Secure Software Development and Beyond

Pascal Junod

Fri Software Days, Fribourg (Switzerland)
February 1st, 2016

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

strong.codes

Fri Software Days 2016

$ whoami

- Prof. @ HES-SO / HEIG-VD
- Co-founder of strong.codes SA
- Research interests:
  - Industrial cryptography
  - Software security, software protection
  - Ethical hacking
- http://crypto.junod.info, @cryptopathe

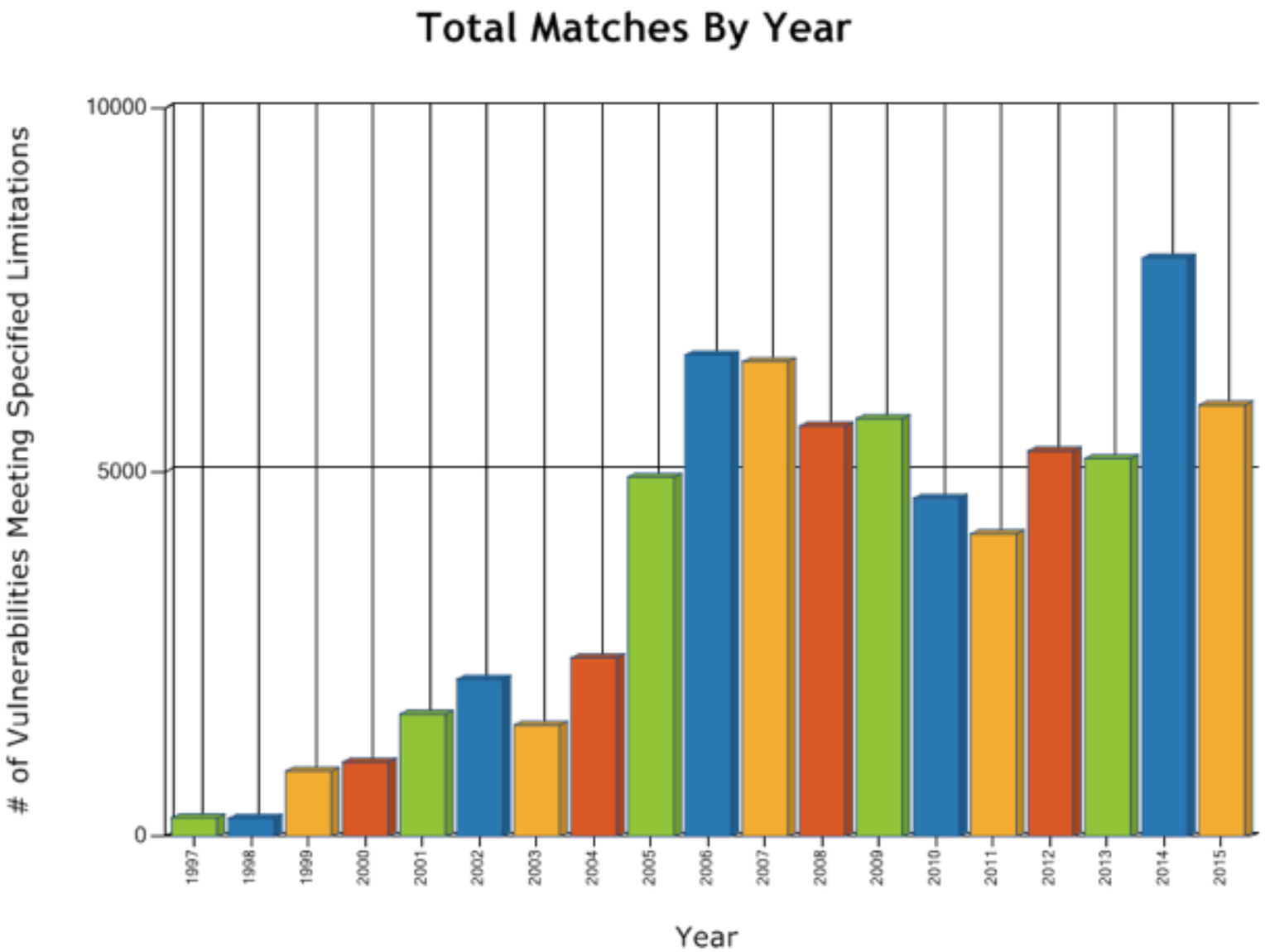
Agenda

Agenda

- Software Security ?
- Secure Software Development
- Man at the End
- Software Protection

