Practical security in a modular world

Martin Toshev

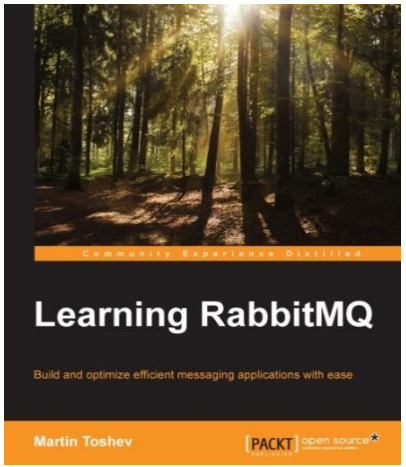
@martin_fmi

Who am I

Software consultant (CoffeeCupConsulting)

BG JUG board member (http://jug.bg)

OpenJDK and Oracle RDBMS enthusiast

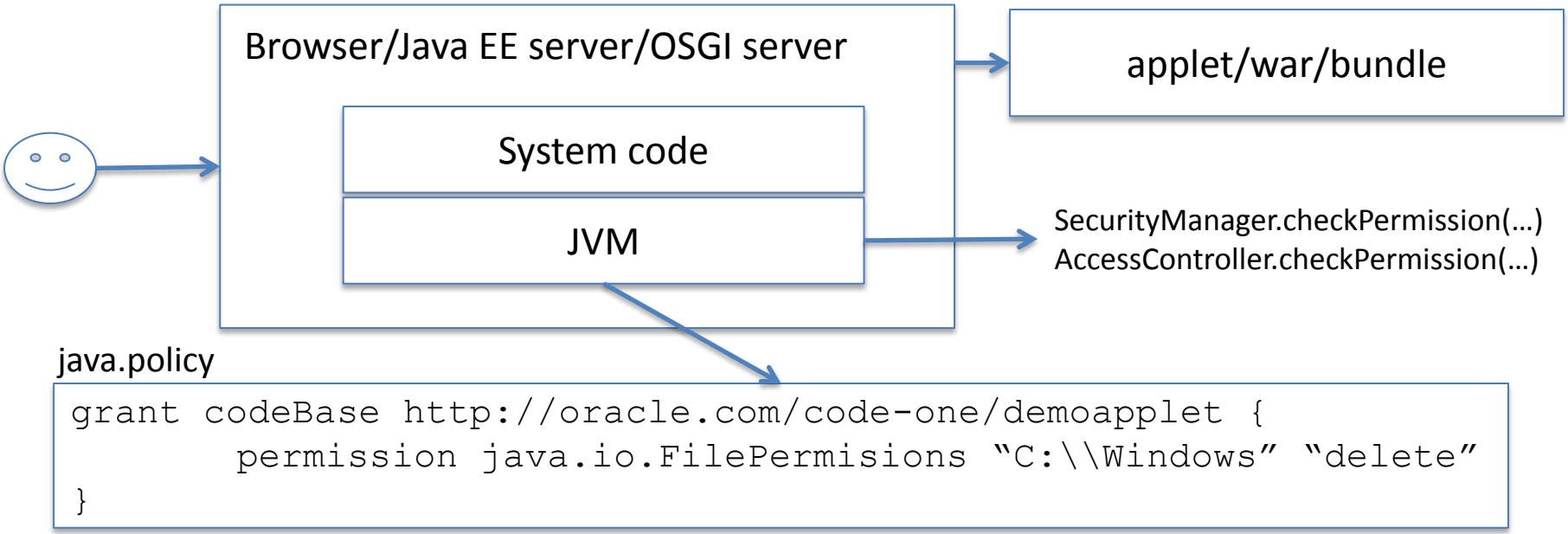


Agenda

- Security sandbox model at a glance
- Security aspects of Jigsaw
- Jigsaw vs OSGi from a security perspective

Security sandbox model at a glance

The big picture



Permission checking

- Typical flow for permission checking:

1) upon system startup a security policy is set and a security manager is installed:

```
Policy.setPolicy(…)
System.setSecurityManager(…)
```

Permission checking

- Typical flow for permission checking:

2) during classloading (e.g. of a remote applet) bytecode verification is done and the protection domain is set for the current classloader (along with the code source, the set of permissions and the set of JAAS principals)

