

Forum: [Hack Forums](#)

Thread: [\[WTF\] java independent programs coded in java/\[WTF\]](#)

pikasas - August 26, 2011, 11:57 pm

Hey!

Could someone explain how some java programs are java independant? For example java crypters make exe files which can be run without java.

Moreover how can java produce exe file when its executable is a jar?

Thank you for your answers.

Will and Integrity - August 26, 2011, 11:59 pm

Is this possible? I've never heard of this before...

Beemo - August 27, 2011, 12:00 am

<http://www.excelsior-usa.com/articles/java-to-exe.html>

Seriously the first link I found in a google search.

Eruditus - August 27, 2011, 12:02 am

(Aug 27, 2011 12:00 AM)Herobrine Wrote: [\[CITE\]](#)

<http://www.excelsior-usa.com/articles/java-to-exe.html>

Seriously the first link I found in a google search.

He means if he made a crypter in Java it would output an executable. Therefore giving it no Java dependency.

randomKID - August 27, 2011, 7:29 am

For some reason, this thread confuses the hell out of me. Is the OP saying that the .exe contains some Java code and somehow, the Java code can be run without having the JRE? Or is he asking how is it possible that a Java application can output a .exe file?

Also, why wouldn't a Java application be able to output a binary file? The .exe will have different data than a .jar will but as long as you can output that data, you can create an .exe file.

Ben121 - August 27, 2011, 7:45 am

If you have java byte code, you NEED the JRE to run it.

Sometimes people use native wrappers to wrap the byte code inside a native executable, but you still require the JRE.

In other cases, some people use compilers which convert byte code to native code, which doesn't require the use of the JRE.

Colby_ - August 27, 2011, 1:46 pm

tldr: Don't ever do this it's pointless.

soul_ninja - August 28, 2011, 7:27 am

i think OP isnt talking about wrappers.

i think he means a Java application Producing an EXE.

which is still a dumb question.

i mean the java program can just write the exe file in byte code.

-

he says "For example java crypters make exe files which can be run without java"

so hes asking why when a java crypter (a crypter coded in Java) can produce a crypted .exe file without any Java dependencies (eg. Java JRE)

Thats because the Java Program takes the original .exe (coded maybe in C++ or what have you) and only modifies that file to be undetected. The java program does not put any Java into the .exe file.

Virux - August 28, 2011, 8:42 am

Right. If it's encrypting another executable, it's just reading the byte stream of that file, modifying it as it exports the crypted version.

A program coded in Java can be converted to an executable, many are. The JRE compiles the Java code to native, and runs it like an exe being 'javaw.exe'(I believe that's correct). So a Java-to-Exe converter would just capture that process.

However, if Java is converted to a native executable, it will lose its cross-platform ability. It will only work on the platform it was compiled for.

soul_ninja - August 28, 2011, 1:02 pm

(Aug 28, 2011 08:42 AM)Virux Wrote: [\[CITE\]](#) Right. If it's encrypting another executable, it's just reading the byte stream of that file, modifying it as it exports the crypted version.This is correct

(Aug 28, 2011 08:42 AM)Virux Wrote: [\[CITE\]](#) A program coded in Java can be converted to an executable, many are. The JRE compiles the Java code to native, and runs it like an exe being 'javaw.exe'(I believe that's correct). So a Java-to-Exe converter would just capture that process.

However, if Java is converted to a native executable, it will lose its cross-platform ability. It will only work on the platform it was compiled for.This is very incorrect

First of all the JRE does not compile the Java code to native code.

it Run the code inside a Virtual Machine (the JVM)

There is no such thing as a Java-to-Exe converter.

There are things called Wrappers as i referred to in my previous post.

But Wrappers only take the .jar file and "Wrap" it inside an .exe

The Java JRE dependencies are still there and its still a Java program.

The difference is the .exe can check if you have Java installed on the client PC, if not. Install it. then Execute the Java Executable, this among a few other things is what Wrappers are mainly used for.

But the fact remains you cannot convert a java program into .exe program magically and think it wont have any Java Dependencies.

BECAUSE

the JRE has all the libraries that the java program uses.
thats why it is always essential that the JRE is installed when running a Program coded in java.

/rant
:)

randomKID - August 28, 2011, 8:02 pm

@soul_ninja AOT compilers compile the Java byte-code into native-code so there are Java-to-Exe converters...

<http://www.excelsior-usa.com/articles/java-to-exe.html>

<http://www.excelsior-usa.com/articles/java-to-exe.html#aot> //Or click here if you only want to read the AOT stuff.

soul_ninja - August 29, 2011, 1:33 am

(Aug 28, 2011 08:02 PM)randomKID Wrote: [\[CITE\]](#) @soul_ninja AOT compilers compile the Java byte-code into native-code so there are Java-to-Exe converters...

<http://www.excelsior-usa.com/articles/java-to-exe.html>

<http://www.excelsior-usa.com/articles/java-to-exe.html#aot> //Or click here if you only want to read the AOT stuff.

i seem to have been disproven :)
thanks for setting it straight.

so i guess my post was not necessarily needed.
Sorry Virux ?

