Systemsicherheit

1. **Risks in Electronic Payment**

   From your personal experience: Which risks are involved in electronic payment systems? Start with thinking about the vulnerabilities of today's methods, procedures, and mechanisms you are familiar with. Here are two possible scenarios:

   (a) Paying with a debit card (e.g. EC Maestro): starting with its use in a shop and ending with the money withdrawal from your bank account.

   (b) Home banking using a static (e.g. snail-mailed) or dynamically generated transaction authentication number (TAN), e.g. sent from your bank via SMS (mTAN) or using a smartphone app (pushTAN).

   What are the advantages of smart cards (such as your thoska), carrying a microprocessor for cryptographic computations?

   Solution: Electronic payment involves theft of data or money. Hackers may get access to bank accounts and use it the same way as the normal user but with different intentions (get rich). To prevent hacking of accounts, banks use different ways of defense.

   While paying with a debit card, the user must provide the card (physical item) and the pin code (knowledge). To prevent bruteforce attacks, a bank account is locked after a short number of invalid pin codes.

   Home banking uses a password (knowledge) and a TAN via Mail/Phone to have the possibility of hackers minimized.

2. **Buffer Overflow Attacks**

   Which vulnerabilities are exploited by a buffer overflow attack? How can you counter buffer overflow attacks? How could you at least mitigate the effects of successful buffer overflow attacks?

   Solution: Buffer overflow attacks aim to trick the softwar to execute futher attack code and exploit whatever the hacker needs. To prevent buffer overflow, one must check the maximum possible length of the input versus the users input. To mitigate any successfull attack, a programm should be contained and not have access to further information or programms but the necessary.

3. **Data vs. Information**

   (a) What is the difference between data and information?

   (b) What are the consequences for systems security?

   Solution: Data is a collection of values like characters, numbers or other data types. Unprocessed data have little to no meaning to a human. Information is processed data so a human can read, understand and use it.

4. **Root Kits**

   Which special properties of root kits make them so extremely dangerous?

   Solution: Invisible, total sustainable takeover of a complete IT system. Root Kits are a comprehensive tool kit for fully automated attacks on all levels of the software stack.

5. **NI: Dynamic Properties**

   Similar to HRU, an NI model is basically formalized through a deterministic automaton. Can we also use it to analyze HRU Safety (no matter if by proof or by simulation)?

   If yes: How would HRU Safety for NI be defined (in prose)? If no: What extension of the NI model in the lecture would be required to enable Safety analysis?

   Solution: With an NI model you can proof if it is NI-secure but not HRU safety as the NI Model holds no subjects or objects but domains and states. It must be extended with sets of subjects and objects to do so.

6. **NI: Motivation**

   Which security problem do NI models address? Name two modern application scenarios where this problem is highly relevant!

   Solution: The two problems NI models address are Covert Channels (Channels not intended for information transfer) and Damage Range.

   E.g. Multi-application Smart Cards and Server systems

7. **BLP and Biba**

   What's the difference in terms of goals and formalism between the BLP and the Biba model?

   Solution: BLP is the upside down Biba model in view of confidential information flow from high to low

   - BLP preserves confidentiality
   - Biba preserves integrity

   Biba classifies with a hierarchy and chooses the lowest possible for each object while BLP uses the highest possible

8. **BLP: Lattice vs. ACM**

   (a) Why does the BLP model contain both: (1) a lattice, which is mapped to subjects and objects via cl, and (2) an ACM?

   (b) What problem might occur from using both (1) and (2)?

   Solution: The Bell-LaPadula Model is a model engineered from know abstractions. The lattice that maps subjects and objects is needed to model confidentiality hierarchy (public or confidential subject and objects). The ACM next to that stores the read and write access of subjects and objects. Both can be used alone to check confidentiality and the ACM can be used to analyze the ACF of the model.

9. **DAC vs. MAC**

   (a) What is the difference between discretionary (DAC) and mandatory access control (MAC)?

(b) What are the weaknesses of discretionary access control systems?

Solution: DAC: Individual users specify access rules to objects within their area of responsibility (ät their discretion"). E.g. locally within a project, team members individually define permissions w. r. t. documents (implemented in project management software and workstation OSs) inside this closed scope. With DAC it can happen to loose confidentiality and it's weaker to social engineering attacks.

