PaaS Security, Hazards in Multitenant Software Platforms

Dr. Luis Rodero-Merino
Avalon Group, Laboratoire de l'Informatique du Parallélisme
INRIA

13th December, 2010

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

New Security Concerns in PaaS

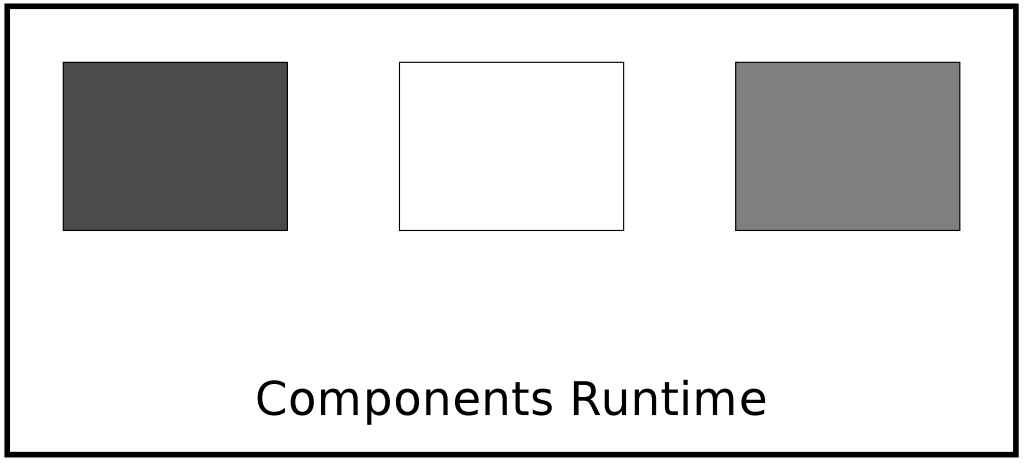
- **Platform-as-a-Service** clouds provide a runtime for users' applications/components

New Security Concerns in PaaS

- **Platform-as-a-Service** clouds provide a runtime for users' applications/components
- **Security** is one of the main concerns for cloud users

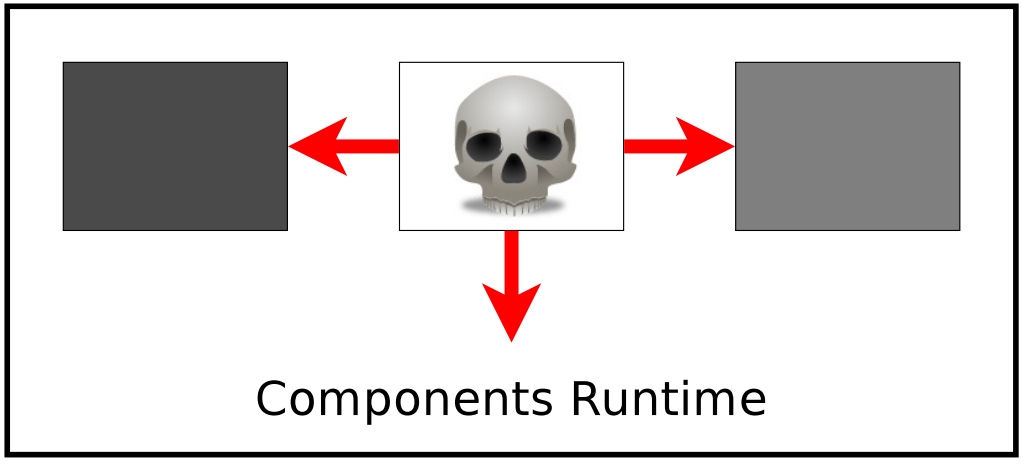
New Security Concerns in PaaS

- **Platform-as-a-Service** clouds provide a runtime for users' applications/components
- **Security** is one of the main concerns for cloud users
- New security challenge in PaaS clouds:
  - Hosting of ***potentially malicious or faulty code***



New Security Concerns in PaaS

- **Platform-as-a-Service** clouds provide a runtime for users' applications/components
- **Security** is one of the main concerns for cloud users
- New security challenge in PaaS clouds:
  - Hosting of ***potentially malicious or faulty code***
  - Must protect the platform and other users' components



Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Standard Java Security

- Bytecode verification (at load-time and runtime)
  - Checks correct bytecode format, possible memory violations, valid type conversion...

