1 exploitable error better that 5 ?
- Which goal (attribute) is more important
  - How to reflect trade-offs in score
- "As much security as possible ?"

...truly secure system is powered off...



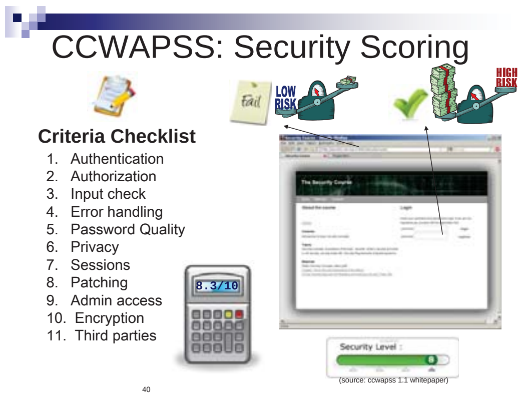
Measuring security?

- Security of system ~ *Cost of breaking*
  - Cost - Effort, Money, Expertise, ...
  - Violate security goal / Reach attacker goal
  - Hard to measure in general

It is also hard to quantify security. You can say that a 'product is 2x faster' and have every consumer have some notion of why and how much better the product, even though even this statement is usually much more complex that it seems. However, what does 'this product is 2x more secure' mean? There are many discussions on which product is more secure e.g. comparisons between windows and linux, firefox and windows explorer, mac and pc, etc. Claims are supported by quoting the number of bugs/vulnerabilities reported, the number security incidents, etc. But how well do any of these really reflect the 'security' of a system. Thinking back to the earlier discussion about what is 'security of a system' one can see that no single number could really adequately capture this. Still, what quantification is possible? If we try to focus our attention on a single aspect of security/single application area one may be able to give some numbers that make sense (just remember that, the more general the statement the more less objective a score is likely to be).

For cryptographic primitives one can look at the (computational) cost of breaking a system. This is often expressed by the entropy that it offers in a given setting e.g. 'this crypto system offers 80-bits of security' reflects that the amount of computation needed to break it is similar to brute-forcing an 80bits key, i.e. trying 2^{80} different possibilities. This is generalized to a measure for security of systems by considering the cost (computational or otherwise) of breaking the system to be its security. (e.g. it would take 2 years and a budget of 10 MEuro to break this system.)

For web applications several security metrics have been defined by checking for common security issues and assigning a risk to each of them. E.g. the CCWAPSS Common criteria for web application security scoring [3] computes a score based on a list of eleven criteria which have to be checked (rating the web service on a scale of 1 to 3 for each item) and assigned a risk level based on difficulty and impact of an attack.



CCWAPSS: Security Scoring

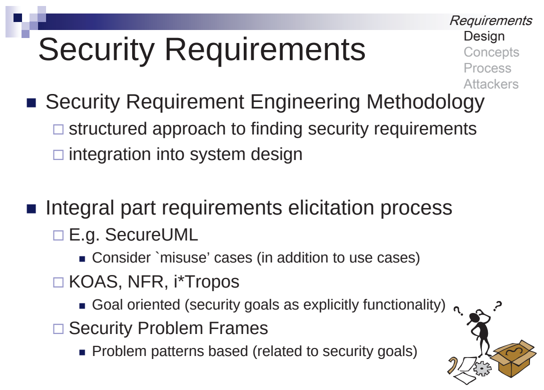
Criteria Checklist

1. Authentication
2. Authorization
3. Input check
4. Error handling
5. Password Quality
6. Privacy
7. Sessions
8. Patching
9. Admin access
10. Encryption
11. Third parties

Fail
LOW RISK
HIGH RISK
8.3/10
Security Level :

(source: ccwapss 1.1 whitepaper)