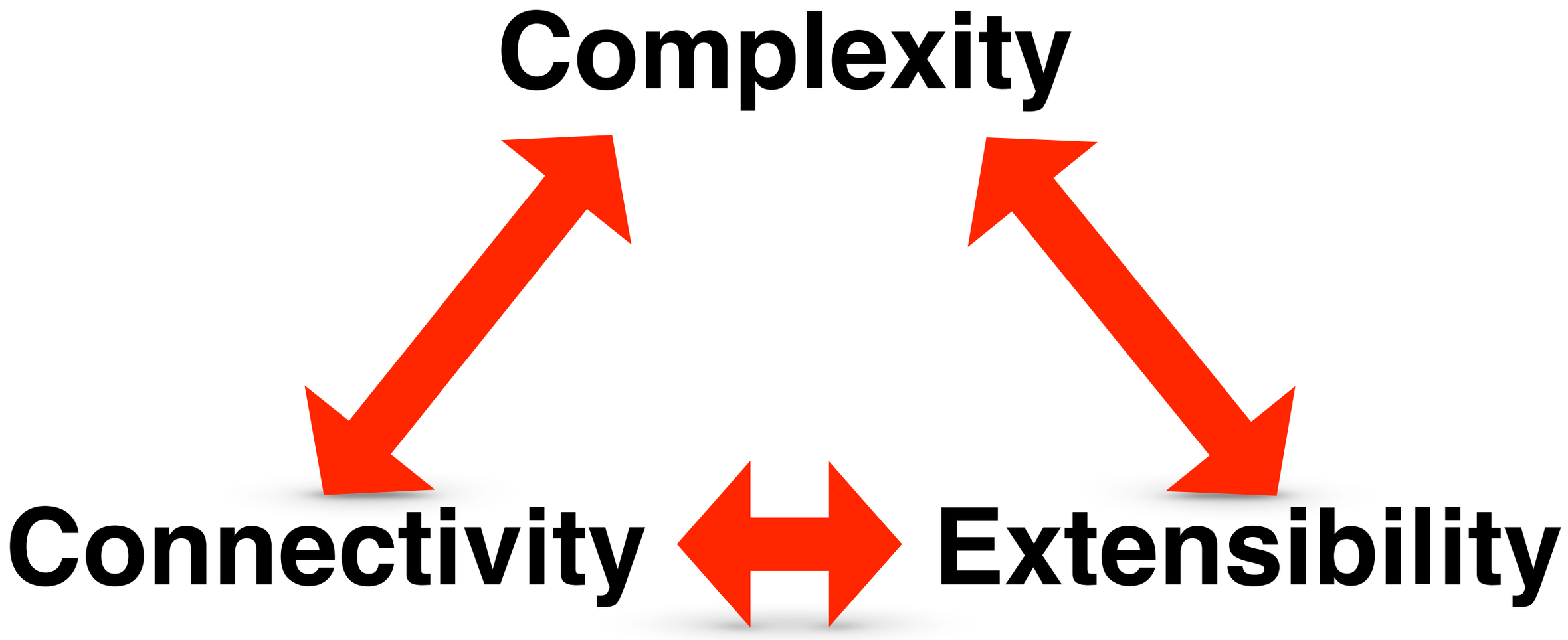
Software Security?

