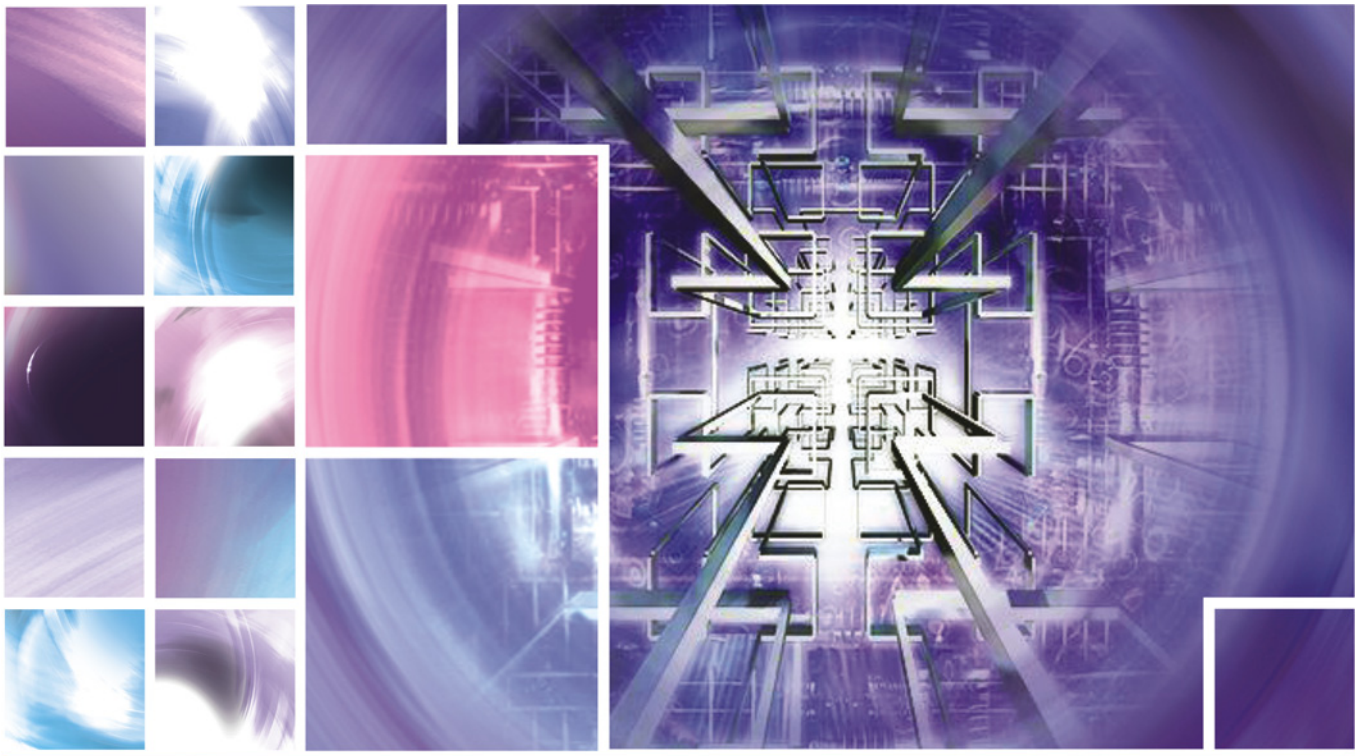


Two Ways of Securing Eclipse RCP Applications

How to protect IP and meet safety-critical requirements

Excelsior White Paper

EXCELSIOR

Table of Contents

Executive summary

Risk of unintended "open source"

Eclipse Rich Client Platform (RCP) applications are implemented in the Java™ programming language and distributed in a portable binary format called Java bytecode. Unlike native binaries created by C++ compilers, platform-neutral Java bytecode is easy to reverse-engineer and there are a number of freely available Java decompilers that can produce amazingly readable source code.

> "Anyone who has worked on a Java project based on the Eclipse Rich Client Platform (RCP) knows how powerful and valuable RCP can be. Unfortunately not all developers realize that there is little protection for their code and embedded resources in the deliverable content they distribute. With the abundance of knowledge and tools for Java disassembly, anyone can obtain a detailed understanding of the inner workings of and intellectual property contained within an RCP application with very little effort."
>
> --Eric Byres, CTO, Byres Security Inc.