40



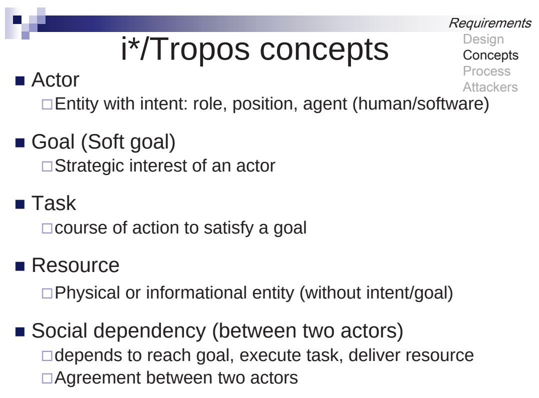
Security Requirements

Requirements
Design
Concepts
Process
Attackers

- Security Requirement Engineering Methodology
  - structured approach to finding security requirements
  - integration into system design
- Integral part requirements elicitation process
  - E.g. SecureUML
    - Consider `misuse' cases (in addition to use cases)
  - KOAS, NFR, i*Tropos
    - Goal oriented (security goals as explicitly functionality)
  - Security Problem Frames
    - Problem patterns based (related to security goals)

1.2.3.3 Security Requirement Engineering

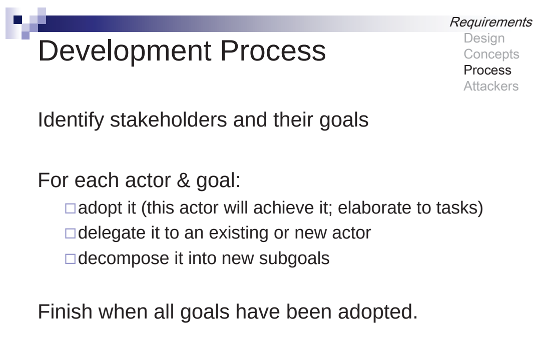
As already mentioned several times, and as will be repeated often again, to really evaluate the security of a system you have to consider it as a whole, know the security goals and the potential threats against these goals. To gather these we need to perform *Security Requirement Engineering*. Throughout the design, implementation, deployment and use of a system we should consider the requirements that the users will have of the system and how attackers will try to exploit the system. Based on this we can come up with and/or evaluate a security design which combines several security solutions to achieve the best possible trade-offs.



i*/Tropos concepts

Requirements
Design
Concepts
Process
Attackers

- Actor
  - Entity with intent: role, position, agent (human/software)
- Goal (Soft goal)
  - Strategic interest of an actor
- Task
  - course of action to satisfy a goal
- Resource
  - Physical or informational entity (without intent/goal)
- Social dependency (between two actors)
  - depends to reach goal, execute task, deliver resource
  - Agreement between two actors



Development Process

Requirements
Design
Concepts
Process
Attackers

Identify stakeholders and their goals

For each actor & goal:

- adopt it (this actor will achieve it; elaborate to tasks)
- delegate it to an existing or new actor
- decompose it into new subgoals

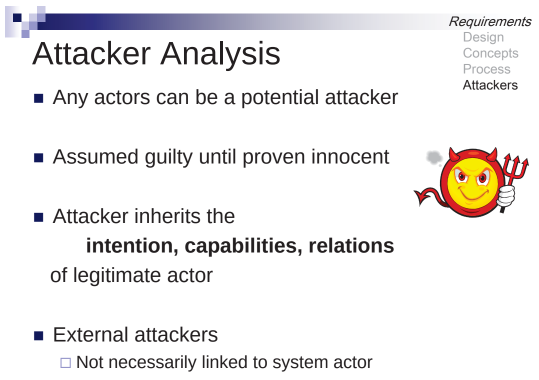
Finish when all goals have been adopted.

Here we shortly cover one example security requirement approach (See also e.g. [4]). Other approaches may work just as well, what is important is that the security requirements are considered throughout in a structured and consistent way.

Identify actors and goals. The first step in gathering the requirements is determining the stakeholders and their interests. The stakeholders are those parties with a legitimate interest in the system that we are designing. Their interest and goals thus have to be considered (though not necessarily completely reached - we may need to make trade-offs between the different goals of the participants).

The stakeholders and their interests become the initial actors and goals in the requirements gathering process. If an agent has the right capabilities, it may *adopt* a goal, i.e. take responsibility to achieve it. If an agent does not adopt the goal it may be delegated to other agents (either existing or new) or be split into new sub goals. Agents do not work in isolation; agents and their goals may depend on/interact with each other. These dependencies should be identified and could lead to new goals and/or agents. They also lead to potential vulnerabilities, e.g. when agents' goals conflict.