Standard Java Security

- Bytecode verification (at load-time and runtime)
  - Checks correct bytecode format, possible memory violations, valid type conversion...
- Class loaders
  - Avoid namespace confusions
  - Avoid impostor code to replace parts of the runtime or legitimate code (*spoofing*)

Standard Java Security

- Bytecode verification (at load-time and runtime)
  - Checks correct bytecode format, possible memory violations, valid type conversion...
- Class loaders
  - Avoid namespace confusions
  - Avoid impostor code to replace parts of the runtime or legitimate code (*spoofing*)
- Access control checking
  - Code centric
  - User centric (JAAS API)

Standard Java Security

- Bytecode verification (at load-time and runtime)
  - Checks correct bytecode format, possible memory violations, valid type conversion...
- Class loaders
  - Avoid namespace confusions
  - Avoid impostor code to replace parts of the runtime or legitimate code (*spoofing*)
- Access control checking
  - Code centric
  - User centric (JAAS API)
- APIs for encryption (PKI), secure communication...

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Open Security Issues

- Herzog, Shahmehri Problems Running Untrusted Services as Java Threads

Open Security Issues

- Herzog, Shahmehri Problems Running Untrusted Services as Java Threads
- **Isolation**:
  - Visibility of object references
  - Blocking through static synchronized methods

Open Security Issues

- Herzog, Shahmehri Problems Running Untrusted Services as Java Threads
- **Isolation**:
  - Visibility of object references
  - Blocking through static synchronized methods
- **Resource Accounting**:
  - Resources are limited (CPU, memory...)
  - Once access to some resource is granted, it can be used with no limitation

Open Security Issues

- Herzog, Shahmehri Problems Running Untrusted Services as Java Threads
- **Isolation**:
  - Visibility of object references
  - Blocking through static synchronized methods
- **Resource Accounting**:
  - Resources are limited (CPU, memory...)
  - Once access to some resource is granted, it can be used with no limitation
- **Safe Thread Termination**:
  - There is no safe way to terminate a Java process
  - `java.lang.Thread.stop()` is deprecated (it unlocks monitors, and so it can leave objects in inconsistent state)

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

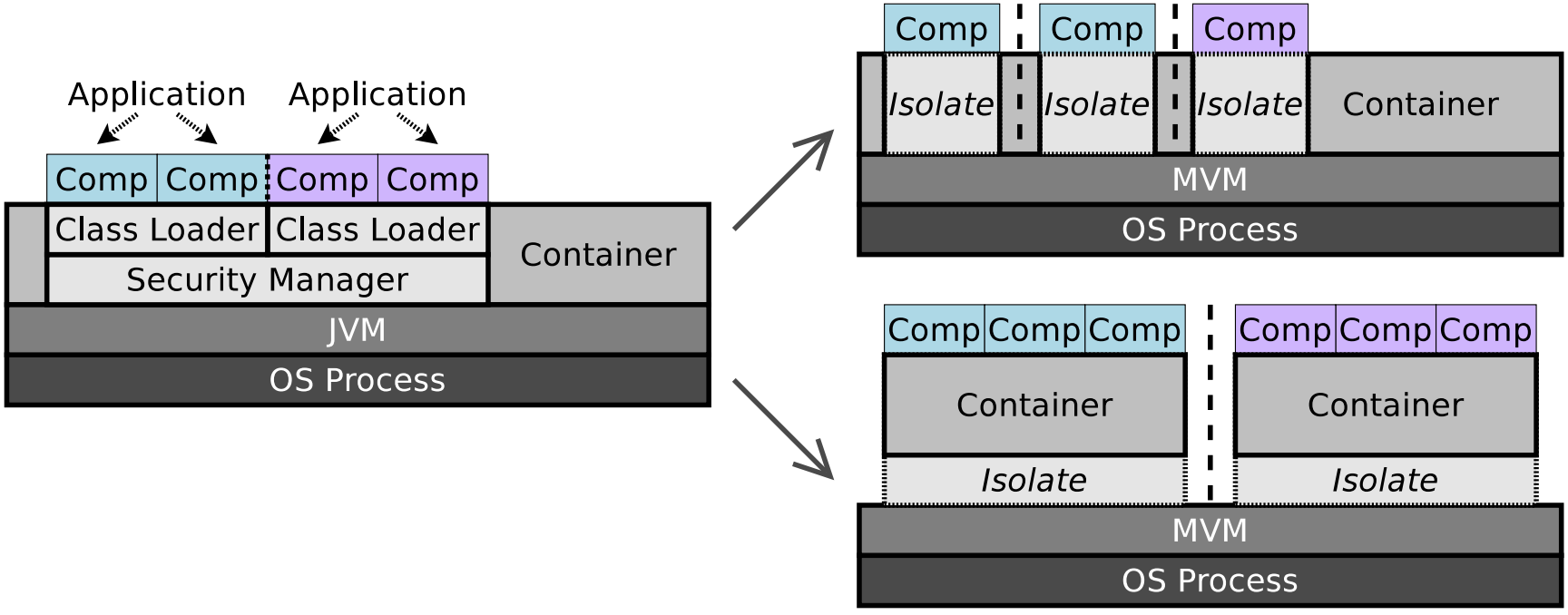
4 Conclusions
- Summing Up
- Conclusions