Uncontrolled access to source code poses a risk of Intellectual Property (IP) theft, software piracy and undesirable exposure of security vulnerabilities, which may result in revenue loss for vendors of commercial Eclipse RCP applications.

This is especially important for the companies that offer solutions for safety critical industries such as Transport, Energy, Finance, Healthcare, Defense, and others. Apart from protecting their own IP, such companies must comply with the customers' security requirements.

Solutions

Companies that want to manage their risks are challenged to protect their RCP applications before deployment to end users. This can be done in two ways:

- *Obfuscation* – modifying bytecode in order to impede decompilation and make the decompiled source code less comprehensible.
- *Native pre-compilation* – transforming the application classes into a native code executable and distributing it without the original Java bytecode.

It is important to choose the right technology and tools that are compatible with the Eclipse RCP, taking into account Total Cost of Ownership (TCO), application security requirements, and high risk application assets.

This whitepaper assesses the strengths and limitations of the available solutions as applied to Eclipse RCP applications.

Securing Eclipse RCP: technologies and tools

Bytecode obfuscation

In essence, obfuscators take Java bytecode (class/jar files) as input, transform it, and save the result again as class/jar files. The transformations used include, in particular:

- *control flow obfuscation* - insertion of non-structured bytecode operators that do not alter program behavior but prevent decompilers from restoring the original source code; and
- *name obfuscation* - assigning meaningless names to classes, methods, and fields, which otherwise appear verbatim in the decompiled code

At a glance, obfuscation looks like a simple extra step in the established build process and the availability of free mature obfuscation tools makes this option attractive to many developers. However, there are some caveats.

Caveats

Effective control flow obfuscation is not a good choice for performance-critical code: it may degrade performance by an order of magnitude. In addition, name obfuscation is not safe in the general case because Java reflection would stop working if all class, method, and field names are altered by an obfuscator. Note that the Eclipse Runtime heavily uses reflection when loading plug-ins and the names declared in OSGi bundle manifests -- imported and exported packages – may not be obfuscated. That said, obfuscation of Eclipse RCP applications is not a straightforward process, and the obfuscator's TCO may unexpectedly increase, especially for applications being actively developed. The underlying reason is that the developers have to manually configure such tools to use obfuscation selectively so as to not compromise application performance and/or integrity.

Yet, the decompiler's inability to fully recreate the compilable source code does not necessarily reduce the risk of IP leaks and application vulnerability exposure. Unfortunately, having certain fragments of the original code produced by a decompiler may be enough to allow tampering and IP theft.

Also, obfuscators cannot protect data files that are distributed with RCP applications. Configuration files, XML/XMI data, such as model specifications used by the Eclipse Modeling Framework, and other application resources are left intact by obfuscators.

Despite these limitations, bytecode obfuscation has value because it delivers some protection against full decompilation. If you are considering using an obfuscator, you may learn more from the materials referenced below.

Further reading

ProGuard [1] is a mature free obfuscator that is often recommended. The article [2] covers bytecode obfuscation pros and cons in more detail and includes references to a spectrum of other obfuscation tools, both free and commercial. The tutorial [3] presents step-by-step instructions on configuring bytecode obfuscators for use with Eclipse RCP applications. Finally, FernFlower [4] is a next-generation analytical decompiler that copes with many flow obfuscation tricks, available on-line as free Software As a Service.

AOT compilation

Native pre-compilation, also known as Ahead-Of-Time (AOT) compilation, involves the translating of Java bytecode to native (platform-specific) code before deploying the application. AOT compilation systems take class/jar files as input and generate a native code executable linked with a runtime library providing memory management, threading, etc. This technique is similar to what C/C++ compilers do, but adapted to Java: the runtime library must be, in essence, a complete Java Virtual Machine (JVM) supporting such language features as reflection and dynamic loading of non-compiled bytecode, should it occur at run time.

Benefits

The strength of AOT compilation is high resistance of the application code to reverse-engineering:

- The RCP applications are distributed *without* Java bytecode, because the compiled executables run directly on hardware.
- A highly optimizing AOT compiler *completely* prevents automatic decompilation of native code back to source code.

