Secure Programming Techniques

Margus Freudenthal, Mart Oruaas

Lecture 2, 2022-02-18
Security Models
A Linguistic Note

- Merriam-Webster dictionary gives fourteen different definitions for the noun “model”. Some of the usages of the word in information technology trace back to different definitions than others.
- This is a norm which one must get used to in professional life.
What are Security Models? 1/2

- There is no such thing as just “security”, security is always driven by some need and bound by some context.
- Security models capture those needs and boundaries.
- Security models are domain specific: physical security vs IT security vs cryptographic security models are very different.
What are Security Models? 2/2

- Generally, a security model is a set of properties of a system and its surroundings and it defines what it means for the system to be “secure” given those properties.
- There are also models for the process of securing the system (Common Criteria, ISO 27000 framework).
There have been attempts to model the whole human sense of security. Drawing from [Mäkinen, 2005].
Kinds of Security Models in Computing

- Data security models: Confidentiality-Integrity-Availability (CIA)
- Threat models: STRIDE
- Cryptographic models: random oracle model, Dolev-Yao
- Access control models: RBAC, ABAC, MAC, etc
CIA Triad

- **Confidentiality** – a product or system ensures that data are accessible only to those authorized to have access.
- **Integrity** – a system, product or component prevents unauthorized access to, or modification of, computer programs or data. undetected manner
- **Availability** – data has attributes that enable it to be retrieved by authorized users and/or applications.

Useful framework to think about security properties, but too simplistic for discussing more complicated security properties.
Parkerian Hexad

Donn B. Parker added additional attributes to the CIA triad in 1998 [Pender-Bey, 2019]. It is one of the many CIA model improvements

- **Authenticity** – the identity of a subject or resource can be proved to be the one claimed
  - Sometimes also called non-repudiation

- **Possession** – control or physical possession of data

- **Utility** – data is in a useful format or state
Security Assurance Models
Common Criteria Model

- Common Criteria [CC, 2017] is a framework for defining the composition and security properties of IT products.
- It has two main components: ways of defining properties and describing how stringently those properties are verified (formalized by Evaluation Assurance Level, EAL).
- It may not be suitable for most of the software products, but it contains some very good ideas which are worth exploring.
- Most notably: how to talk about an IT product’s structure very clearly.
Topics Covered in the Common Criteria

- Composition, interfaces, boundaries - all parts of any complete architectural description.
- Security objectives - assets, threats, description of the environment.
- Security architecture - enforcements, security domains, initialization processes.
- Life-cycle support - development process, vulnerability tracking.
- Testing methodology, coverage.
NIST Cybersecurity Framework 1/2

Five functions that form a successful cybersecurity program [NIST, 2018]

- **Identify** – understand assets, risks, context, strategy, etc.
- **Protect** – take measures to protect critical services
- **Detect** – identify occurrence of a cybersecurity event
- **Respond** – take action regarding the detected event
- **Recover** – restore capabilities and services
Five functions work in combination, supporting each other

When speaking of security, people typically think about preventing attacks, but total prevention is usually not possible. We can only talk about deterrence – making things more difficult for the attacker

Example:

- Fences and locks prevent (actually inconvenience, deter) theft
- Alarms and watchful neighbors detect incidents
- Police or private security responds to incidents
- Insurance helps to recover from losses
Implementation Models
Mandatory and Discretionary Access Control 1/2

- **Discretionary access control** – subjects (typically owners of resources) can define access control rules for objects.
- They can also set "insecure" access rights, leading to violation of security policy.
- This is the most common model, implemented nearly everywhere.
Mandatory and Discretionary Access Control 2/2

- **Mandatory access control** – the system enforces the security policy.
- The policy is centrally controlled by a security policy administrator.
- In pure system, the policy defines all the actions a subject can take. In a hybrid system, the system policy limits the access control decisions of the subjects.
- In Linux, SELinux and AppArmor aim to achieve similar results.
Cryptography: Dolev-Yao model

- The adversary in this model can overhear, intercept, and synthesize any message and is only limited by the constraints of the cryptographic methods used. In other words: “the attacker carries the message.” [Dolev and Yao, 1983]

- The model may come from cryptography, but is very useful mental framework for anyone working on distributed systems and their quality.

- The main value of this model for any practitioner is the lack of any (wrong) assumptions about the capabilities of the adversary.
Access Control: XACML 1/3

- XACML: eXtensible Access Control Markup Language [OASIS, 2005]
- A language for writing access control rules and a protocol for requesting access control decisions. Provides a very clear model for thinking about access control

Example rules:

- A person may read any record for which he or she is the designated parent or guardian, and for which the patient is under 16 years of age
- A physician may write to any medical element for which he or she is the designated primary care physician, provided an email is sent to the patient
Access Control: XACML 2/3

1. View record #123
2. Can Alice view record #123?
3. Evaluate policies
4. Retrieve additional attributes
5. Permit, Alice can view record #123
6. View record #123

PEP

PDP

PAP

PIP
Access Control: XACML 3/3

- Policy Decision Point – makes the decision whether some access request is satisfied or not
- Policy Enforcement Point – processes the user’s request, asks the PDP for decision and acts on the decision
  - Also responsible for executing obligations (e.g., create audit log record)
- Policy Information Point – reads attribute values (for resources, subjects, environment)
- Policy Administration Point – manages access authorization policies
Execution Models
Program Execution

- “The meaning of term \( t \) can be taken to be the final state that the machine reaches when started with \( t \) as its initial state.” [Pierce, 2002]
- In layman’s terms: the meaning of a computer program is what it does, in what state does it end up in
- State transitions of a program depend on the execution model of the programming language
- Program’s security and safety depend on its state transitions, in many ways. It comes down to understandability, flexibility, implementation quality, etc of the execution model
Program Execution Models

- Program execution models are defined by the language, usually together with the execution environment (OS syscalls, VM specification etc).
- Poorer execution models tend to cause large amounts of problems.
- “How ISO C became unusable for operating systems development” [Yodaiken, 2022]
- Modern languages, starting somewhat with Java and ending with Go and Rust try to define the execution models in such a way that programmers have minimal number of ways to make mistakes.
Program Execution Environments

- CPU-s and GPU-s, including modern hyperoptimized silicons with their leaky behaviour
- Native code in classic “fat” operating systems: Windows, Linux, UNIX
- Mobile operating systems, based on some fat platform with a specialized layer on top of it
- Interpreted languages: PHP, Python, Bash
- Virtual machines: JVM, WebAssembly, cryptocurrency VMs
- Web browser
- Cloud environments
Security Issues with Execution Environments

- All EEs come with their own peculiarities which a developer planning to write secure code must understand.
- C’s NUL-terminated strings are called “a billion dollar mistake.”
- Interpreted languages (esp. PHP and Bash) do everything to execute every random input they get.
- Ethereum’s poorly designed Solidity language and VM have been costing millions every day for the last quite many months.
Security Models: Conclusions

- As can be seen, security model(s) come in very different shapes and sizes.
- As any IT system is multifaceted, every single security model can cover only some of those facets.
- This is why we need multiple kinds of models.
References I


On the security of public key protocols.
References II

Strategic Security.

Cybersecurity framework version 1.1.

eXtensible Access Control Markup Language (XACML) Version 2.0.
http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
References III

The Parkerian hexad.
Master’s thesis, Lewis University.

Types and programming languages.
MIT Press.

How ISO C became unusable for operating systems development.
CoRR, abs/2201.07845.