DistriNet Research Group

Research Directions in Secure Software Engineering

Riccardo Scandariato
Katholieke Universiteit Leuven
Belgium

DistriNet Research Group

Acknowledgments

- This work is part of the SoBeNeT project, an IWT project funded by the Flemish government
- Visit http://sobenet.cs.kuleuven.be

DistriNet Research Group

DistriNet: The research group

- Distributed Systems and Computer Networks (DistriNet)
  - Founded in 1984
  - Department of Computer Science at the K.U.Leuven university
- 6 academic staff members, 8 post docs, 49 researchers and PhD students, and 5 part-time members (*68 persons*)
- DistriNet works on a wide range of problems

DistriNet Research Group

Fask forces

- Agents
- Embedded systems
- Language technology and middleware
- Networking
- SecAnonym
  - Privacy and anonymity
- SecDam
  - Distibuted applications and middleware
- SecLan
  - Language technology and formal methods

2 professors
2 postdocs
9 researchers/PhDs
1 part-time

DistriNet Research Group

Secure Software Engineering?

- MITRE has been tracking vulnerabilities
  - Common Vulnerabilities and Exposures (CVE)
- 4500 vulnerabilities were tracked in 2005
  - 55% increase
  - 75% due to faulty application software
- Good quality building blocks are available
- Vulnerabilities as result of immature software engineering

DistriNet Research Group

Secure Soft Eng – At a glace

- Security requirements
  - Evaluation study (and extensions) of Tropos, KAOS, MUC, Problem Frames
  - With Bart De Win (and master student)
- Security-aware process
  - Activities, profiles, RUP, XP
  - With Bart De Win (and 2 juniors)
- Security patterns
  - Pattern-based construction of security-aware architectures
  - With 2 PhDs
- Security metrics
  - Measuring security properties

DistriNet Research Group

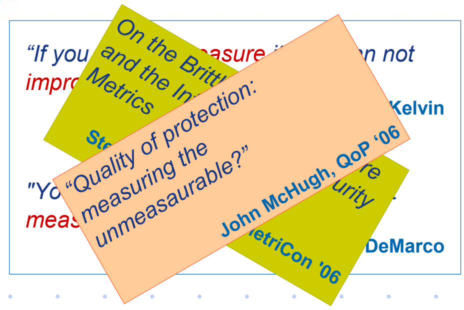
Towards a Measuring Framework for Security Properties of Software

Metrics

Presented at CCS - QoP '06
Riccardo **Scandariato**, Bart **De Win**, Wouter **Joosen**

DistriNet Research Group

Motivation



DistriNet Research Group

Motivation

"A major difference between a well-developed science such as physics and some of the less well-developed sciences (security?) is the degree to which things are measured"

Fred Roberts, Measurement Theory with Applications to Decision-making, 1979

DistriNet Research Group

Quality attributes

- Internal attributes of a product can be measured in terms of the product itself
  - During creation of the product
  - E.g., structural properties such as coupling
- No externally visible quality of a product
  - No meaning in themselves

DistriNet Research Group

Quality attributes

- External attributes are features of the product that are externally visible (w.r.t. environment)
  - Measurable directly only after creation
  - E.g., reliability, maintainability, security
- See ISO 9126 – Quality model
- Functionality
  - Suitability
  - Accuracy
  - Interoperability
  - Compliance
  - Security

DistriNet Research Group

Prediction models

- Estimate the future external attributes of a system from present internal attributes
  - "Quality from properties"
- Empirical exploration of internal/external relationships (past projects)

Problem statement

- Lack of quantitative methodologies to assess security
- Larger gap for key software development phases (e.g., design)

- Which properties must be considered?
- How to measure them?
- What impact do they have?

The SE lesson (maintainability)

- *External attribute* (quality) quantitatively estimated by measuring *internal attributes* (properties) such as size and complexity
- Those properties can be seized at different levels of abstraction
- Empirical exploration of internal/external relationships[1]

[1] **Li and Henry, *Object-Oriented Metrics that Predict Maintainability*, 1993**

Approach

- Properties from security principles and best practices
  - Saltzer-Schroeder
  - *OWASP*
  - McGraw, *Building Secure Software*
  - NIST Special Publication 800-27, *Engineering principles for IT security*

Security principles (e.g., OWASP)

- Apply defense in depth (complete mediation)
- Use a positive security model (fail safe defaults)
- Fail safely
- Run with least privilege
- Avoid security by obscurity (open design)
- Keep security simple (economy of mechanism)
- Detect intrusions (compromise recording)
- ...