One of the reasons for this strength is that code optimizations implicitly create a complex many-to-many relation between Java source code and generated native code. For example, the same Java expression may appear in the resulting executable as dozens of different assembly patterns, and vice versa, the same CPU instructions may be emitted for different Java expressions. This effectively makes reverse-engineering very expensive and time-consuming.

Note also that unlike bytecode flow obfuscation, AOT compilation does not compromise application performance.

Issues to consider

Using AOT compilation for the protection of Eclipse RCP applications requires careful consideration.

- *Standards compliance.* The implementation of the AOT compilation system must support all Java features and APIs required by the Java SE platform specification.
- *Platform dependence.* The RCP application should be compiled and distributed separately for each platform. This may be less of an issue for vendors who already offer separate product downloads for each platform, for example, due to the use of platform-specific application installers.
- *Protection capabilities.* Part of the application code may be loaded with Java custom classloaders, for instance, Eclipse plug-ins are handled by the Eclipse Runtime classloaders. Not all AOT solutions may correctly pre-compile, and therefore, protect such code.
- *Integration cost.* AOT compilation tools should be easy to use with RCP applications and integrate into the automated build and testing process. Otherwise, the TCO of such a solution may be prohibitive for use in production development.

The IBM® J9 JVM included with the IBM® WebSphere® Real Time product, has an AOT compilation option to improve start-up time and guarantee more deterministic behavior of Java applications [5]. However, by design, it stores the native compiled code with the original bytecode in optimized jar files thus providing no code protection.

GCJ, the GNU compiler for Java, supports AOT compilation and is available for free [6]. Whether the GCJ implementation complies with the Java specification remains uncertain and the use of the GNU Classpath libraries puts it well behind the current Java SE 6 in terms of platform API support. As reported in [7], GCJ was able to compile the Eclipse IDE, but this was a very old version not based on Equinox OSGi. To date, there is no evidence or information proving that GCJ can be used for compiling Eclipse RCP applications.

Native pre-compilation for the Eclipse RCP is fully supported in Excelsior JET [8], a complete Java SE 6 VM with an AOT compiler.

Excelsior JET solution

Excelsior JET, a Java SE 6 VM with an AOT compiler and installation toolkit, provides special support for the Eclipse RCP with a focus on code and data protection. It enables developers to export the entire RCP application, including the Eclipse Runtime and plug-ins (OSGi bundles), into native code form and distribute the application without the easy-to-hack jar files.

> "As pioneers in the area of static Java compilation Instantiations is delighted to see Excelsior JET expand into the Eclipse universe. It will be useful to many Eclipse developers as they move their software toward deployment"
>
> --Mike Taylor, CEO Instantiations, Eclipse Foundation Board Member

Product maturity

Entering the market in 2000, Excelsior JET is now used by hundreds of customers, from corporations to individual programmers, in over 60+ countries.

Excelsior JET has passed the official Java Compatibility Kit (JCK) test suite and is certified Java Compatible on a number of platforms. The JCK conformance testing guarantees that all language features and APIs defined in the Java SE platform specification are supported and their implementation complies with all Java standards.

Eclipse RCP support

Excelsior JET enables the protection of Eclipse RCP applications by supporting the core of the Equinox OSGi Runtime at the JVM level: the loading of OSGi bundles is performed by a JVM layer in co-operation with the Eclipse Runtime.

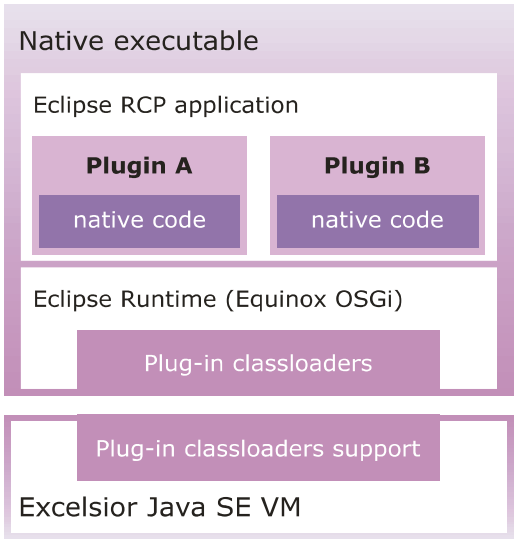


Fig1. JVM-level support for Equinox OSGi

The key point is that this architecture allows AOT compilation of plug-ins and distribution of the RCP application without jar files, which otherwise would not be possible due to Eclipse's custom classloading policy.

