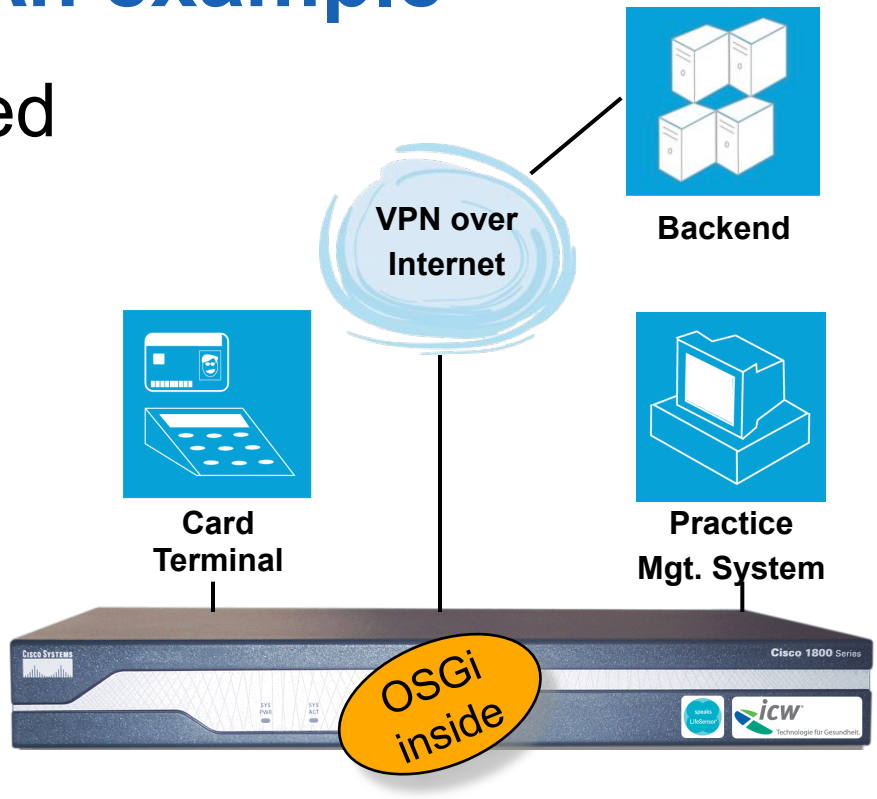

OSGi Alliance
June 10-11, 2008 Berlin, Germany
icw®
Technology for Health
OSGi Alliance
Community Event
Do not disturb my circles -
Application isolation with OSGi
Mirko Jahn*, Boris Terzic, Markus Gumbel*
OSGi Community Event 2008

# Agenda

- Motivation
- Java Security shortcomings
  and what OSGi can add
- What is still missing:
  Concept for bundle-like domain security
- Implications
- Outlook

# Why more security? An example

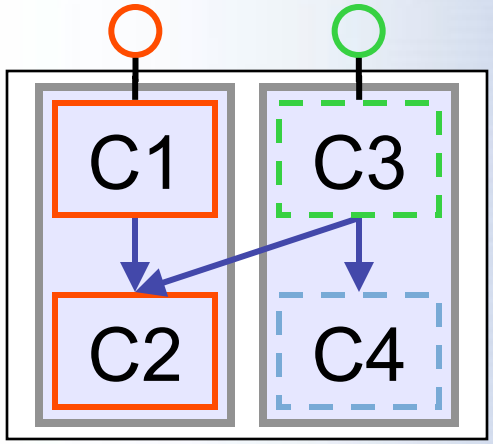
- Germany's internet-based e-health-infrastructure project
- Infrastructure has to be very secure
- Business logic (e.g. prescription workflow) requires digital signatures




"ICW healthcare connector based on Cisco AXP"