Solutions for Isolation: Multi-tasking Virtual Machine

- Introduces the concept of *isolate* (set of threads)
  - Non-shared heap, static vars or `Class` instances

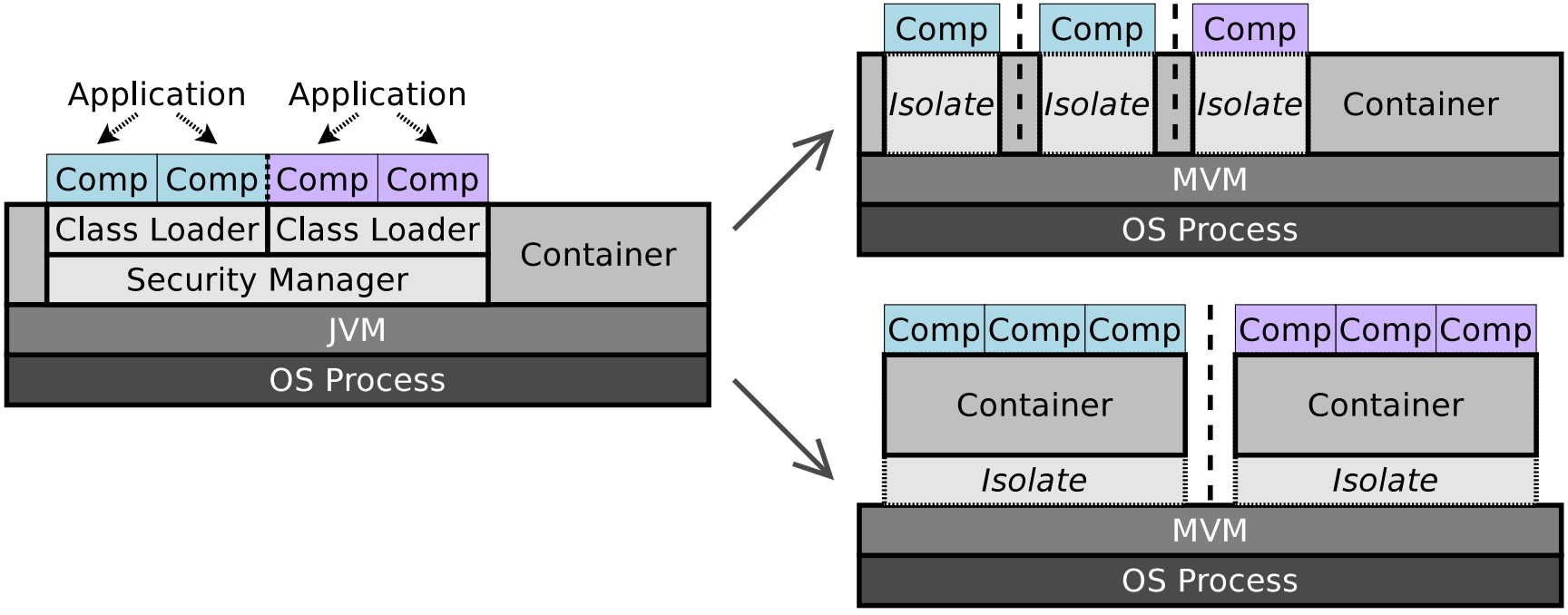
Solutions for Isolation: Multi-tasking Virtual Machine

