

Forum:[Hack Forums](#)

Thread: [\[Tutorial\]\[Source\] Store an EXE as an image! \[Free\]](#)

MrZander - September 16, 2010, 5:50 pm

Store an EXE as an image!

Tutorial and Code by MrZander

Hello HF, here is my second tutorial on this site.

After seeing a vb.NET example of storing a string as an image, I decided to tackle this.

This is CLI, I didn't bother making a GUI.

First is our imports:

```
Code import java.awt.image.BufferedImage;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Scanner;
import javax.imageio.ImageIO;
EXE To Image
```

Now, get the location of the EXE and open read all of the bytes

This process takes a while depending on size of file!

```
Code public static void filetoimage(){
    try{
        String dir="";
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the location of your exe.");
        dir = s.next();
        FileInputStream fstream = new
        FileInputStream(dir);
        DataInputStream in = new DataInputStream(fstream);
        byte b[] = new byte[in.available()];
        int count=0;
        System.out.println("Reading bytes, please wait . . .");
        while (in.available()!=0){
            b[count]=(in.readByte());
            count=count+1;
        }
        in.close();
```

Next, get the location to save the image and create it

```
Code imagebytes=b;
        System.out.println("Enter the DIRECTORY to save your file to.");
        saveto = s.next();
        int c = 0;
        int c2=1;
        int size = (int) Math.sqrt(b.length)+2;
```

```

int r, g, b2, rgb;
BufferedImage buff = new BufferedImage(size,size, BufferedImage.TYPE_INT_RGB);
System.out.println("Creating image . . . 0/"+Integer.toString(b.length));
    for(int y = 0;y<size;y++){
        System.out.println("Creating image . . .
"+Integer.toString(c)+"/"+Integer.toString(b.length));
        for(int x=0;x<size;x++){
            if(c<b.length-1){
                r=b[c];
                g=c2;
                b2=c2;
                rgb = 0xff<<24 | r<<16 | g<<8 | b2;
                buff.setRGB(x, y, rgb); //ALL BYTE DATA IS STORED IN THE RED VALUE
                c=c+1;
                c2 = c2+1;
                if(c2 == 253){
                    c2=1; //c2 is simply used to change the color of Green and Blue, looks prettier :P
                }
            }
        }
    }Finally, save it!

```

This is obviously customizable, change the name of the executable and so on.

```

Code    System.out.println("Saving image.");
        ImageIO.write(buff, "png", new File(saveto+"image.png"));
        System.out.println("Image saved:");
        System.out.println(saveto+"image.png");
        System.out.println("Finished.");
    }
    catch (Exception e){
        System.err.println(e.getMessage());
    }
}

```

Image to EXE

Here is where the fun starts!

Retrieving the EXE from the image:

First, get the location of the image using a scanner

```

Code public static void imagetofile(){
    String dir="";
    Scanner s = new Scanner(System.in);
    System.out.println("Enter the location of your image.");
    dir = s.next();
}

```

Next, We created a new BufferedImage and populate it with the image selected. Then we get the attributes of the image (Height and Width).

```

Code try
{
    BufferedImage image = ImageIO.read(new File(dir));
    byte b[] = new byte[image.getHeight()*image.getWidth()];
    int xmax = image.getWidth()-1;
    int ymax = image.getHeight()-1;
    int count=0;
    System.out.println("Opening image. . . ");
}

```

Now for the meat of it, we read the [b]RED value of each pixel in the image.[/b]

```
Code for(int y = 0;y<=ymax;y++){
    System.out.println(Integer.toString(count)+"/"+Integer.toString(b.length));
    for(int x = 0;x<=xmax;x++){
        int pixel = image.getRGB(x,y);
        int red = (pixel >> 16) & 0xff;
        b[count]=(byte)red;
        count=count+1;
    }
}
```

And Finally!