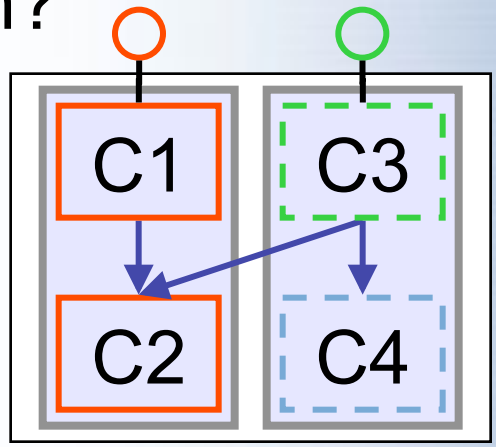
# Secure and certified applications

- Two types of applications
  - "Must-have" (and highly secure)
  - on demand
- Reuse (3rd party) components
- Contradiction
  - Certification requires a minimal and inflexible but at least an isolated system
  - Added value applications require installation of new and potentially malicious software



# Application domain separation

- How to isolate an application domain?
  - OS level (e.g. virtual OS)
  - Process level
    (different native applications)
  - ☑ Component level (within one JVM)




- Holds true for other domains as well
  - e.g. automotive sector
  - Resource-friendly
  - Less complex and easier to maintain

Fort Knox, Kenntucky – Screenshot taken from google maps.

# Plain Java™ 2 Security

- Enforced by the JVM
  - Loadtime: Code verifier
  - Runtime: ClassLoader and SecurityManager
- customizable security through extension of `java.security.Permission`
- ProtectionDomain objects as main instance to handle roles and permissions
- Policy files as configuration entity

# Shortcomings of plain Java™ 2 Security

- Not focused on “component” isolation, but application isolation (like applets)
- No notion of service level security
- No actual notion of lifecycle or dynamism

From Clay Bennett: http://www.claybennett.com/pages2/security.html (slightly adapted ;-) )

# What OSGi has to add?

- Finer grained isolation with new permissions
  - `PackagePermission` (import, export)
  - `BundlePermission` (provide, require, host, fragment)
  - `ServicePermission` (register, get)
  - `AdminPermission` (execute, lifecycle, resolve,...)

  ➔ Better encapsulation of components
- Dynamic management of policies/ permissions
  - Permission Admin (`org.osgi.service.permissionadmin.*`)
  - Conditional Permission Admin (`org.osgi.service.condpermadmin.*`)