Published Software Vulnerabilities

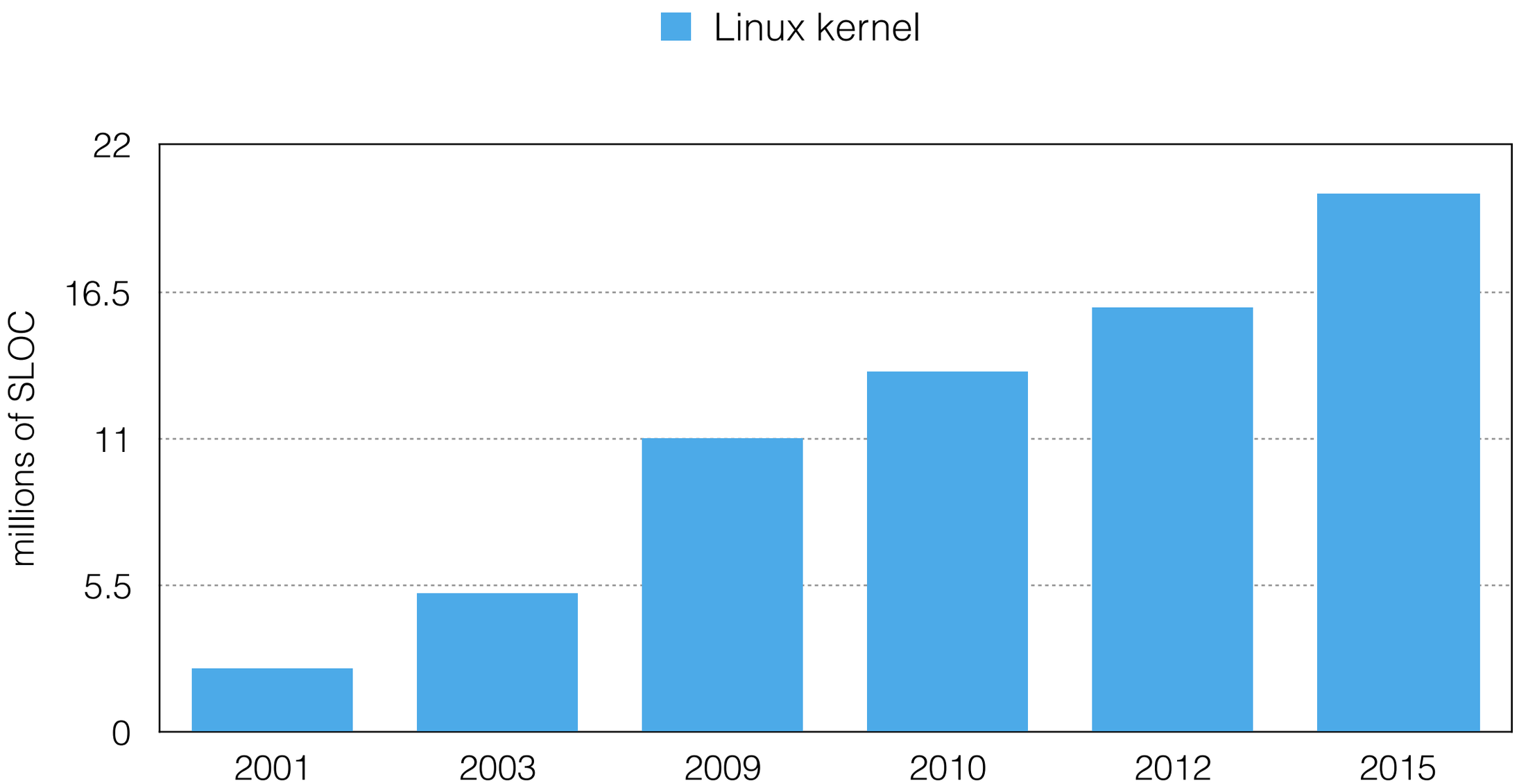


Source: https://web.nvd.nist.gov/view/vuln/statistics

Trinity of Troubles



Complexity


Linux kernel
millions of SLOC
22
16.5
11
5.5
0
2001
2003
2009
2010
2012
2015

Connectivity

- Ubiquitous networking

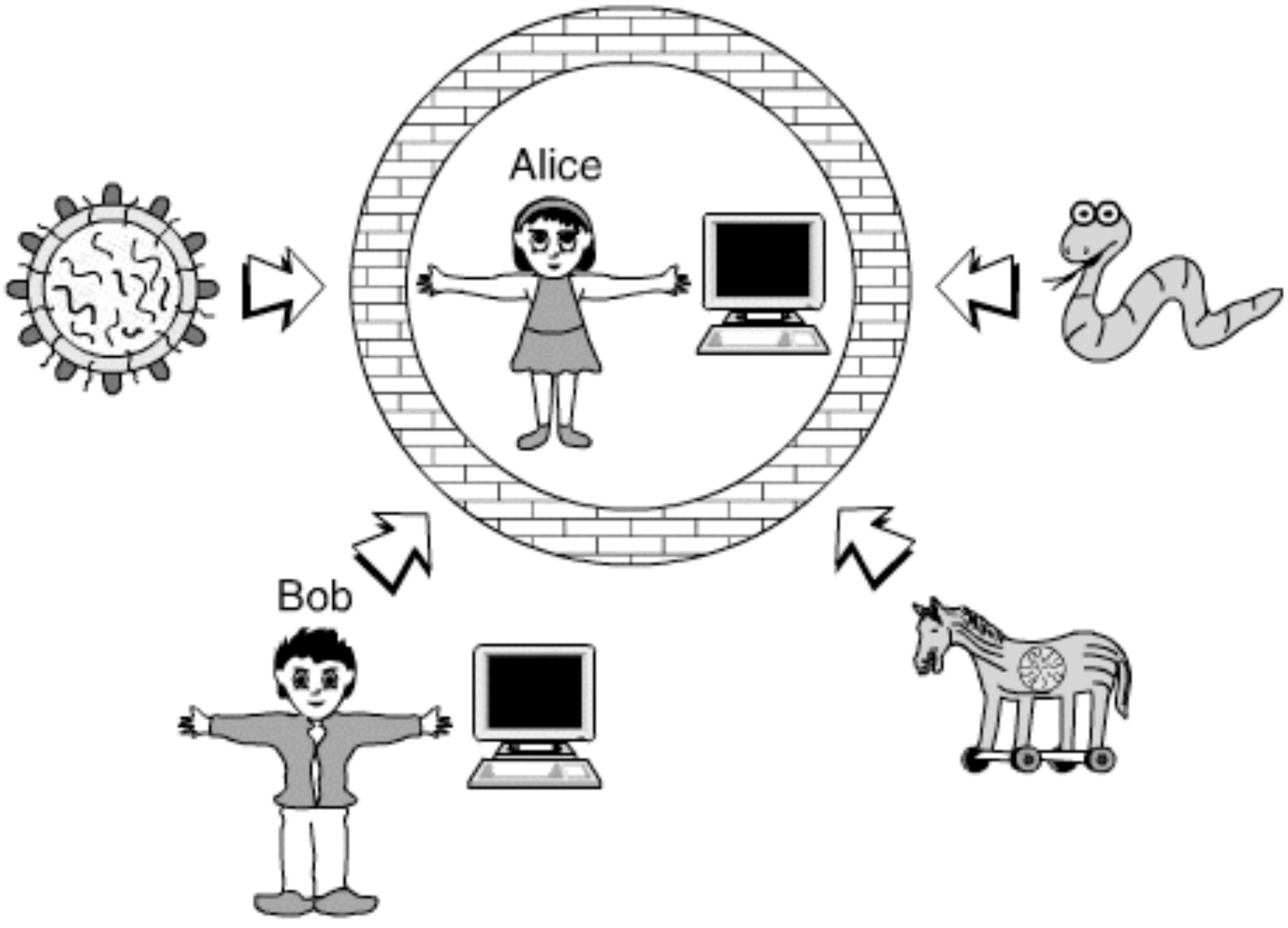
- Service Oriented Architectures, networking of legacy applications

Extensibility

- Plugins, extensions
- Loadable OS drivers
- Scripting, components, applets, controls
- Extensible platforms: J2EE, .NET, etc.