MAC: System designers and administrators specify system-wide rules, that apply for all users and cannot be sidestepped. E.g. globally for the organization, such that e. g. only documents approved for release by organizational policy rules (implemented in servers and their communication middleware) may be accessed from outside a project's scope.

10. **TAM: Type System**

How is the type system in TAM formally represented within the HRU-based automaton? Which parts of the type system may change during runtime?

Solution: All Subjects can also act as objects and can be used to model delegation of rights. Each objects has a type from a type set trough a mapping style. An HRU model is a special case of a TAM model $T = \{tSubject, tObject\}$ with $\forall s \in S : type(s) = tSubject; \forall o \in O \backslash S : type(o) = tObject$

11. **TAM: Motivation**

What is the goal of the TAM security model?

Solution: AC model, similar expressiveness to HRU that can be directly mapped to implementations of an ACM: OS ACLs, DB permission assignment tables. Better suited for safety analyses: precisely statemodel properties for decidable safety.

12. **RBAC ACF**

As with any AC model, the formal components of RBAC are designed to enable access control decisions. However, in the ACF definition of RBAC0 (which is the basis for ACFs of the other RBAC96 models), the component UA is not included. Why?

Solution:

13. **RBAC Safety**

How can we analyze RBAC safety? Which information is needed for this, and is it provided by RBAC96 models?

Solution:

14. **IBAC vs. RBAC vs. ABAC**

What is the key difference between IBAC and RBAC models? How about ABAC models then?

Solution: IBAC is the fundamental model and provides basic expressiveness. The abstraction is very low and not application oriented. No indirection between subjects and objects, Subjects and Objects have direct rights assigned to each other.

RBAC: indirection via roles assigned to subjects

ABAC: indirection via arbitrary attributes assigned to subjects or objects. More scalable and manageable.

15. **HRU Safety Undecidability**

    Given HRU Safety is undecidable, what is the actual merit of this model? What can we do to handle the undecidability problem in practice?

    Solution: We can narrow down the complexity of an HRU model to make it decidable and check, if the model instantiation itself is safe. E.g. a mono-operational model is not useful but expressive and is efficient to analyse and design.

    Another way may be heuristic analysis to have heuristics guided safety decisions.

16. **HRU: Unix Read**

    How do we model a read operation, such as for a Unix-OS file system, in HRU? Remember that this operation neither modifies the subject set, nor the object set, nor the ACM.

    Solution:

17. **ACM vs. HRU**

    What is the idea that distinguishes an ACM from an HRU model? How is it formally represented?

    Solution: An Access Control Matrix is a well-structured store to efficiently evaluate and completly analyze an Access Control Function. It only models a single state and cannot predict future proliferation of rights.

    The Harrison-Ruzzo-Ulman Model uses the single protection state of an Access Control System and deterministic automata to model runtime changes of these protection states. Formally it uses a State Transition Scheme with given primitives.

18. **HRU: Output Function**

    Why does an HRU automaton not have an output function?

    Solution: The result of the automaton (if its safe or not) is not resulting in an output function but the state space q' of Q in comparison to q (inital state; $safe(q_0, r)$)

19. **Core-based Model Engineering**

    Assume you have to design the security policy for a very simple hospital information system, including

    - users in roles such as physician, nurse, etc.

- legal information flows between these roles
- one operation to change a user's roles.

Re-use the model abstractions you know from chapter 3 to express this policy as a core-based model by answering the following questions:

(a) Which formal components do you need beyond the actual model core? (2 sets and 2 relations should suffice!)

(b) What is the core specialization?

(c) What is the core extension?

(d) What are possible pre- and post-conditions of the only operation?

Solution:

20. **Common Model Core**

What is the common model core shared by models such as HRU, DRBAC, BLP, Brewer-Nash, and NI?

Solution: The common model core is $\langle Q, \sum, \delta, q_0 \rangle$

21. **Hybrid Models**

Name the semantical concepts from AC, IF and/or NI models that can be found in

- the Brewer-Nash model
- the LR-CW model
- the the MLS-CW model.

Compare how these three models express allowed information flows according to the CW policy.

Solution: Brewer-Nash: no information flow between competing objects. Competing objects belong to the same class. With the history relation, subjects can access new objects only if they didn't access competing objects before.

Least-Restrictive: includes time as a model abstraction. no conflicting information is accumulated in the subject/object potentially increases the amount of information in the subject/object

MLS: labels entities. Class of an entity (subject or object) reflects information it carries and is reclassified whenever an data object is read.