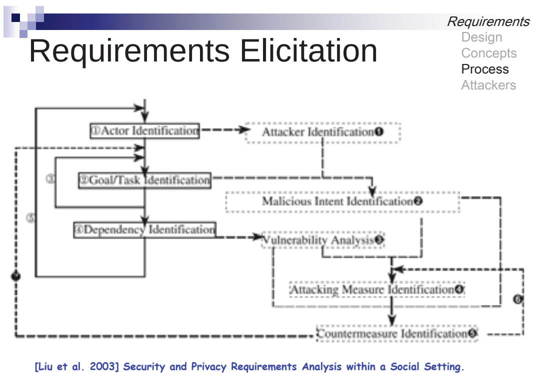
So far the process matches a typical functional requirement engineering approach. To also deal with security requirements we also need to consider attackers and possible attacks on the system.



Requirements
Design
Concepts
Process
Attackers

Attacker Analysis

- Any actors can be a potential attacker
- Assumed guilty until proven innocent
- Attacker inherits the
  intention, capabilities, relations
  of legitimate actor
- External attackers
  - Not necessarily linked to system actor



Requirements
Design
Concepts
Process
Attackers

Requirements Elicitation

①Actor Identification
Attacker Identification❶
②Goal/Task Identification
③
Malicious Intent Identification❷
⑤
④Dependency Identification
Vulnerability Analysis❸
Attacking Measure Identification❹
❻
Countermeasure Identification❺

[Liu et al. 2003] Security and Privacy Requirements Analysis within a Social Setting.

Identify attackers, vulnerabilities and attacks Outsiders may try to attack our system and they need to be considered along with their goals, however, also the risk of attacks by insiders need to be accounted for. Each agent in the system could potentially become an attacker, using its capabilities and place in the system to reach their goals at the expense of the goals of other agents. Both type of attackers are modeled as agents in the system but with malicious intent as their goal.

Based on vulnerabilities and the malicious intent of attacker agents we identify potential attacks and assign countermeasures to protect against such attacks. The countermeasures themselves may lead to new actors/goals and/or open the possibility for new attacks which need to be considered. Refinement of the system continues until all goals have been assigned, dependencies taking into account, and vulnerabilities addressed.

1.3 Conclusions and where to go from here

The goal after this lecture/chapter is that you will never look at the word 'secure' in the same way again: Whenever you encounter 'secure' always think - what set of security requirements (which security attributes for which resources) are really meant by 'secure' (what are the security policy and model) and what type of attacker is considered (what is the attacker model).

This chapter provides a general introduction in the topic computer security and security engineering. The notions introduced here will return in more detail in the chapters that follow. For more general information on security see the Security Engineering [2] book by Andersson which provides a nice overview of different topics in a very accessible way.

1.3.1 Literature

Suggested reading (check the course page [1] for the most up to date list of suggested reading materials):

- Security Engineering Introduction [2, Ch 1].
- A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs [4].
- CCWASPSS white paper [3].
- A metric framework to drive application security improvement [6].

1.4 Exercises

1. Find a security related news article and analyze it (collect some background information when needed).
   - What is the security issue in this article ? Is it related to availability, confidentiality, integrity, etc.
   - For a security incident; what was the failure, why did it occur, how could it have been prevented, how should it be solved
   - For a solution/technology; what problem is solved, how can it be used, will it work
   - For an opinion/analysis/...; do you agree, what are possible other/counter arguments
   - For a more general article; what is the issue you have identified, did the article address this issue? How well was the issue described, does the article get the key points correct, Did it miss any issues.
   - Is the article biased/one-sided? Does it consider the issue from the perspective of different stakeholders?
   - Do you agree with the conclusion of the article?
2. An online music store allows its members to listen to music with embedded ads for free and to download music without ads for a fee. Members can also recommend songs to other members and get a free ringtone if at least five people listen to a song based on this recommendation. Do the first steps in a basic 'security requirements engineering' for this scenario: identify actors, their interests and interdependencies. Also find attackers and their goals. (As we discuss security tools in later lectures the design can be extended with potential countermeasures.)