- Introduces the concept of *isolate* (set of threads)
  - Non-shared heap, static vars or `Class` instances



Solutions for Isolation: Multi-tasking Virtual Machine

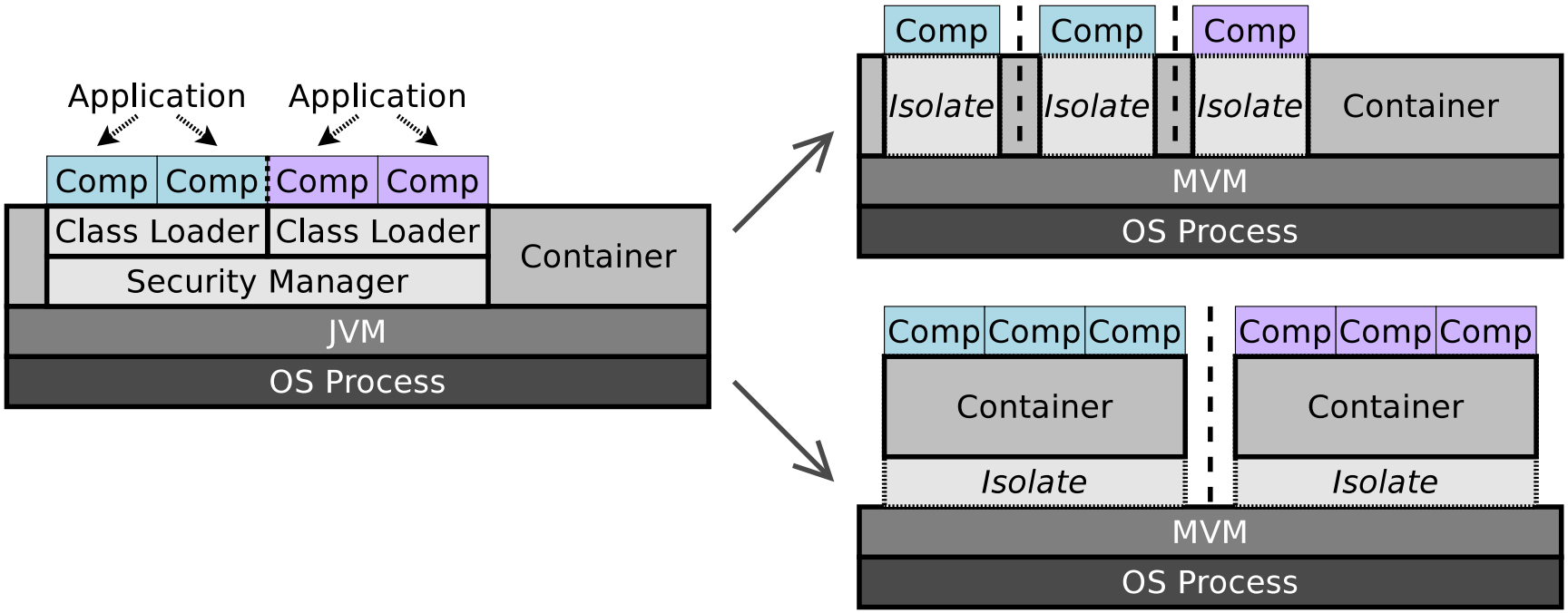
- Introduces the concept of *isolate* (set of threads)
  - Non-shared heap, static vars or `Class` instances



- Leaded to the JSR 121 Isolation API, two implementations:
  - RI based on collaborating JVMs (one per OS process)
  - MVM itself (all isolates in one single JVM)

Solutions for Isolation: Multi-tasking Virtual Machine

- Introduces the concept of *isolate* (set of threads)
  - Non-shared heap, static vars or `Class` instances



- Leaded to the JSR 121 Isolation API, two implementations:
  - RI based on collaborating JVMs (one per OS process)
  - MVM itself (all isolates in one single JVM)
  - New version of MVM "*ready*" from Sun/Oracle, but...