We get the directory to export to and we write all the bytes in bytearray b[].

```
Code System.out.println("Enter the DIRECTORY to save the exe to: ");
String saveto = s.next();
try
{
    FileOutputStream fos = new FileOutputStream(saveto+"executable.exe");
    fos.write(b);
    fos.close();
}
catch(Exception ex)
{
    System.out.println("FileNotFoundException : " + ex);
}
System.out.println("File saved:");
System.out.println(saveto+"executable.exe");
System.out.println("Finished!");
}
catch (Exception e)
{
    System.out.println(e.getMessage());
}
}
```

Here is an example of hjsplit(a file splitter) stored as an Image

Spoiler ([Click to View](#))

And there we have it!

Thanks for reading and hope it helps.

FULL SOURCE:

Spoiler ([Click to View](#))<http://dl.dropbox.com/u/9086539/base.java>

The Real Kobe Bryant - September 16, 2010, 6:01 pm

There have been a lot of tutorials on this, but this is a really good one, nice job!

MrZander - September 16, 2010, 6:05 pm

(Sep 16, 2010 06:01 PM)Snoop Dogg Wrote: [\[CITE\]](#) There have been a lot of tutorials on this, but this is a really good one, nice job!

Thanks!

I searched for some, couldn't find any in Java however.

Chrollo Lucifer - September 17, 2010, 2:49 am

Nice tutorial !

I should try this. =D (Sep 16, 2010 06:05 PM)MrZander Wrote: [\[CITE\]](#) I searched for some, couldn't find any in Java however.I don't think it's a repost. Anyway, good job !

PS:

That hjsplit image is such a beautiful rainbow, hha.

!!!! - September 19, 2010, 8:19 am

So what is this actually?

is it like if I open the image file(.jpg,etc) will I open the hidden EXE?

MrZander - September 19, 2010, 11:14 am

(Sep 19, 2010 08:19 AM)Sanju Wrote: [\[CITE\]](#) So what is this actually?

is it like if I open the image file(.jpg,etc) will I open the hidden EXE?

No, it stores the bytes of an exe file in the red value of each pixel in an image, so that you can transfer the exe as an image, and later, extract it for execution.

Brutal Chaos - September 20, 2010, 10:23 am

From what I understand, this is basically a secret way of sending a .exe in a jpg file, the receiver has to then extract and voila he has the .exe.

[H]ell - September 21, 2010, 2:16 am

Wow, great tutorial!

Respect !

Im learning java in school atm. so going to be active allot in the java section ;p

Sensible - September 21, 2010, 11:14 am

Lucky [2012]! I'm stuck doing dreamweaver and learning how mice work in school. >>

on topic: Yeah, It looks pretty helpful. Although I'm fairly sure it saves it as a .png instead of a .jpg due to the type of compression.

Human Skulls - September 22, 2010, 9:19 pm

Dude this is dank. +rep on this tut

MrZander - September 22, 2010, 10:14 pm

Thanks guys!

I'm working on some more projects, I hope to post them soon.

Let's hope this section starts getting active!

xms - September 23, 2010, 1:24 pm

Looks interesting...but how its going on? i need another exe to decrypt and execute this hidden source?

MrZander - September 23, 2010, 3:47 pm

(Sep 23, 2010 01:24 PM)xms Wrote: [\[CITE\]](#) Looks interesting...but how its going on? i need another exe to decrypt and execute this hidden source?

Yes, but it is much easier to bypass an AV with this.

Ablaze - September 30, 2010, 8:42 pm

So brainstorming. . .

Let's say we have an archive of two files. We could name the first file keygen.exe and the second logo.png,jpg w/e.

We bind an extractor.exe to ... a keygen, so that when the keygen runs it first reads the "logo" image file, and executes the examined bytes. Then after it has loaded the keylogger, rat etc etc. . . it could load the keygen so it doesn't look suspicious?

Because either way we need a .exe on the other computer that will decode the image, right?

So isn't this kind of pointless (as of right now), because we are going to need a .exe in the picture somewhere. And if someone had a FUD keylogger, server anyway then they could just bind THAT to the keygen instead of an extractor to read the secret code in the image?