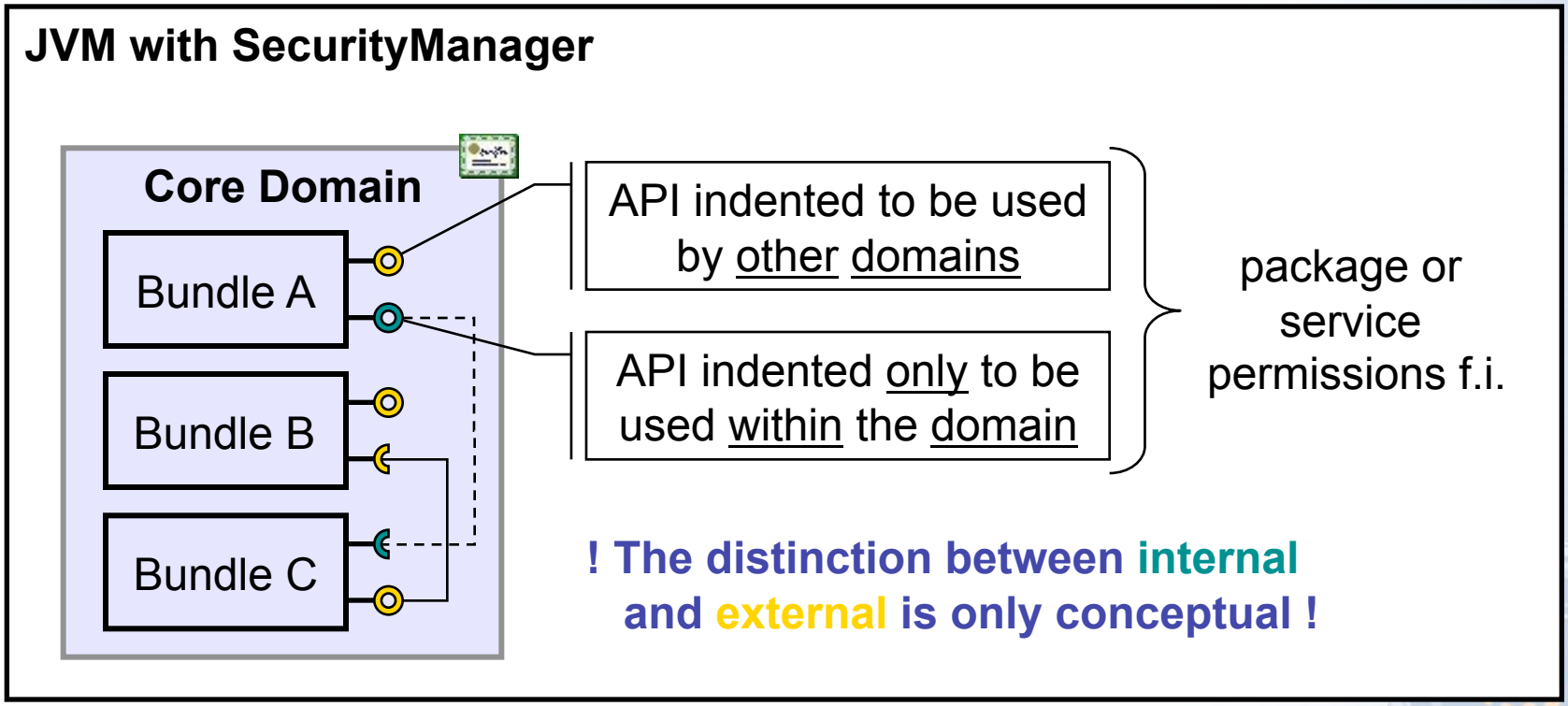
# How to isolate your code? Identify…

- … application domains and sub-domains
- … intra and inter domain boundaries
  - The minimal set of Permissions each bundle needs (explicitly via Java API calls)
  - Import and export statements in the manifest files
  - Services
  - Lifecycle dependencies
  - System (sockets, file access, properties,…)