Solutions for Isolation (II): KaffeOS & I-JVM

- KaffeOS
  - Based on OS-like *processes*
  - Implements mechanism for communication among processes by safe *shared heaps*

Solutions for Isolation (II): KaffeOS & I-JVM

- KaffeOS
  - Based on OS-like *processes*
  - Implements mechanism for communication among processes by safe *shared heaps*
- I-JVM
  - *Isolates* based on sw modules (*bundles*). It is possible for threads to traverse them
  - An isolate's object can be passed to other in method call
  - Isolates share objects
  - Avoids the inter-isolates communication overhead

Solutions for Isolation (II): KaffeOS & I-JVM

- KaffeOS
  - Based on OS-like *processes*
  - Implements mechanism for communication among processes by safe *shared heaps*
- I-JVM
  - *Isolates* based on sw modules (*bundles*). It is possible for threads to traverse them
  - An isolate's object can be passed to other in method call
  - Isolates share objects
  - Avoids the inter-isolates communication overhead
- Jnode? JX?
  - They provide isolation too!
  - But they are Java-based OSs, not only modified JVMs. Hard to think about them as PaaS environments

Solutions for Resource Accounting

- JVM Tooling Interface
  - It can be used to inspect the usage of resources

Solutions for Resource Accounting

- JVM Tooling Interface
  - It can be used to inspect the usage of resources
  - Overhead, breaks Java portability (native code)

Solutions for Resource Accounting

- JVM Tooling Interface
  - It can be used to inspect the usage of resources
  - Overhead, breaks Java portability (native code)
- JRes, JRAF-2 apply bytecode rewriting to inject accounting

Solutions for Resource Accounting

- JVM Tooling Interface
  - It can be used to inspect the usage of resources
  - – Overhead, breaks Java portability (native code)
- JRes, JRAF-2 apply bytecode rewriting to inject accounting
- MVM-related framework: Resource Management API
  - Applied to clusters of VM instances
  - Leaded to JSR 284: Resource Consumption Management API
  - Implemented, unknown if it will be added to standard Java

Solutions for Resource Accounting

- JVM Tooling Interface
  - It can be used to inspect the usage of resources
  - – Overhead, breaks Java portability (native code)
- JRes, JRAF-2 apply bytecode rewriting to inject accounting
- MVM-related framework: Resource Management API
  - Applied to clusters of VM instances
  - Leaded to JSR 284: Resource Consumption Management API
  - Implemented, unknown if it will be added to standard Java
- KaffeOS accounts CPU time and mem
  - Support for more resources can be added

Solutions for Resource Accounting

- JVM Tooling Interface
  - It can be used to inspect the usage of resources
  - – Overhead, breaks Java portability (native code)
- JRes, JRAF-2 apply bytecode rewriting to inject accounting
- MVM-related framework: Resource Management API
  - Applied to clusters of VM instances
  - Leaded to JSR 284: Resource Consumption Management API
  - Implemented, unknown if it will be added to standard Java
- KaffeOS accounts CPU time and mem
  - Support for more resources can be added
- I-JVM accounts resources used per bundle
  - CPU, mem, threads, connections., bytes r/w

Solutions for Safe Thread Termination

- Solved in MVM and KaffeOS, as it is possible to stop isolates and processes

Solutions for Safe Thread Termination

- Solved in MVM and KaffeOS, as it is possible to stop isolates and processes

- In I-JVM, isolates are stopped by `StoppedIsolateException`
  - It cannot be caught inside the isolate, i.e. it cannot be ignored
  - But threads can traverse isolates, so synchronized entities outside the isolate can be left in an inconsistent state

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Security in J2EE Containers

- EJBs
  - Restricted environment: an EJB cannot modify class loaders, access to files, access non-static fields, create threads...
  - Security enforced by standard security mechanisms

Security in J2EE Containers

- EJBs
  - Restricted environment: an EJB cannot modify class loaders, access to files, access non-static fields, create threads...
  - Security enforced by standard security mechanisms
- Servlets
  - Apart from support for authentication and SSL, there is little about security enforcement

Security in J2EE Containers

- EJBs
  - Restricted environment: an EJB cannot modify class loaders, access to files, access non-static fields, create threads...
  - Security enforced by standard security mechanisms
- Servlets
  - Apart from support for authentication and SSL, there is little about security enforcement
- Solution proposed:
  - MVM has been applied to isolate J2EE apps
  - Each J2EE app (along its servers) are deployed on a set of isolates