MrZander - September 30, 2010, 8:52 pm

(Sep 30, 2010 08:42 PM)Ablaze Wrote: [\[CITE\]](#) So brainstorming. . .

Let's say we have an archive of two files. We could name the first file keygen.exe and the second logo.png,jpg w/e.

We bind an extractor.exe to ... a keygen, so that when the keygen runs it first reads the "logo" image file, and executes the examined bytes. Then after it has loaded the keylogger, rat etc etc. . . it could load the keygen so it doesn't look suspicious?

Because either way we need a .exe on the other computer that will decode the image, right?

So isn't this kind of pointless (as of right now), because we are going to need a .exe in the picture somewhere. And if someone had a FUD keylogger, server anyway then they could just bind THAT to the keygen instead of an extractor to read the secret code in the image?

It's just a POC

You can use it however you want to.

Ablaze - September 30, 2010, 9:02 pm

Oh no its a great concept, I was just wondering if you had any specific ideas that would be more lethal than like what I listed.

Good job man.

soul_ninja - October 29, 2010, 2:19 pm

lol at Integer.toString()

Arnow - October 29, 2010, 2:29 pm

What's the point of this since when we executes the image it doesn't open the executable..

DarkShadowz - October 30, 2010, 2:39 am

Whats the point of this? Making a crypter of course xD.

It is a great exe data storage method to bypass AV's.

MrZander - October 30, 2010, 12:27 pm

(Oct 29, 2010 02:19 PM)soul_ninja Wrote: [\[CITE\]](#) lol at Integer.toString()

I learned VB.NET first lol, it's a habbit.

MirrorDude - November 1, 2010, 4:48 pm

So.. when i open it, it install the EXE (my server) into my computer...?

MrZander - November 1, 2010, 10:26 pm

(Nov 1, 2010 04:48 PM)MirrorDude Wrote: [\[CITE\]](#) So.. when i open it, it install the EXE (my server) into my computer...?

It's not a complete program, just a POC.

You will have to modify it to your needs, which in your case would mean making it retrieve the exe from the image and then executing it.

SuburbanInsanity - December 18, 2010, 10:14 pm

error in console when I run..?

```
Code Exception in thread "main" java.lang.NoClassDefFoundError: base/class
Caused by: java.lang.ClassNotFoundException: base.class
  at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
  at java.security.AccessController.doPrivileged(Native Method)
  at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:248)
Could not find the main class: base.class. Program will exit.
```

Ben121 - December 19, 2010, 3:54 am

(Dec 18, 2010 11:14 PM)demoniax Wrote: [\[CITE\]](#) error in console when I run..?

```
Code Exception in thread "main" java.lang.NoClassDefFoundError: base/class
Caused by: java.lang.ClassNotFoundException: base.class
  at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
  at java.security.AccessController.doPrivileged(Native Method)
  at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:248)
Could not find the main class: base.class. Program will exit.
Try compiling it before you run it.
```

SuburbanInsanity - December 19, 2010, 10:43 am

I did, I am not a noob. I compiled it and I runned base.class. I just downloaded the file all made. I don't know if and where I need to change it...

MrZander - December 19, 2010, 2:56 pm

(Dec 19, 2010 11:43 AM)demoniax Wrote: [\[CITE\]](#) I did, I am not a noob. I compiled it and I runned base.class. I just downloaded the file all made. I don't know if and where I need to change it...

If you are not a noob, then you should be able to solve this issue yourself.

_s1gma - December 19, 2010, 4:56 pm

shit stenography is shit

try using the last bit (or two) of every byte to store data (two is best, not noticeable and 4 pic bytes = 1 data byte), that way it will actually keep a lot of the image intact and a message will be so much more obscure.

for even more points check for an empty alpha channel (png format) and stick it all in there.