# How to isolate your code? Assign…

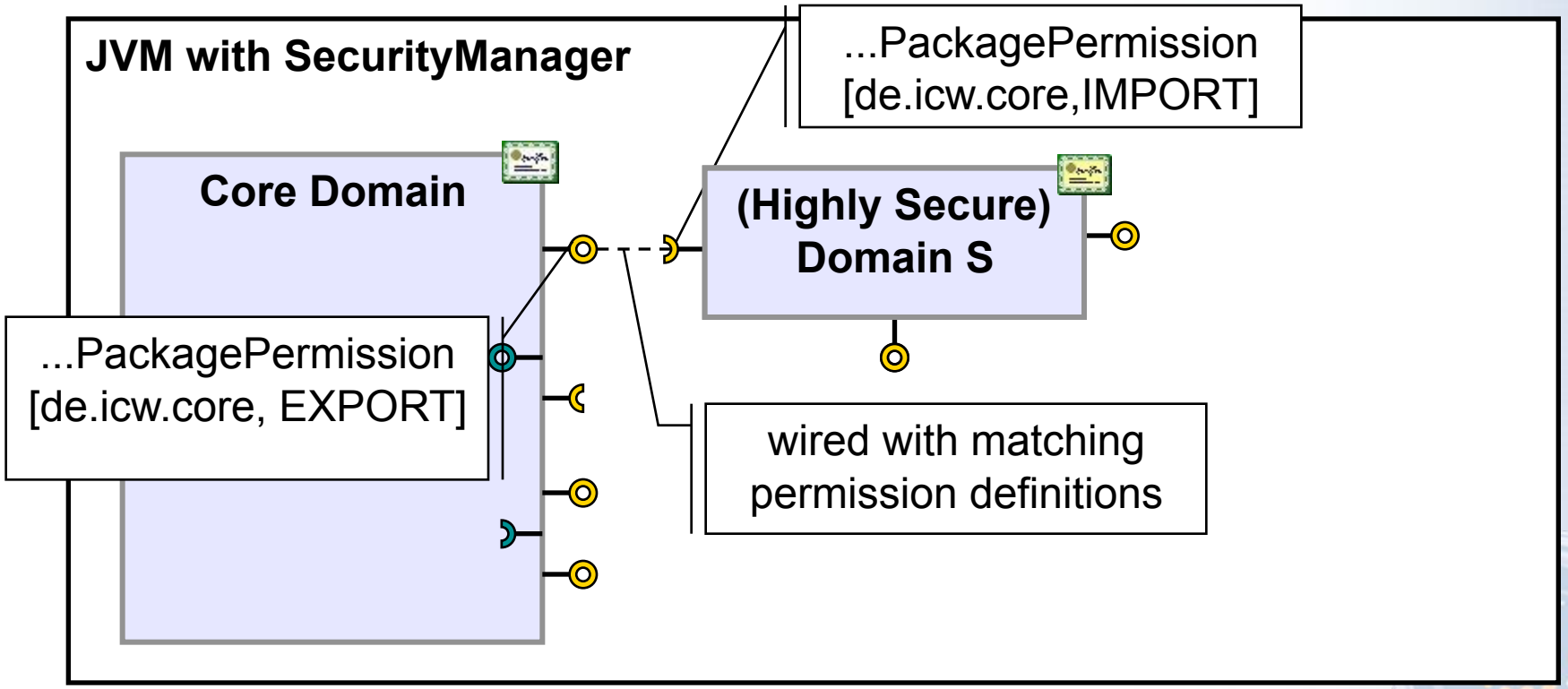
- … different certificates for each domain/ sub-domain
- … permissions on these certificates based on the findings before (including intra and inter domain dependencies)
- … scoping limit for each bundle (only the actual permission needed by the bundle) – use the `permission.perm` file