At the same time, Excelsior JET provides flexibility: the developers may opt to native compile only certain plug-ins, leaving the rest "as is" (in the form of jar files containing OSGi manifests). This also means that new plug-ins can be installed into a pre-compiled RCP application as usual.

Such close integration of OSGi and Java SE Runtimes creates a secure environment for running Eclipse RCP applications. On the one hand, it keeps sensitive code protected from decompilation and prevents tampering with OSGi bundles as the Eclipse Runtime itself is also pre-compiled. On the other hand, it supports the loading new plug-ins, for example, those created by third parties.

Established protection features

The following table summarizes the features implemented in Excelsior JET to provide code and data protection for Eclipse RCP applications.

Feature	Benefit
AOT compilation of OSGi bundles (plug-ins)	Protects sensitive Java code from decompilation by translating it to optimized native code
	Enables distribution of RCP applications without jar files
AOT compilation of Eclipse Runtime	Protects the Eclipse Runtime from unauthorized modifications by translating it to native code linked with the compiled RCP application
Tamper notification	Detects attempts to replace pre-compiled plug-ins in the deployed RCP application
Encryption of string literals and reflection data	Safely scrambles string literals and names of classes, methods, and fields so that the executable no longer includes them in their original form (co-operative with the Excelsior JVM to make Java reflection work without limitations)
Packing resource files into the compiled executable	Hides sensitive data such as configuration files, XML/XMI data and other application resources by packing them into the resulting executable
Encryption of resource files	Safely scrambles the packed resource files so that the executable no longer includes them in their original form (co-operative with the Excelsior JVM to decrypt the resources on demand at application run time)
Generation of trial versions	Creates time limited versions of RCP applications to ease distribution of free trials to potential customers

Integration costs

To estimate the cost of integrating the Excelsior's AOT compilation technology into your application lifecycle, you need to consider the learning curve of the Excelsior JET product and identify how your established processes will be impacted.

Excelsior JET is more about deployment than about development: it compiles an exported RCP application ready for distribution, which is the last step before final Quality Assurance (QA). Therefore, it results in a minor impact on the development process: developers continue using the same tools and practices for design, coding, debugging, and testing.

> "Excelsior JET's AOT compilation of Java bytecode makes the process extremely simple and effective - there was no easier way to protect our RCP application from reverse engineering. Once Excelsior JET was used on our ready-for-release bytecode, our QA team was able to demonstrate that the resulting application met or exceeded all required functions with only two issues that were resolved in less than a day."
>
> --Michael Thomas, Lead Developer, Byres Security Inc.

Excelsior JET includes standalone GUI tools to setup build projects and a command line interface for use with build automation systems like Apache Ant. The basic functionality is also available in the Eclipse plug-in for Excelsior JET [9].

In addition, the AOT compiler preserves the file structure of the exported RCP application by simply replacing the standard RCP launcher with the native compiled executable:

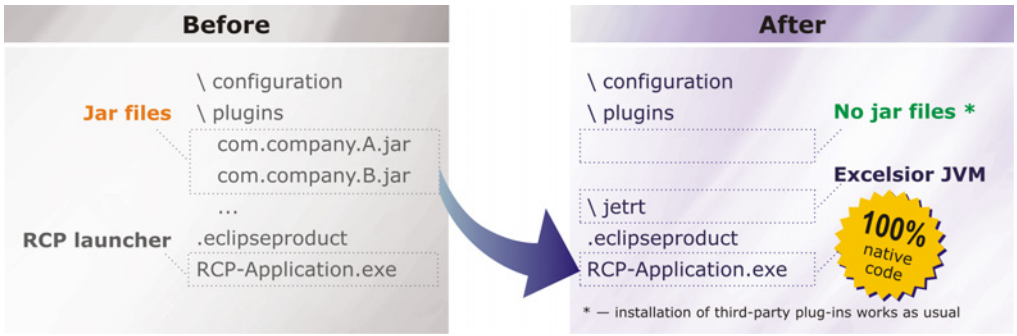


Fig2. Structure of exported Eclipse RCP applications

This kind of "mimicry" eases integration of Excelsior JET into existing build and QA processes.