Properties & metrics (1/3)

- Principle: Keep It Small and Simple
  - Property: Size
    - Metrics: SE metrics (e.g., Chidamber & Kemerer)
  - Property: Complexity
    - Metrics: Software engineering metrics
  - Property: Size of attack surface
    - Metric: # points of access (archi)
    - Metric: # classes processing user input (design)
    - Metric: Coverage of validation routines (code)

Properties & metrics (2/3)

- Principle: Implement Layered Security
  - Property: Lines of defense
    - Metric: # data validations per information flow
    - Metric: # authentication/authorization checks per usage scenario
- Principle: Someone Must Be Accountable
  - Property: Degree of accountability
    - Metric: # non-audited operations vs. total # ops

Properties & metrics (3/3)

- Principle: Find and Minimize Criticalities
  - Property: Number of critical modules
    - Metric: Instability of components
    - Metric: Number of entities to be trusted
    - Metric: Risk-based priority
- Principle: Separation of Concerns
  - Property: Degree of security SoC
    - Metric: Concern diffusion (modules&operations)

The way ahead (short term)

- More properties must be elicited
  - CC, ISO 17799
- Extend classification framework
  - E.g., security objectives, application vs. environment
- Guidelines to correlate and interpret
  - Empirical studies

The way ahead (longer term)

- Methodology to easily select the right metrics for the job (a là GQM)
  - See work on patterns&metrics[1]
- The cost of measures must be low
  - Automation

[1] **Heyman and Huygens, *Software security patterns and security metrics*, MetriCon 2006**

The way ahead (patterns)

- Attach metrics to security patterns to bring them closer to application semantics
  - Right granularity
- Input guard
  - #guards / #access points (development)
  - #filtered calls / #calls (operational)
- Audit interceptor
  - #invocations / #audit events (operational)

A Systematic Approach to Secure Design with Patterns

Patterns

Submitted to AsiaCCS '07
Koen **Yskout**, Thomas **Heyman**, Riccardo **Scandariato**, Wouter **Joosen**

Motivation

- Existing methods to define security requirements
  - KAOS, Tropos, problem frames, etc.
- Huge gap between requirements and design (w.r.t. security)
  - A lot of expertise needed
- One way to bridge the gap: security patterns
  - Well-known technique to provide domain-independent, time-tested knowledge and expertise
  - Preserve this knowledge in a reusable format

State-of-the-art

- List of sources were analyzed
- Christopher Steel, et al, Applied J2EE Security Patterns: Architectural Patterns and Best Practices
- Markus Schumacher, et al, Security Patterns: Integrating Security and Systems Engineering
- Open Group guide
- Darrel M. Kienzle, Patterns repository
- Ronald Wassermann, Betty H.C. Cheng, Security Patterns, Michigan State University
- Yoder and Barcalow, Architectural Patterns for Enabling Application Security
- …

SOTA – Critique

- Overlap
- Hard to use (many levels of abstraction)
- Unstructured (no system of patterns)
- Bad patterns
  - Concepts
  - Principles and practices

What makes a good (archi) pattern

- A set of element types (e.g., data repository)
- A topological layout of the elements indicating their inter-relationships
- A set of semantic constraints
- A set of interaction mechanisms that determine how the elements coordinate through the allowed topology

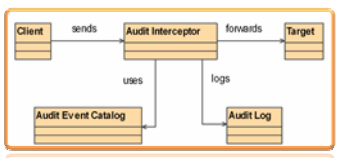
From: Software Architecture in Practice, L. Bass, P. Clements, R. Kazman

Good pattern: Audit Interceptor

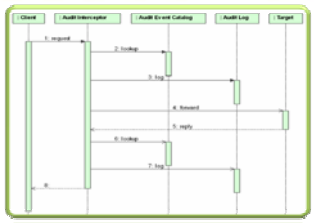
- Problem: You want to audit requests and responses to and from the business tier
  - centralized and declarative
  - auditing decoupled from the applications
- Solution
  - Audit Interceptor intercepts requests and responses.
  - It creates audit events based on the information in a request and response
  - It uses declarative mechanisms defined externally to the application
- Pros
  - Better separation of concerns
    - Burden removed from business component developers
    - Centralization reduces code replication
  - Declarative approach supports evolution

(J2EE Core Security Patterns)

Good pattern: Audit Interceptor



Good pattern: Audit Interceptor



DistriNet Research Group

Not so good pattern: Role Right Def

- Problem: How can we assign rights to the roles when we want to implement a least privilege policy?
  - Solution: Define use cases and interpret actors as roles
- Is this a pattern at all?

(M. Schumacher)

DistriNet Research Group

Contribution

- Inventory created by reduction (80 –> 35)
  - Complexity-wise, removed the overlaps
  - Quality-wise, removed candidates of a less appropriate level of detail
  - Taxonomy-wise, removed patterns at the wrong level of abstraction
- Uniform description according to a template
- System of patterns: meta-information (search & selection)
  - Role in the process
  - Intent in fulfilling a security objective
  - Interrelationships
  - Labels
- Methodology to create secure design with patterns
  - Process to guide pattern selection

DistriNet Research Group

Switch to next slide-set

- A pattern-based approach to build security-aware architectures

DistriNet Research Group

Conclusions

- A lot of security patterns out there…
  … but unstructured, overlapping and hard to use
- Provided instruments for organizing and structuring
- Instruments also help to measure the value of a pattern
- Process/guidance on using the patterns
- Road test of methodology => (almost[1]) works
- Identification of missing patterns

[1] Availability doesn't seem to fit quite well…

DistriNet Research Group

Current status

- Quality review
  - Pattern inventory
  - Documentation of inventory, instruments and system
    - Submitted to AsiaCCS '07
    - K.U. Leuven tech report
- Ongoing
  - Documenting design process
    - USENIX – Security Symposium
    - IEEE Software – Special issue on patterns
  - Applying to larger test case (e-health authorization service)

DistriNet Research Group

That's all folks!