Secure Software Development

Usual Threat Model



Source: Collberg, Nagra, « Surreptitious Software », Addison-Wesley, 2009

Security Development Lifecycle

- Process advocated by Lipner & Howard, « S. Lipner and M. Howard, «*The trustworthy computing security development lifecycle*», Microsoft Corp., 2005.

- Available on http://msdn.microsoft.com/en-us/library/ms995349(classic).aspx

#1 - Security Requirements

- « Security from the grounds up » —> integration of security into the development process
- Nomination of a security advisor
- Identification of key security objectives (risk-driven approach)
- Describe challenges and plans

#2 - Security Design

- Security architecture definition
- Definition of software attack surface
- Threat modeling
  - Assets
  - Risks and their likelihood
- Definition of countermeasures

Threats

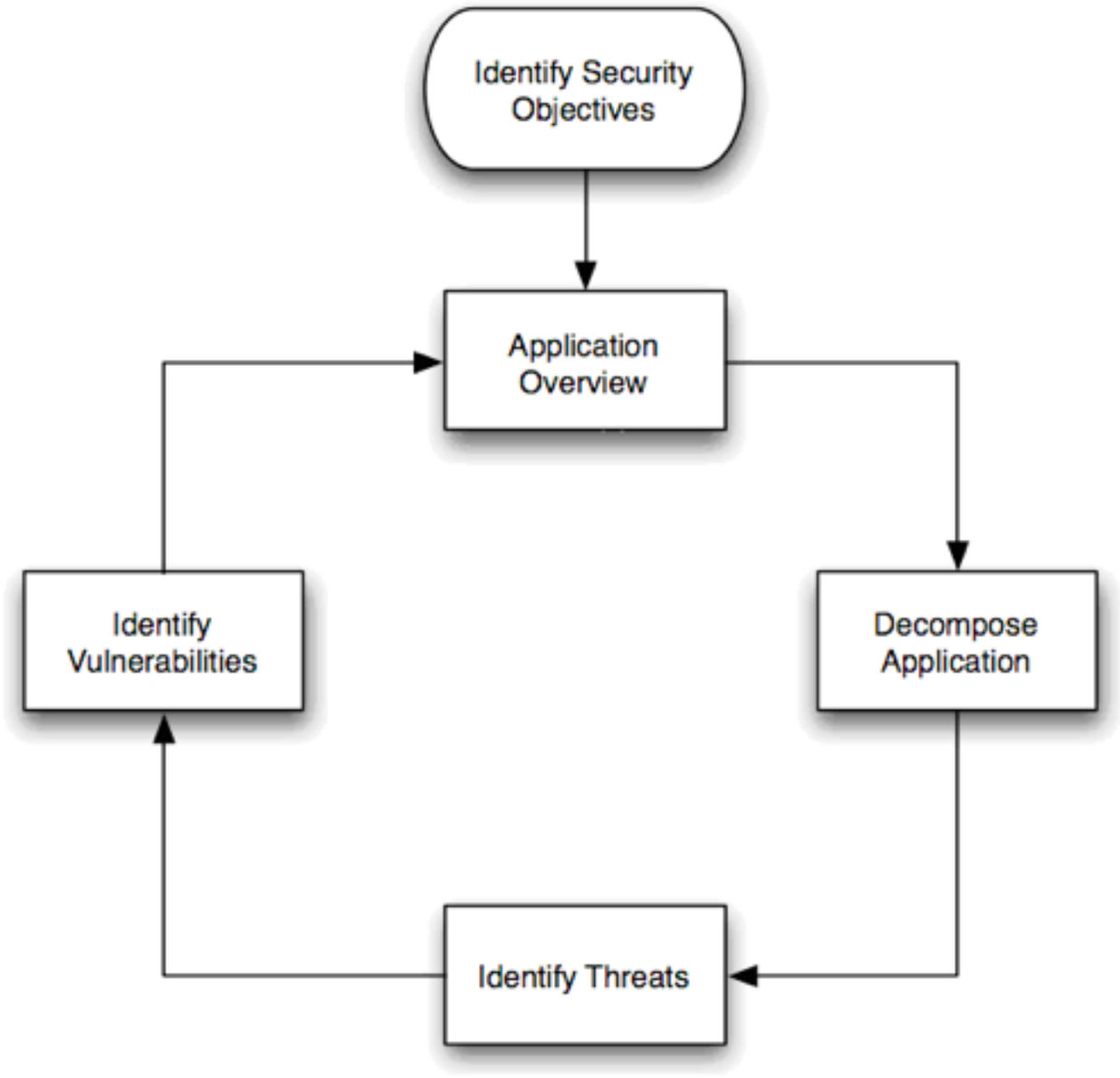
- Non-target specifics: viruses, worms, trojans, etc.
- Employees
- Organized crime, cyber-criminals
- Corporation, nations
- Natural threats

Vulnerabilities

API abuse // Authentication // Authorization // Availability // Code permissions // Code quality // Concurrency // Configuration // Cryptography // Encoding // Environmental