Noteworthy is that the native compiled RCP applications remain fully interoperable with testing tools including those for automatic SWT GUI testing, such as QF-Test [10], which is extremely beneficial for final QA.

Lastly, usage of Excelsior JET for Eclipse RCP applications is very straightforward as illustrated by this video tutorial [11]. It takes just a few minutes to learn all steps required for AOT compilation and preparing the compiled application for deployment.

Platform support

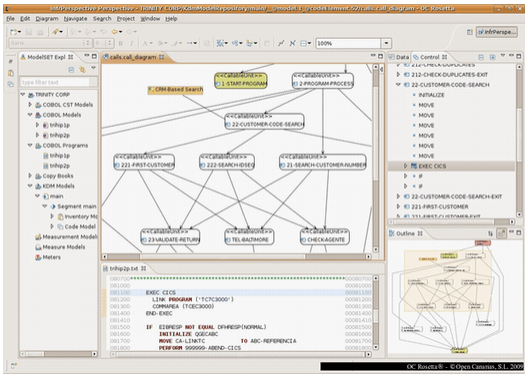
Excelsior JET supports Java SE 6 on Microsoft Windows and Linux operating systems running on 32-bit Intel x86 and compatible hardware. The compiled applications also work in 32-bit compatibility mode on 64-bit x86 systems. The Equinox OSGi runtime versions 3.1 through 3.6 are supported.

Customer examples

Below are given a few real world examples of using the Excelsior JET technology for securing commercial RCP applications.

Open Canarias S.L.

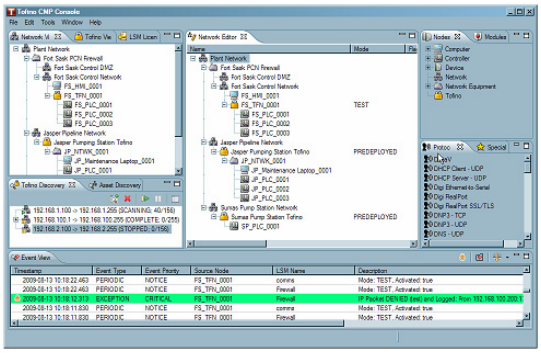
OC Rosetta ®, a tool for modernization driven by legacy system models, is built on top of the Eclipse RCP framework and EMF. With more than 22,000 Java classes (not including those of Eclipse and EMF), 371 bundles, and managing large-sized models, OC Rosetta represents a complex RCP application and provides quite a challenge for any protection scheme involving manual setup.



According to Victor Roldan Betancort, Software Developer at Open Canarias S.L.: "Excelsior JET represents the best, and possibly only available, solution capable of protecting not only Eclipse RCP- and OSGi-based applications, but also those that rely on the modeling technologies available in the Eclipse platform."

Byres Security Inc

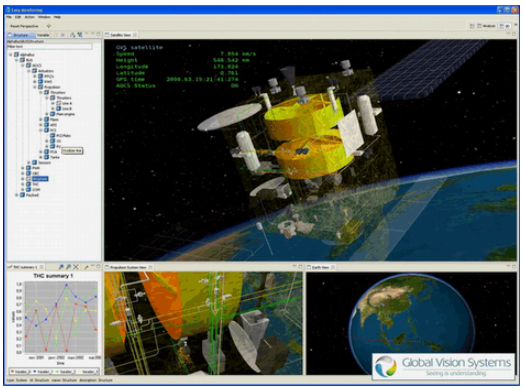
The Tofino Central Management Platform product is used for managing security appliances in critical Supervisory Control And Data Acquisition (SCADA) applications such as nuclear power plants and oil pipelines. This package is RCP-based, since it allows flexibility in adding plug-ins.



“When it comes to Eclipse RCP applications, our research indicated that Excelsior JET was really the only code protection solution for our RCP based product.” – said Eric Byres, CTO, Byres Security Inc. – “It is also the best solution we could hope for. We recommend the use of Excelsior JET to anyone developing a Java application containing sensitive code or vital intellectual property which must be protected.”

Global Vision Systems

EasyMonitoring is a real-time 3D visualization product based on the Eclipse RCP. The product is also used as a platform for developing customer tailored plug-ins for analyzing 3D and real-time data.



