Group 06
SOSEC Assignment 3

Java Security Architecture (JVM, Manager, APIs)

2020-09-30

Ranjana Ghimire
Justas Narusas
Efstathios Psyllakis
Oskar Elfving Söderström
Motti Aimé

Time Allocation

Task	Hour allocation / person
Zoom discussion meeting for team charter and finalising the team charter on 09-18-2020 at 11:00	1 hr (Completed)
Going over the Java Security Architecture	2 hr (Completed)
Study design security principles	30 min (Completed)
Identify the design patterns	2 hr (Completed)
Overview and discussion of the assignment	2 hr (Completed)
Team meeting with Alan presenting all the findings on 09-22-2020 at 15:00	2 hr (Completed)
Documentation of the overall work on 09-22-2020 at 15:00	2 hr (Completed)
Finalising the documentation on 09-22-2020 at 20:00	30 min(Completed)
Total Time Spent:	**12hr**

Signees

1. Ranjana Ghimire
2. Justas Narusas
3. Efstathios Psyllakis
4. Oskar Elfving Söderström
5. Motti Aimé

JAVA security architecture

Intent: The intent of models following Java security Architecture is to provide a platform where all kinds of sources can execute without imposing any risks to the users addressing the risks imposed via connection to the internet.

Context: Java Security Architecture addresses the problem of unauthorized users gaining access into the computer from the network via execution of malicious codes.

Problem: There are two types of sources for programs, trusted and untrusted. The trusted sources ensure that such a program does not employ risk into the environment and users are assured of such sources. But what about untrusted sources, we still need those sources for various programs. To be aware of such sources, we have antivirus programs and monitoring which is tedious work. This is where Java Security Architecture triumphs.

Description: Using Java Security Architecture, a sandbox is created where all the activities connected to the internet are performed. This means, sandbox will be creating its own demilitarized zone blocking access into the local environment to the outsiders.

Consequences: Addresses security issues by denying read and write access to local disk, loading external libraries, creating new processes and denying network connection beside the host computer. The restriction is that all the network related activities must be done within the sandbox and intruders might pose risk via bypassing the sandbox. *(Java Security Architecture: -, 2002) (Security Developer Guide, 2019)*

	Intent	**Context**	**Problem**	**Description**	**Consequences**
Access control Mechanisms and Algorithms	To be able to decide whether the access to the resource should be permitted or denied.	Any application where protection must be enforced.	Security manager has difficult in determining if security policy should be on effect	The location of where Java classes are obtained is encapsulated. Requests to perform specific tasks are encapsulated, granting permissions to code source is limited to specific number of codes.	Access to the resources needs to be predefined.
Secure Class loading	To be able to install software components at runtime.	Initial setup/execution of any application.	Solves security attacks at the run-time checks of the classes which defines class object permissions while locating and fetching class files.	The classes are loaded dynamically only once and can be loaded from a defined remote location. The class loaders are definable and can be created with degree of isolation.	Same class can not be loaded multiple times.

Security Management	To be able to check currently used policies and checks on access control.	Any running applet embedded into browsers and in local code	Applets are embedded into the browsers which can not be constrained access into system resources	The security manager acts as a centralized point to handle security tasks of any classes.	Permissions are not set by default, thus might pose right in even reading the file or writing to the file as a potentially dangerous operation.
Guarded Object and SignedObject	To be able to provide access control mechanism for scenarios where resource user and resource supplier threads are different	Any running application	Difficulty for consumer thread to provide supplier thread with access control information	The consumer will check access control via guarded object provided by supplier	Certain information can be lost when creating guarded objects.
Permission classes & Security policy	To be able to define the right permission for accessing system resources and have them documented in a policy	Initial class definition and single sign on authenticator	Solves the problem of programs being able to access more system resources than needed which could lead to compromised availability, integrity and availability of data.	If a program needs to access i.e. a configuration file, it needs to have access to read the file system. But that access can be limited to only one particular file saved in one particular location. Restricting access to all other files and directories on the system.	Permissions need to be defined properly to make sure access is granted to all resources needed for the application to function properly and not restricting access to necessary resources.

Security Design Patterns

Going through the JAVA Security Architecture, we were able to identify the following patterns which we have aligned them to Java Security objects/methods .