Error handling // General logic error // Input validation // Logging and auditing // Password management // Path // Protocol errors // Range and type errors // Sensitive data protection // Session management // Synchronization and Timing // Unsafe mobile code // Dangerous APIs // …

Threat Modeling



Source: http://www.owasp.org

Threat Modeling Methods

- Various available methods:
  - STRIDE
  - DREAD
  - TRIKE
  - CVSS
- Generic goals: identify components, data flows, trust boundaries

#3 - Implementation of Security

- Application of adapted coding and testing standards
- Specialized libraries
- Use of static analysis code scanning tools in IDE
- Code reviews

#4 - Software Release

- Final security review
- Internal / external security audit (white-/black-box « pen-test »)
- Answer the question « Is this software ready to go to production from a security standpoint? »

#5 - Security Servicing

- Monitoring
- Evaluation of reported vulnerabilities
- Release of security advisories and patches
- Take response actions (review of development process, legal, etc.)

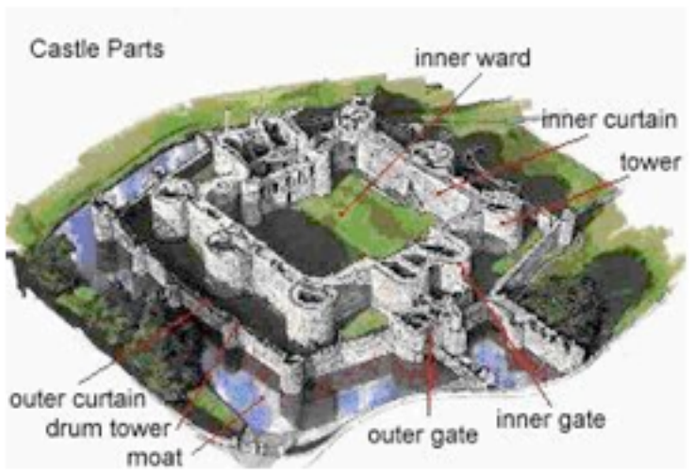
Basic Security Principles

Least Privilege

Give each component only minimal rights necessary to carry out its task.

Defense in Depth

Use several redundant protection mechanisms.



User Participation

A security procedure is efficient only if all its users adhere to its principles.

Choke Point

A security mechanism is effective only if there is absolutely no means to bypass it.

Deny by Default

Forbid everything but what is authorized.

Weakest Link

The adversary will always attack the weakest link of the security chain.

Fail Securely

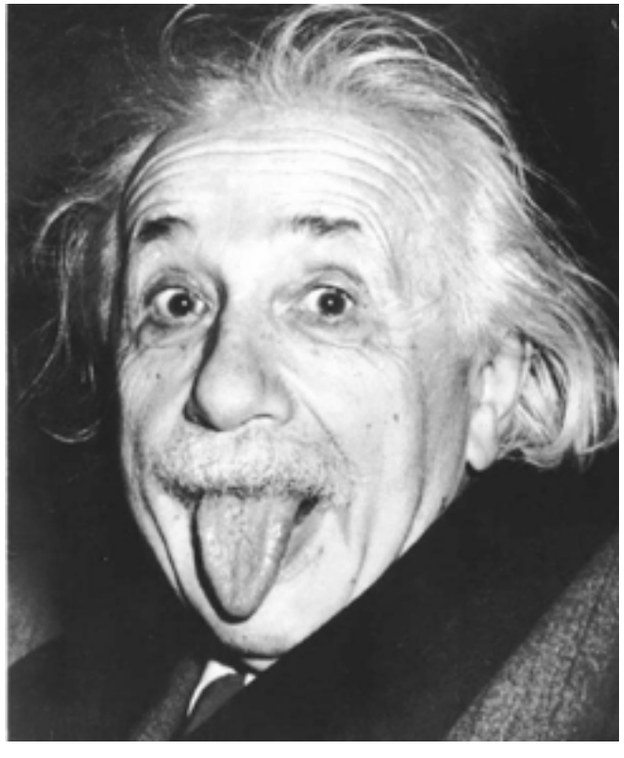
Always ensure that an exception does not open a security hole.

No Security by Obscurity

Do not rely on the secret of the implementation to keep the system secure.

Simplicity

Everything should be made as simple as possible, but not simpler.