# Define your domain application(s)



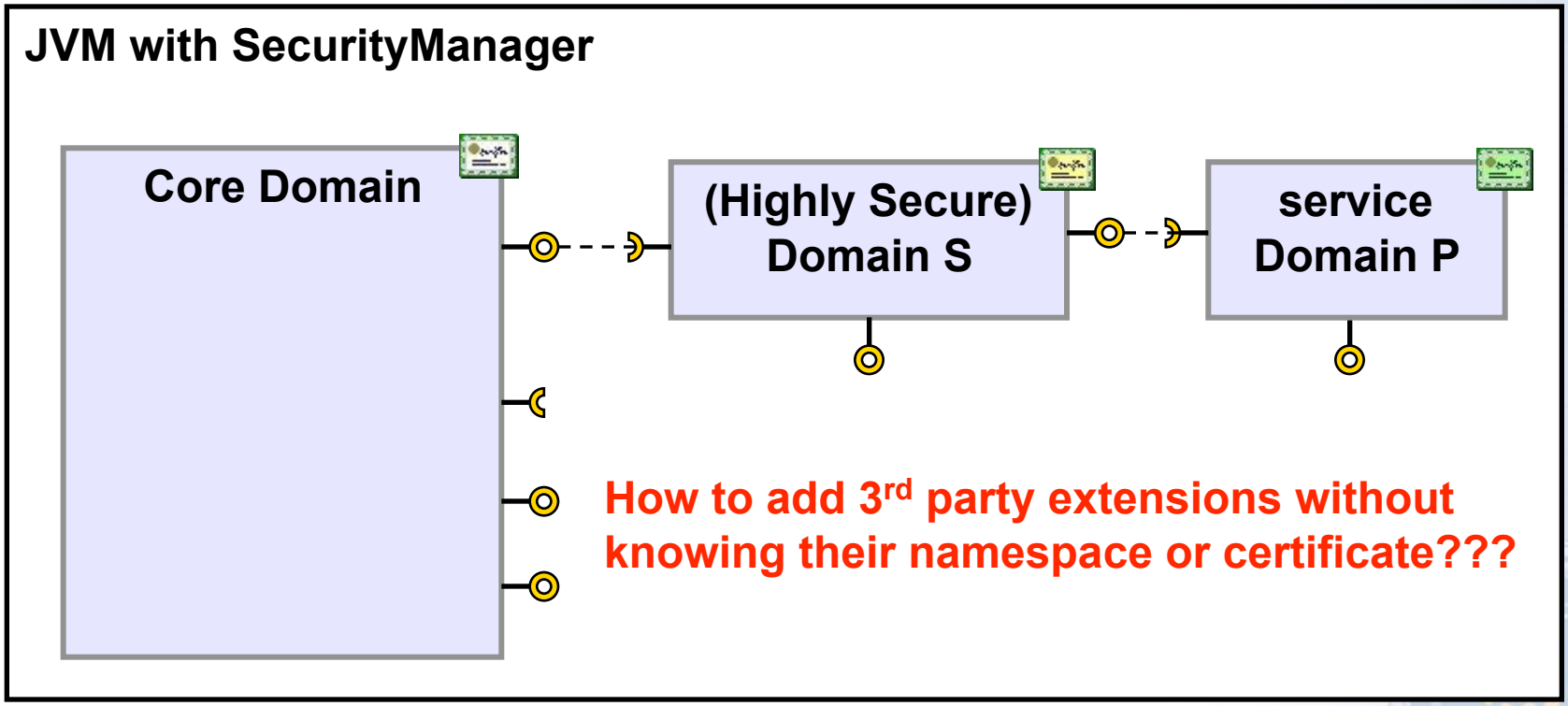

Public API export
Public API import
Intra domain API export
Intra domain API import

# How to wire your domains?




Public API export
Public API import
Intra domain API export
Intra domain API import