Security in OSGi containers

- OSGi is based on *bundles*, components that can use/expose services
  - Each bundle decides which packages are hidden or exported
  - The container checks if bundles can access to packages and resources

Security in OSGi containers

- OSGi is based on *bundles*, components that can use/expose services
  - Each bundle decides which packages are hidden or exported
  - The container checks if bundles can access to packages and resources
- 25 security flaws detected
  - 17 can be solved programmatically
  - Remaining 8 must be addressed at JVM level (all related to isolation, res accounting and thread termination)

Security in OSGi containers

- OSGi is based on *bundles*, components that can use/expose services
  - Each bundle decides which packages are hidden or exported
  - The container checks if bundles can access to packages and resources
- 25 security flaws detected
  - 17 can be solved programmatically
  - Remaining 8 must be addressed at JVM level (all related to isolation, res accounting and thread termination)
- Solutions proposed:
  - I-JVM specifically aimed to solve OSGi security issues
  - OSGi + Isolation API on top of MVM

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Security in .NET

- **Isolation**
  - Possible through *Application Domains* (AD)
  - ADs are isolated:
    - Not possible to call code of some AD from another
    - Each AD keeps its copy of static variables
    - Not possible leaked references

Security in .NET

- **Isolation**
  - Possible through *Application Domains* (AD)
  - ADs are isolated:
    - Not possible to call code of some AD from another
    - Each AD keeps its copy of static variables
    - Not possible leaked references
- **Resource Accounting**
  - Profiling of the CLR is possible, but no generic accounting framework is available

Security in .NET

- **Isolation**
  - Possible through *Application Domains* (AD)
  - ADs are isolated:
    - Not possible to call code of some AD from another
    - Each AD keeps its copy of static variables
    - Not possible leaked references
- **Resource Accounting**
  - Profiling of the CLR is possible, but no generic accounting framework is available
- **Thread Termination**
  - `System.Threading.Thread.Abort()` can just be ignored

Security in .NET

- **Isolation**
  - Possible through *Application Domains* (AD)
  - ADs are isolated:
    - Not possible to call code of some AD from another
    - Each AD keeps its copy of static variables
    - Not possible leaked references
- **Resource Accounting**
  - Profiling of the CLR is possible, but no generic accounting framework is available
- **Thread Termination**
  - `System.Threading.Thread.Abort()` can just be ignored
- Comparatively, little research about multitenancy in .NET

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Virtual Platforms Summary

Security Feature	JVM	CLR	MVM	I-JVM	KaffeOS
Access control mechanisms	Based on Permissions and Policies	Based on Permissions and Policies	Similar to JVM	Similar to JVM	Similar to JVM
Reference leak	Not fixed	Fixed with ADs	Fixed with Isolations	Fixed with Isolations	Fixed with Processes
Shared static references	Not fixed	Fixed with ADs	Fixed with Isolations	Fixed with Isolations	Fixed with Processes
Block by synchronized static components	Not fixed	Fixed with ADs	Fixed with Isolations	Fixed with Isolations	Fixed with Processes
Thread termination	Not fixed	Not fixed	Fixed with Isolations	Not Fixed	Fixed with Processes
Resource accounting	Profiling by JVMTI. Res acc specified by JSR 284	Profiling mechanism	Generic resource manage-ment API	CPU, mem, #threads, #net conns, I/O	CPU and memory

Outline

1 Introduction: PaaS Clouds & Security Challenges
- Multitenancy-induced Risks

2 Java Platform Security
- Standard Java Security
- Open Issues: Isolation, Accounting and Thread Termination
- Solutions to Open Security Issues
- Java-based Containers: J2EE and OSGi

3 .NET Platform Security
- Isolation, Accounting and Thread Termination in .NET

4 Conclusions
- Summing Up
- Conclusions

Conclusions

- Multitenancy introduces new threats to virtual sw platforms

Conclusions

- Multitenancy introduces new threats to virtual sw platforms
- To have in mind:
  - Isolation
  - Resource accounting
  - Thread termination

Conclusions

- Multitenancy introduces new threats to virtual sw platforms
- To have in mind:
  - Isolation
  - Resource accounting
  - Thread termination
- There is not any container system that solves all issues

This was all...

Thank you!

Questions, comments?