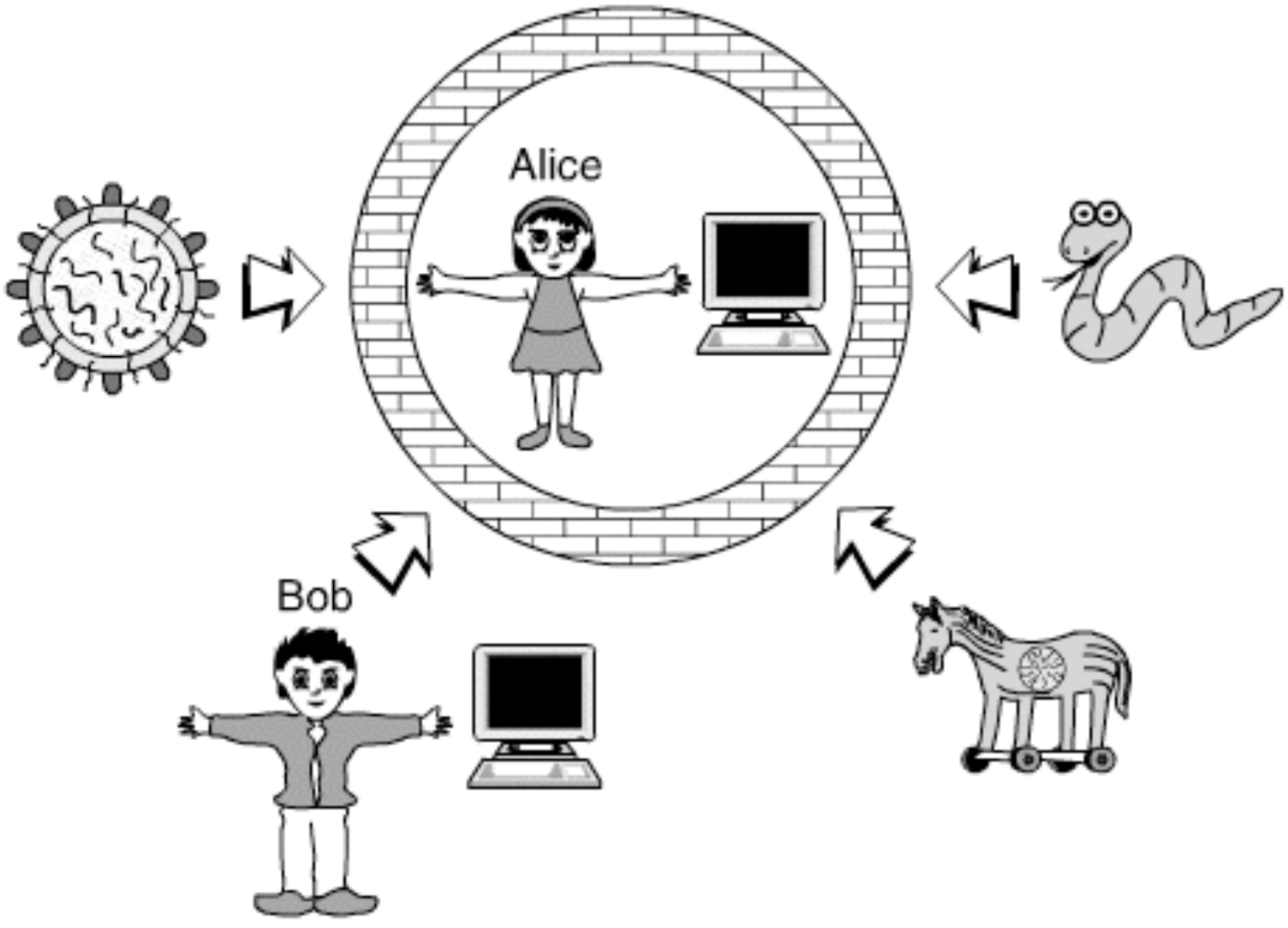


Man at the End

OK, then?

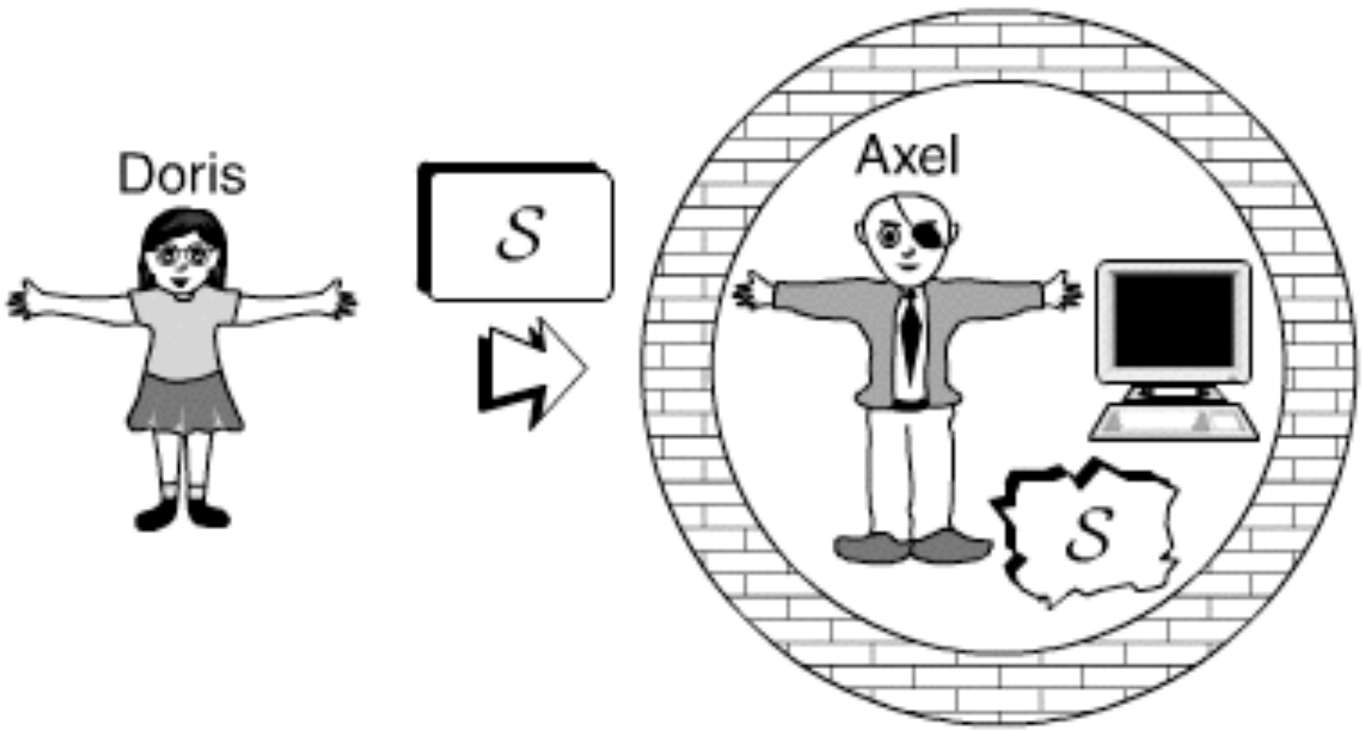
- What happens when you have implemented a perfectly secure software?
- It gets hacked!