Protection Domain

- The protection domain is set during classloading and contains the code source, the list of principals and the list of permissions for the class

```
object.getClass().getProtectionDomain();
```

- Two types of protection domain: system and application

Permission checking

- Typical flow for permission checking:

3) when system code is invoked from the remote code the SecurityManager is used to check against the intersection of protection domains based on the chain of threads and their call stacks

Permission checking

- Typical flow for permission checking:

```
SocketPermission permission = new
SocketPermission(“oracle.com:8000-9000","connect,accept");
SecurityManager sm = System.getSecurityManager();
if (sm != null) {
       sm.checkPermission(permission);
}
```

Permission checking

- Typical flow for permission checking:

4) application code can also do permission checking against remote code using a SecurityManager or an AccessController

Permission checking

- Typical flow for permission checking:

```
SocketPermission permission = new
SocketPermission("oracle.com:8000-9000", "connect,accept");
AccessController.checkPermission(permission)
```

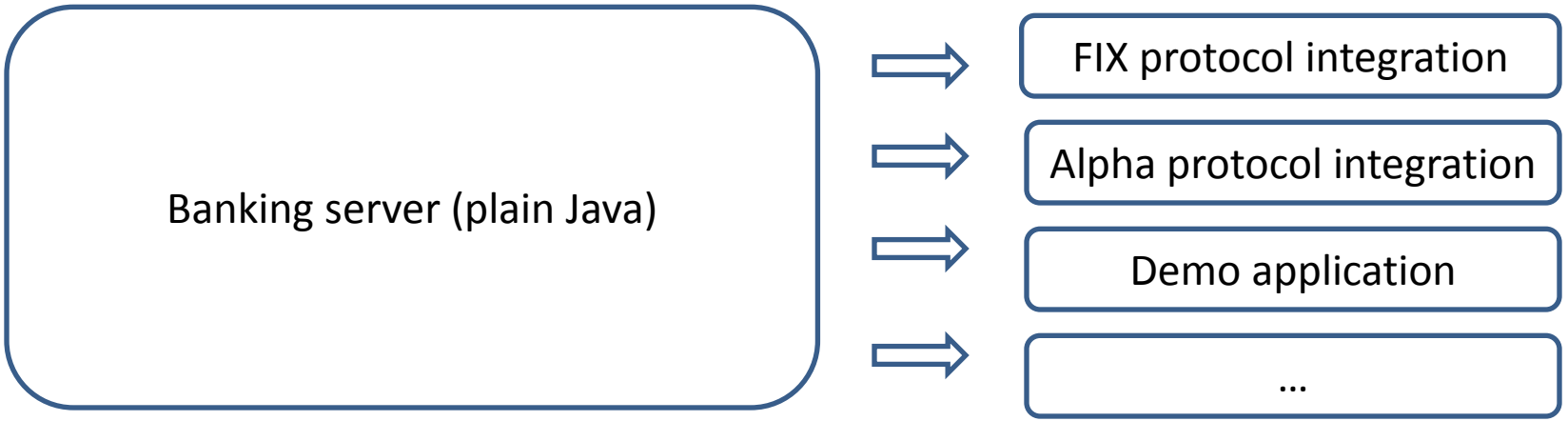
Permission checking

- Typical flow for permission checking:

5) application code can also do permission checking with all permissions of the calling domain or a particular JAAS subject

```
AccessController.doPrivileged(…)
Subject.doAs(…)
Subject.doAsPrivileged(…)
```

