Systematic Security Checking on OSGi Bundles for Remote Healthcare System

Jinsoo Hwang\textsuperscript{1}, Kichang Kim\textsuperscript{2}

\textsuperscript{1}Department of Statistics, Inha University, Korea
\textsuperscript{2}Department of Information and Communication Engineering, Inha University, Korea
kchang@inha.ac.kr

\textbf{Abstract.} OSGi (Open Service Gateway Initiative) is now emerging as a promising software environment for remote healthcare systems. OSGi allows dynamic program loading and incremental update resulting in a very flexible system which is essential for remote healthcare system where constant program updating is needed. This flexibility however requires a very sensitive security solution instead since remote malware can also be injected through the same mechanism. This paper proposes a technique to check the security of external bundle, the program loaded dynamically. The paper shows the details of the technique and proves its validity.

\textbf{Keywords:} Remote Healthcare System, healthcare security, remote health monitoring, OSGi security, Java security

1 Introduction

Internet is growing fast and affecting the health business also, especially in remote healthcare systems. With the development of wearable and embedded devices, the Internet now allows a full-blown pervasive healthcare environment. Small devices attached to the patient monitor everyday behavior, every detailed habit, and even every eating pattern and convey them to the central server at the remote healthcare system.

As the system gets more complicated and as more devices are invented and connected to the local patents, the flexibility of the entire healthcare system became critical in assuring the success of the system. It is because of the flux of small software programs that need to be installed in the small devices. Each medical device has its own requirement and process to capture the patient state and handling data, which means we need totally a new program for each new device. Furthermore, even for existing device, a new finding in the medical profession requires a new measurement or new application of drugs which will demand an additional or modified module to be uploaded to the medical devices. All these dynamic changes in the medical environment can be handled only by some very flexible system that can adapt itself swiftly to the changing outside world.

OSGi (Open Service Gateway Initiative) is such a platform which was invented from the beginning to support the dynamicity of the system. OSGi was originally
developed for automobile telematics. Automobile telematics is in fact very similar in its requirement to the remote healthcare system. Automobiles are constantly evolving and new technology keeps flowing into automobile industry changing the functions and structures of the automobile. OSGi is well equipped with systematic techniques called bundles to handle this dynamicity. This property of OSGi can serve the need of remote healthcare system by supporting dynamic loading of medical modules remotely. As in automobile, however, OSGi for remote healthcare system needs to provide clean security solution before it is fully employed.

We propose a technique to validate the security of the imported bundles through the checking of the call chain invoked by the bundle. Our technique checks whether the bundle in question accesses the system resource or not. If there is no system resource access, we let go of the bundle. However, if the bundle accesses system resource, we have to make sure it does not allow the access right to one who does not have the proper permission. The rest of the paper is organized as follows. Section 2 surveys related researches. Section 3 explains our technique to handle the security problem of OSGi bundles, and Section 4 gives a conclusion.

2 Related Researches

Numerous researchers have studied security problems in remote healthcare system. The security issues in the context of pervasive environment that envelopes remote healthcare system has been studied by Bhattasali et al. [1] and Kavitha et al. [2] in relation with Internet of Things that comprise it. Pervasive nature of remote healthcare system naturally allows it to be an easy target of malicious security threats. The problem of authenticity losing that can allow malicious attackers to access patient's information without valid identification has been studied by Venkatasubramaniam et al. [3]. Meingast et al. [4] examines the loss of integrity when the health related data in the central server has been modified by the attackers. Modification of patient data can potentially lead to death of the patient or at least inappropriate treatment. Effective layer-based remote healthcare system has been suggested by Bhattasali et al. [5] whose system can provide reliable, cost-effective, and still fast response time. Their system provides simple but fast authentication based on bio-authentication mechanism.

OSGi is well regarded as one of promising platforms for remote healthcare system. OSGi provides dynamic application mechanism, called bundle, to be installed, started, stopped, and uninstalled on the target remote healthcare system without rebooting the underlying Java Virtual Machine [6]. Many researchers expressed their concern on the security of this platform. Various solutions have been proposed, and an exhaustive vulnerability list has been reported in [7] and [8].