Usual Threat Model



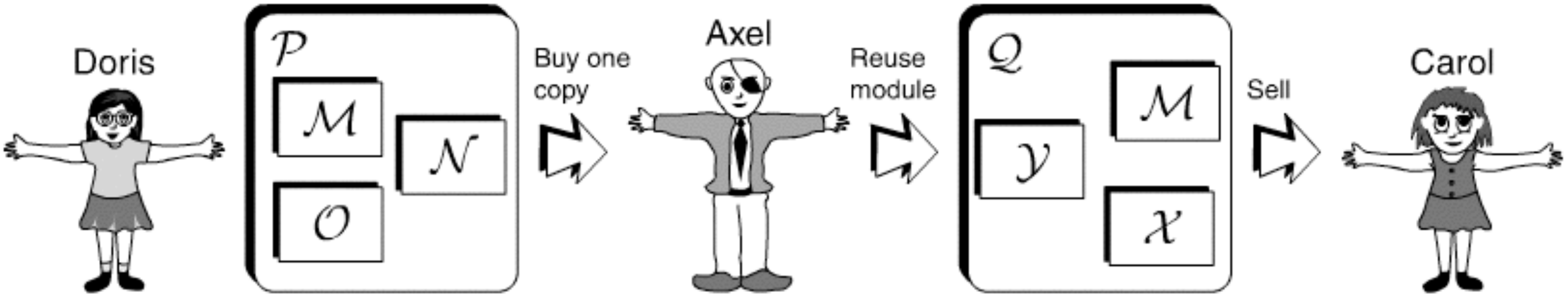
Source: Collberg, Nagra, « Surreptitious Software », Addison-Wesley, 2009

Man at the End



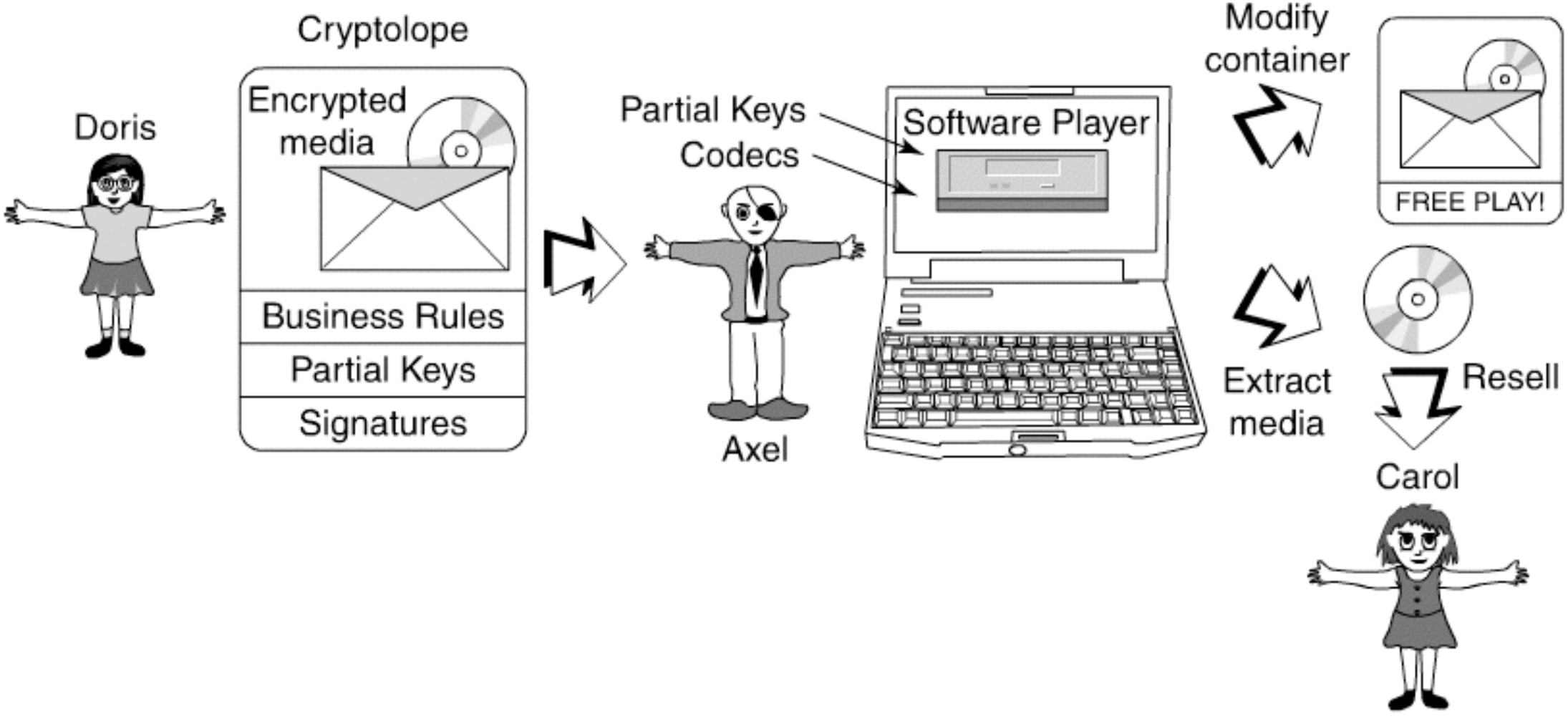
Source: Collberg, Nagra, « Surreptitious Software », Addison-Wesley, 2009