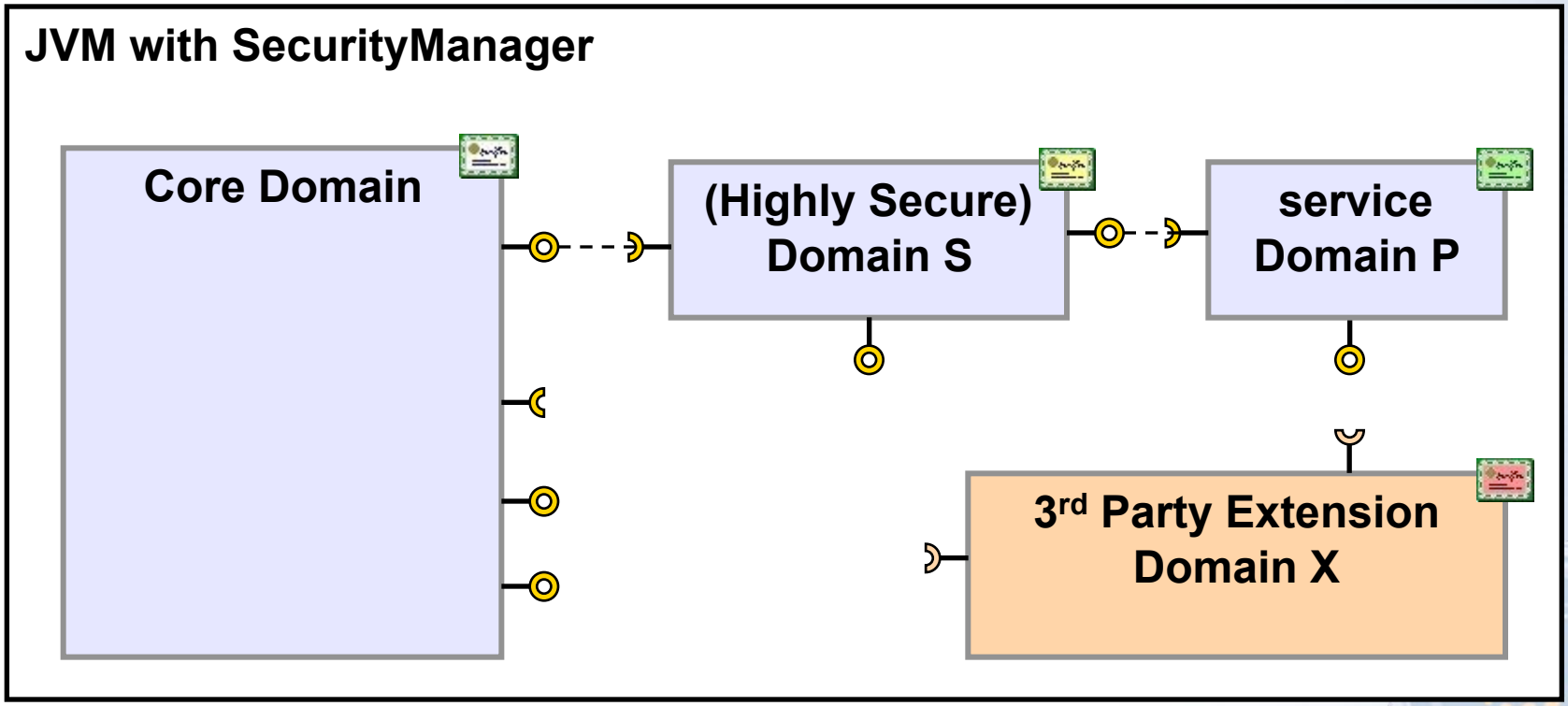
# How to wire your domains?