Security Pattern	**Implementation in JAVA**	**Elaboration**
Privilege Separation *(Dougherty, 2009)*	GuardedObject	If a subject needs to access an object, it requests it from the supplier, which has access rights to the object. The supplier fetches the objects and embeds it in a *GuardedObject* and presents only that to the subject. The supplier also creates a *Guard* for the *GuardedObject* with security checks that have to be met for obtaining the object. This provides a separation between code running with higher privilege from the code running on lower privilege *(Security Developer Guide, 2019).*
Authorization *(Rosado et al., 2006)*	SignedObject/JAAS	The authorization occurs via signed key in Java and JAAS, which ensure that the authorized object has required rights and permissions *(Security Developer Guide, 2019).*
RBAC *(Rosado et al., 2006)*	JAAS	It is used in the JAAS service in order to assign permissions to roles *(Security Developer Guide, 2019).*
Secure Logger pattern *(Dougherty, 2009)*	Keytool	Keytool is used to certificates and keys and protects them with a password *(Security Developer Guide, 2019).*
Reference monitor*(Rosado et al., 2006)*	Security manager	Security manager acts as a monitoring tool as well as controlling parameters for all the accesses *(Security Developer Guide, 2019).*
Execution Domain Pattern*(Rosado et al., 2006)*	Java Virtual Machine	The virtual machine ensures a protected area where all the execution can occur, creating its own execution domain *(Security Developer Guide, 2019).*
Authorization Pattern*(Rosado et al., 2006)*	Permission classes & Security Policy	Security policy created a list of policies through which all the authorization occurs and classes are defined in an encapsulated manner which defines that only the called objects are loaded based on access modifier *(Security Developer Guide, 2019).*

Security Design principles

1. **Least common mechanism**
   According to the Sandbox model, the local code can gain access to vital system resources compared to remote code that can access only limited sources provided in the sandbox. *Access Controller* ensures all the permissions and requests are encapsulated allowing for least common mechanism implementation. *(Security Developer Guide, 2019)*

2. **Complete mediation**
   Security Manager acts as a central control and delegating point for all the permissions in the JAVA applet. This ensures that all the privileges pass through the *security manager* before execution of any task thus ensuring Complete Mediation. *(Security Developer Guide, 2019)*

3. **Fail-safe defaults**
   The applets to be loaded are turned off by default and only after specifically granting permissions, the applets start to perform secured action. Access controller allows only required applets to pass by default, Class definition ensures proper calling of functions and Security manager applies restricted access by default. All these come together to provide a fail-safe default. (*CS551: Problem Set 4 Sample Answers*, 2020)

4. **Least privilege**
   The doPrivileged method ensures that a minimal amount of privilege is assigned to invoke the methods thus ensuring Least Privilege. *(Security Developer Guide, 2019)*

5. **Open Design**
   The Java source code is made available for inspection. Anyone can download and inspect the source code for further design. The open source ensures the principle of open design. (*Can You Still Use the Java Programming Language for Free? What You Need to Know - TechRepublic*, 2019)

References

Can you still use the Java programming language for free? What you need to know—TechRepublic. (2019). Retrieved September 22, 2020, from https://www.techrepublic.com/article/can-you-still-use-the-java-programming-language-for-free-what-you-need-to-know/

CS551: Problem Set 4 Sample Answers. (n.d.). Retrieved September 22, 2020, from http://www.cs.virginia.edu/~evans/cs551/problem-sets/ps4-answers.html?fbclid=IwAR06-SRRwKArUd_Yw5krKs8cVzKMWECAYHXAG8xsVSiK5mfQAi0y6aGQkHc

Dougherty, C. (2009). *Secure Design Patterns*. 118.

Java Security Architecture: -. (2020). Retrieved September 22, 2020, from https://docs.oracle.com/javase/7/docs/technotes/guides/security/spec/security-spec.doc7.html

Rosado, D. G., Gutierrez, C., Fernandez-Medina, E., & Piattini, M. (2006). A study of security architectural patterns. *First International Conference on Availability, Reliability and Security (ARES'06)*, 8 pp. – 365. https://doi.org/10.1109/ARES.2006.18

Security Developer Guide. (n.d.). Oracle Help Center. Retrieved September 22, 2020, from https://docs.oracle.com/en/java/javase/12/security/java-se-platform-security-architecture.html#GUID-D6C53B30-01F9-49F1-9F61-35815558422B