Virux - August 29, 2011, 8:32 am

(Aug 29, 2011 01:33 AM)soul_ninja Wrote: [\[CITE\]](#) i seem to have been disproven :)
thanks for setting it straight.

so i guess my post was not necessarily needed.
Sorry Virux ?

Haha, it's alright.

But yeah, in order for the program to run, something must be executing machine code(which would be the JRE). So it's taking the .jar and converting it(virtually) to machine code, to be executed.

http://en.wikipedia.org/wiki/Just-in-time_compilation

That being said, an external program can snoop on this bytecode-to-machine conversion and use the converted bytecode for the exe.

I know this is the general idea, but I don't think the specifics are correct.

pikasas - August 29, 2011, 8:53 am

Thank you for all your posts! Everything seems great. AOT's make java programs java independant (well let's say that after crypting they are still java programs).

Unfortunately, I don't understand one more thing. I'll use a java crypter for an example (again).

Let's say that we have got a java crypter and we are crypting an exe. Woohoo it produces our crypted file BUT its java dependant (?) because the stub is written in java too (ofcourse you can use

c++ stub but what is the point in making java crypter + c++ stub).

So as I understand to make our crypted file independent from java, we have to use AOT crypter with both the java crypter and the stub?

OFF TOPIC: does java have a self-compile function as .NET does (CodeDom)?

Ben121 - August 29, 2011, 9:49 am

<http://download.oracle.com/javase/7/docs/api/javax/tools/JavaCompiler.html>

Immensify - August 29, 2011, 2:41 pm

Hmm this is very interesting. I always thought what Soul has been saying to be true. Is it really possible to convert the native code of a jar file to an executable?

Wouldn't that essentially be putting JVM-intended code in a non-JVM environment, or is the bytecode to native code mapping the last point the JVM is useful?

Also Ben we've already discussed wrappers, they're talking about Java native code to executable converters.

randomKID - August 29, 2011, 9:17 pm

@Solarsonic

In reference to your first question, if, by native-code, you mean Java byte-code, then yes. Otherwise, native code should be executable already.

I don't really understand your second question but I feel like I get the gist of it. The JVM essentially reads the Java byte-code and executes the equivalent native-code. If you simply convert the Java byte-code into native-code and store it in an executable, then you've removed the need for the JVM (which I think was your question).

Immensify - August 29, 2011, 9:34 pm

(Aug 29, 2011 09:17 PM)randomKID Wrote: [\[CITE\]](#) @Solarsonic

In reference to your first question, if, by native-code, you mean Java byte-code, then yes. Otherwise, native code should be executable already.

I don't really understand your second question but I feel like I get the gist of it. The JVM essentially reads the Java byte-code and executes the equivalent native-code. If you simply convert the Java byte-code into native-code and store it in an executable, then you've removed the need for the JVM (which I think was your question).

So you're saying that once Java bytecode is mapped to native code by the JVM it no longer needs the JVM and is essentially an executable?

So what is the point of the JVM environment then? Why not just code in a native language like C++?

randomKID - August 29, 2011, 10:00 pm

Because Java is a "write once, run everywhere" language. You write it into an "international language" (java byte-code) and then you have a "translator" (jvm) for each country that can interpret that language and translate it into the native language (native-code or machine-code). This way, you don't have to write the same program for the Windows, Unix based, etc...

Immensify - August 29, 2011, 11:43 pm

(Aug 29, 2011 10:00 PM)randomKID Wrote: [\[CITE\]](#) Because Java is a "write once, run everywhere"

language. You write it into an "international language" (java byte-code) and then you have a "translator" (jvm) for each country that can interpret that language and translate it into the native language (native-code or machine-code). This way, you don't have to write the same program for the Windows, Unix based, etc...

Ah that's a nice point. This thread was a great read, I hope to see you around here more often man :)

Ben121 - August 30, 2011, 12:35 am

(Aug 29, 2011 02:41 PM)Solarsonic Wrote: [\[CITE\]](#) Hmm this is very interesting. I always thought what Soul has been saying to be true. Is it really possible to convert the native code of a jar file to an executable?

Wouldn't that essentially be putting JVM-intended code in a non-JVM environment, or is the bytecode to native code mapping the last point the JVM is useful?

Also Ben we've already discussed wrappers, they're talking about Java native code to executable converters.

My link was in reference to "self compiling code".

Ben121 - August 31, 2011, 5:52 am

For those of you that are interested, the invocation api is part of the jni which allows you to load a VM into a native application.

Immensify - August 31, 2011, 11:17 am

(Aug 31, 2011 05:52 AM)Ben121 Wrote: [\[CITE\]](#) For those of you that are interested, the invocation api is part of the jni which allows you to load a VM into a native application.

Very interesting. Would this require the JRE to run the native application?

Ben121 - August 31, 2011, 11:20 am

(Aug 31, 2011 11:17 AM)Solarsonic Wrote: [\[CITE\]](#) Very interesting. Would this require the JRE to run the native application?

You don't need the JRE to run the native application, on windows this would just be a .exe. But you will need the JRE to load the VM as the VM is part of the JRE.

I've put together a little example which I'll post a bit later on.