Malicious Reverse Engineering



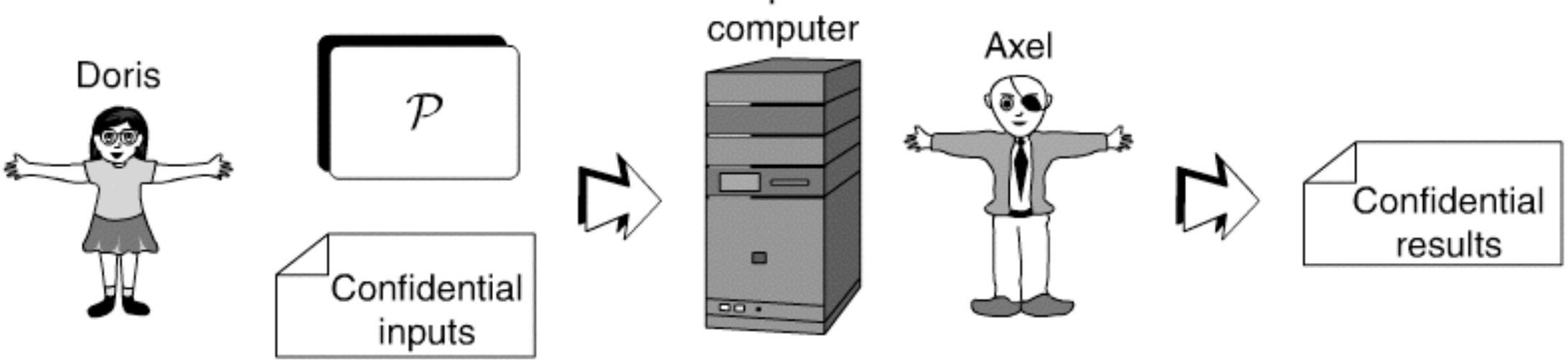
Source: Collberg, Nagra, « Surreptitious Software », Addison-Wesley, 2009

Digital Rights Managements



Source: Collberg, Nagra, « Surreptitious Software »

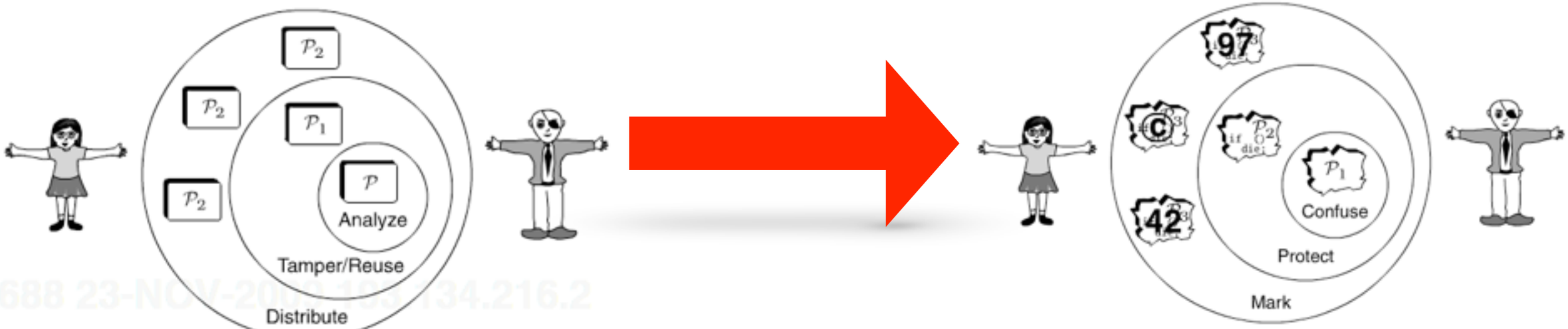
Cloud Computing



Source: Collberg, Nagra, « Surreptitious Software », Addison-Wesley, 2009

Software Protection

Man at the End



Source: Collberg, Nagra, « Surreptitious Software », Addison-Wesley, 2009

Existing Defenses

- Obfuscate the code, to slow down a reverse engineer

- Tamper-proof the code, to prevent rogue modification of the executable

- Mark the executable, to identify illegal distribution

Thank You !