3 Systematic Security Checking on OSGi Bundle

There are three basic permissions in OSGi to protect the system resources: Bundle Permission, Service Permission, and Package Permission. OSGi validate the user
request based on these permissions. However just checking these basic permissions may not be enough since the classes inside the bundle may not follow the security guidance faithfully. It is well known that there are numerous cases of system breaches via some vulnerable Java system classes. Illegal users could access the system resource through these vulnerable classes even when all the above three permissions do not allow it. Detecting vulnerable Java classes, therefore, is important to protect the system from malicious OSGi bundles. Previous approaches on detecting vulnerable Java classes have been largely manual. We need a systematic technique to detect Java classes that can cause a security problem.

In this section, we provide an algorithm for such a systematic approach. The main idea behind our technique is to identify unsafe methods that allow system access without proper security checking. `forName()` is one such example. This method is one frequently used to load a new class into the system. Since once a class is loaded the user can call any method belonging to that class, a class with system power enables the user to manage and access system information. Because of this fact, `forName()` must be very careful not to load system classes unless the request is submitted by one with proper permission. However `forName()` is a prime example of unsafe method.

There are two kinds of `forName()`: one with a single argument, `forName(X)`, and the other with three arguments, `forName(X, false, cl)`. Both eventually calls `checkPermission()` method to validate the permission of the caller. If the caller does not have the proper permission, `forName()` will reject the class-loading request. But the path leading to `checkPermission()` has a branch that bypasses `checkPermission()`. Actually before embarking on the path to `checkPermission()`, `forName()` checks the parent caller of the current class. If the parent caller is NULL, which means the parent is a system class, `forName()` bypasses `checkPermission()`; otherwise, it follows the normal path leading to `checkPermission()`.

Why does Java system allow bypass of security checking in `forName()`? It is because of system performance. `checkPermission()` will examine the entire call stack to find out any classes on the stack who do not have proper permission for the current request. Checking all classes in the call stack whenever there is a request to load a new class puts too heavy burden on the system, especially when the depth of the stack is very deep. The Java solution for this performance issue is to omit full call stack checking when the system has a sound reason to believe there is no security threat by the request. The system believes the request is not dangerous if the request is submitted by some system class. All system classes are part of Java system and are supposed to be securely complete.

However, it has been found that there are numerous system classes which are not security complete. One example is `AverageRangeStatisticImpl` which belongs to `com.sun.org.glassfish.external.statistics.impl` package. There is a method called `invoke()` in `AverageRangeStatisticImpl` class. This method can be called by normal user (who does not have system permission) in order to invoke any method indirectly. This means the user can call `invoke()` with `forName` method as an argument, effectively invoking `forName()` through `AverageRangeStatisticImpl` class. Since `AverageRangeStatisticImpl` class is a system class, the user now is executing `forName()` as a system class and bypasses the full security checking. It is clear the blame should be on `AverageRangeStatisticImpl`, because this whole problem is caused by `AverageRangeStatisticImpl` which does not perform proper security checking.
However the attack was possible because `forName` has provided bypassing of security checking in the beginning. The basic idea of our solution is to construct call chain for all classes in a given bundle. Unsafe methods are pre-known, and we search all dangerous methods that exist on paths reaching to unsafe methods.

4 Conclusion

OSGi enables a highly flexible and constantly updatable remote healthcare system by allowing dynamic program loading. This dynamic feature however requires rather complex security checking measures. Previous researches on OSGi security concentrate on bundle permission issue or security policy enforcement problem. However permission checking at bundle level or policy enforcement at top level is not enough to provide necessary security for OSGi-based remote healthcare system. This paper provides a technique to detect security problem in a given bundle before it is injected to the remote healthcare system. The technique builds a call tree for all classes in the bundle and traces any suspicious methods that can possibly reach so-called unsafe methods. For these dangerous classes the system examines whether the class has proper security checking code in it.

Acknowledgments. This work was supported by INHA UNIVERSITY Research Grant.

References