Public API export

Public API import

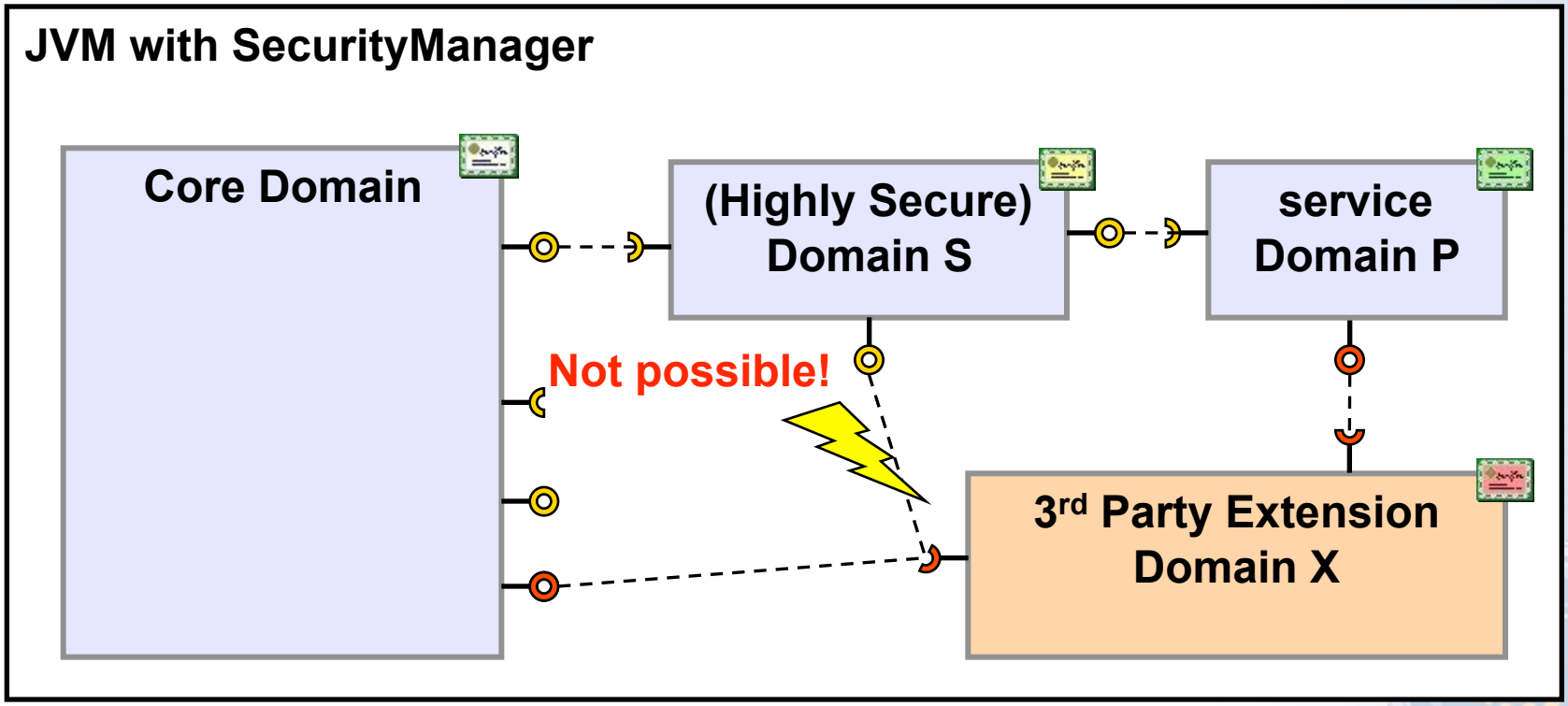
# How to integrate 3rd party extensions?




Public API export

Public API import

# How to integrate 3rd party extensions?




➔ Now, only the exposed API is security relevant!

# Run & Refactor

- Run usually reveals a huge amount of missing permissions
- Refactor
  1. Investigate missing permissions
     ➔ either add permissions or use doPrivileged()
  2. Examine permissions assigned to the certificates for undesired overlaps or dependencies.
     ➔ move or refactor/ redesign
  3. API analysis on tainted parameters based on permissions assigned to the certificates

# Remaining Challenges

- How to dynamically assign permissions on domains without prior knowledge of their features and without potentially compromising installed and certified domains/ applications?
- How to handle differing permissions depending on the version of a component?
- What about start-up behavior? First come, first serve? How to define, which bundle is the one to set permissions?
- How bullet proof is your OSGi container? Who can tell?

➔ A whitepaper on (OSGi) security is desirable

icw®
Technology for Health


# Q & A

OSGi Community Event 2008

# References


- [1] Secure Code Guidelines (Sun) http://java.sun.com/security/seccodeguide.html
- [2] Security Documentation (Sun) http://java.sun.com/javase/6/docs/technotes/guides/security/
- [3] Coding for least privilege http://en.wikipedia.org/wiki/Least_privilege
- [4] InterComponentWare AG (ICW) http://www.icw-global.com
- [5] Cisco AXP Router and the ICW Box http://www.cisco.com/en/US/prod/collateral/routers/ps9701/data_sheet_c02_459078.html
- [6] JSR 294 – Improved modularity support in the Java Programming Language http://jcp.org/en/jsr/detail?id=294
- [7] Spring Application Platform http://www.springsource.com/web/guest/products/suite/applicationplatform

# Trademarks & Copyrights

- **Java** is a trademark of Sun Microsystems, Inc. in the United States and other countries.
- **OSGi** is a trademark of the OSGi Alliance.
- All other trademarks mentioned are trademarks of the respective owners