"We have thoroughly tested Excelsior JET and it works perfectly with such a complex RCP application as EasyMonitoring" - said Baptiste Gendron, President of Global Vision Systems – "We are mainly using it to protect the source code against reverse engineering, and to improve ease of use by not having to deploy the JRE. We are amazed by what the Excelsior team has achieved and we highly recommend the use of their product for deploying RCP applications in a secure and easy way."

Onendis Systems

Smart Positioning Architect is an RCP-based tool for private use to design, configure and calibrate the environment for Wi-Fi real time location systems. It also provides a real time and on-demand visualization of locations of enterprise assets like objects, people, and mobile devices. Using this tool, an enterprise have a global visibility of what happens and can improve its workflow.

"With Excelsior JET, we can distribute our system without the fear of somebody is going to decompile it" – said Mikel Hernando, Software Engineer at Onendis Systems. "Moreover, we can send time limited demos of our system. This enabled us to start distributing demos of our software to anybody who demand it, thus increasing our potential customer base and benefits."

Learn more

You may find more information about Excelsior JET for the Eclipse RCP, including

- Pre-compiled RCP applications, e.g. the Eclipse IDE 3.6 Classic (Helios)
- Case studies
- Flash tutorial

and download a fully functional Evaluation Package at

http://www.ExcelsiorJET.com/RCP

If you have any questions about using the Excelsior JET technology for the Eclipse RCP, contact Excelsior Engineering Team at java@excelsior-usa.com or submit a call back request using this web form

http://www.excelsior-usa.com/callme.php

Conclusion

Selecting tools and technologies for securing RCP applications is similar to selecting an appropriate insurance plan: the two main considerations are budget and need.

This whitepaper presented available solutions that are vastly different in terms of their protection capabilities. Therefore, preparing a risk plan based on the business impact of IP theft and application security vulnerabilities could help you take the right decision.

If you do not anticipate a high revenue loss due to security breaches, a simple solution like obfuscation may be a good choice. If, however, the estimated losses are significant, a complete solution encompassing code and data protection, combined with a secure runtime environment offered by native pre-compilation, is most likely the preferred strategy.

References

1. ProGuard – a Java class file obfuscator and shrinker.
   http://proguard.sourceforge.net
2. Dmitry Leskov. *Protect Your Java Code - Through Obfuscators And Beyond.*
   http://www.excelsior-usa.com/articles/java-obfuscators.html
3. Patrick Paulin. *Obfuscating an RCP Application*.
   http://rcpquickstart.com/2007/06/22/obfuscating-an-rcp-application/
4. Fernflower – a Java decompiler.
   http://www.reversed-java.com/fernflower/
5. IBM WebSphere Real Time
   http://www-01.ibm.com/software/webservers/realtime/
6. The GNU Compiler for the Java Programming Language.
   http://gcc.gnu.org/java/
7. John Healy, Andrew Haley and Tom Tromey. *Eclipse goes native*,
   Linux Journal, Issue 123, July 2004.
   http://www.linuxjournal.com/article/7413/
8. Excelsior JET – a Java SE implementation with AOT compiler.
   http://www.ExcelsiorJET.com/
9. Eclipse plug-in for Excelsior JET.
   http://www.excelsior-usa.com/rcp-plugin.html
10. QF-Test - GUI Testing for Java and Web.
    http://www.qfs.de/
11. Excelsior JET for Eclipse RCP, video tutorial.
    http://www.excelsior-usa.com/tutorials/jet/eclipse/

About Excelsior

Founded in 1999, Excelsior LLC is a software company providing advanced Java execution environments and high-quality software development services. Focusing on Java SE platform support, Excelsior delivers integrated Java-compatible solutions designed to address the needs of performance- and safety-critical application development.

Excelsior is a Java Authorized Licensee and a Solution Member of the Eclipse Foundation. The company's flagship product, Excelsior JET, improves performance, provides code protection, and eases deployment of Java applications in Microsoft Windows and Linux environments.

To learn more about Excelsior, visit http://www.excelsior-usa.com

Excelsior LLC

6 Lavrenteva Ave.
Novosibirsk 630090 Russia

Phone: +7 (383) 330 55 08
Fax: +1 (509) 271 52 05
Web: www.excelsior-usa.com