Example: banking app server

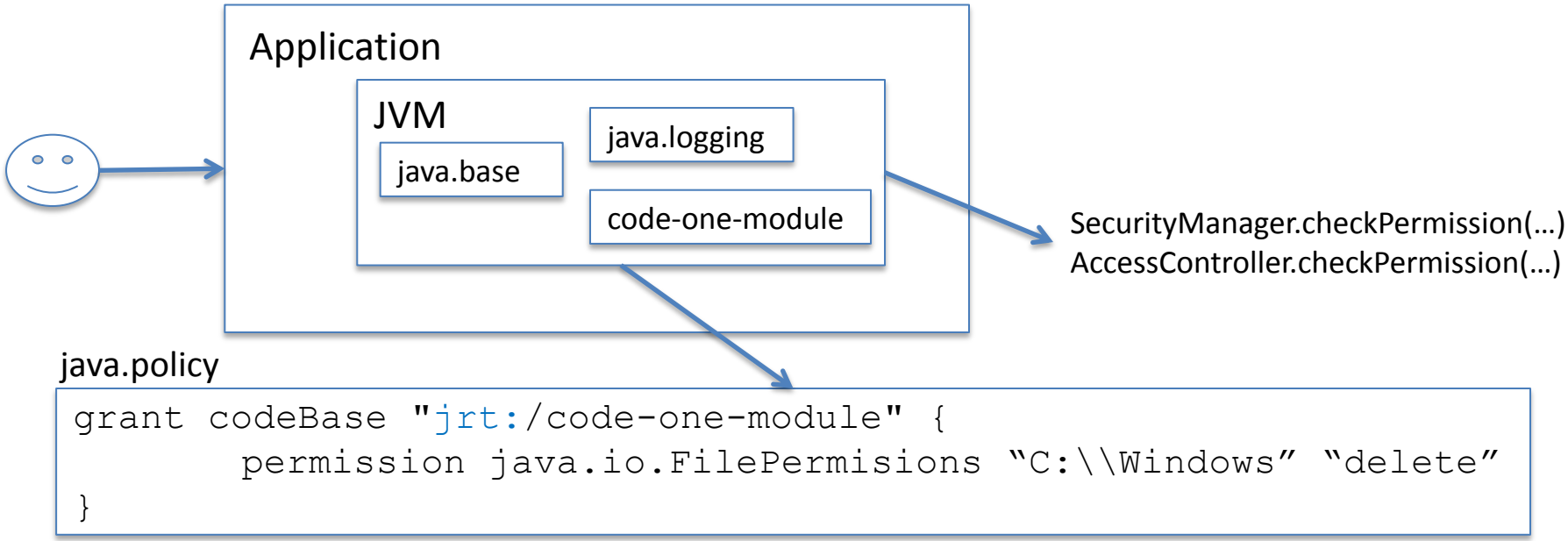


Security sandbox model at a glance

(demo)

Security aspects of Jigsaw

The big picture



Security implications

- The security model remains the same with Java modules
- System code is split into modules and applications can use a stripped down VM => improved security
- Application code can be split into modules with stronger encapsulation at runtime => improved security

Access control

- Access control is governed not by the class loader(s) of the module's classes but by the module itself

- Access modifiers are fulfilled by another layer of encapsulation: exported/opened packages

Runtime modules

- Modules can also be defined at runtime with multiple classloaders and grouped into module layers for that purpose:

```
obj.getClass().getModule().getLayer().defineModulesXXX(…)
```

Security aspects of Jigsaw

(demo)

OSGi vs Jigsaw
from a security perspective

OSGi security model

- An extension of the Java security model

- The OSGi spec provides a set of custom permissions such as PackagePermission (in order to specify whether a bundle exports/imports a package) or ServicePermission (to get or register an OSGI service)

OSGi security model

- The PermissionAdmin and ConditionalPermissionAdmin classes provide additional permission management on top of SecurityManager

- Local permissions can be specified for each bundle in **OSGI-INF/permissions.perm** and are useful for bundle security auditing

OSGi vs Jigsaw

- Both a Jigsaw module and an OSGi bundle have a distinct protection domain that defines the set of permissions for the Jigsaw module/OSGi bundle

- Both a Jigsaw module and an OSGi bundle can be signed and the set of permissions can be defined on the signer of the Jigsaw module/OSGi bundle

OSGi vs Jigsaw

- A Jigsaw module doesn’t have the notion of “local permissions” as an OSGi bundle

- A runtime Jigsaw module can have classes from multiple classloaders that have different protection domains

Summary

- The new module system in Java brings better security while still fitting in platform's security architecture

- The new module systems introduces yet another layer of access control for applications

Thank you !

Q&A

demos: https://github.com/martinfmi/practical_security_in_a_modular_world

References

Java platform security architecture
http://docs.oracle.com/javase/7/docs/technotes/guides/security/spec/security-spec.doc.html

Java Platform Module System (JSR 376)
http://openjdk.java.net/projects/jigsaw/spec/