SuburbanInsanity - December 19, 2010, 5:47 pm

(Dec 19, 2010 03:56 PM)MrZander Wrote: [\[CITE\]](#) If you are not a noob, then you should be able to solve this issue yourself.

I fixed it. Thanks.

Spoiler ([Click to View](#))Message too short.

Nmanpure Pkr - December 22, 2010, 5:10 pm

Nice tut and thanks for the image, cant wait to see what it is decompiled. Lol and this is actully a great way to compress files.

dipidolor - February 15, 2011, 4:09 am

Yea the Idea is nice, but there is a bug with the size. Trying on textfiles, I've noticed that it appends some empty bytes on EOF. The bigger the File, the more 0's.

-Darkness- - February 15, 2011, 12:25 pm

Or we can put rar files in jpg's with exe's in them. ;)

prbindia - March 25, 2011, 3:53 am

Amezing Tut dude if i can i had reped u :-)

Zyx - March 25, 2011, 6:43 pm

Looks awesome. I haven't seen any similar tutorial in Java, since i haven't been here for so long. But i'll definatly try this out tomorrow :)

Thanks :)

dynfo - July 3, 2011, 10:30 am

It's not working.

Spoiler ([Click to View](#))msgtoshort
msgtoshort

17R0LL - July 3, 2011, 11:27 am

Looks really interesting, bookmarked and will review later. Thanks.

BronzeByte - March 11, 2012, 1:03 pm

Your Java markup/style is horrible... Please use the common standards -.-

Also this is pretty useless as Java can't invoke Exe's in most cases -.-

MrZander - March 12, 2012, 12:42 am

(Mar 11, 2012 01:03 PM)BronzeByte Wrote: [\[CITE\]](#) Your Java markup/style is horrible... Please use the common standards -.-

Also this is pretty useless as Java can't invoke Exe's in most cases -.-

Ok, I understand the style issue.

But what the fuck are you talking about.

1. This can clearly be adapted to store ANY filetype.
2. Java CAN invoke an .exe in most cases as most people have Windows.

If you can't see that, you are retarded.

Mohatu - March 12, 2012, 5:14 am

I've been wanting to post on this thread for a while but gravedigging is not my thing. I left the gravedigger above a nice red apple for his incompetence. Great tutorial, I haven't seen anything like it before.

Morchid - March 12, 2012, 10:58 am

this is a really good one, nice job !!!! :)

holysheetman - April 1, 2012, 3:26 pm

woohoo! not bad at all...very good =)

Morchid - April 2, 2012, 9:48 am

is it like if I open the image file(.jpg,etc) will I open the hidden EXE?

MrZander - April 2, 2012, 12:01 pm

(Apr 2, 2012 09:48 AM)Adele V. Thomas Wrote: [\[CITE\]](#) is it like if I open the image file(.jpg,etc) will I open the hidden EXE?

No, you still need to have a program to decrypt the image.

novalyze - April 2, 2012, 12:51 pm

`pixel[i >> 2] = (b[i++] << 24) | (b[i++] & 0xff) << 16) | (b[i++] & 0xff) << 8) | (b[i++] & 0xff);`

lord_nathan9 - June 29, 2012, 12:54 pm

Seems very useful for transferring malicious code for later extraction. I like it.

d3admouse - June 30, 2012, 2:58 pm

Lol it looks nice, gonna try it. What it does it exactly?

Jire - July 1, 2012, 7:15 pm

(Apr 2, 2012 12:51 PM)novalyze Wrote: [\[CITE\]](#) `pixel[i >> 2] = (b[i++] << 24) | (b[i++] & 0xff) << 16) | (b[i++] & 0xff) << 8) | (b[i++] & 0xff);`Exactly man! This method is very wasting, and can be compressed further. Because it's RGB+A, 4 bytes / pixel, store 4 bytes per pixel and leave PNG to the compression :-)

Able to reduce file size to as low as 15% of original size with this method, epic.

joussyd - October 30, 2012, 11:31 pm

what compiler did you use? i mean how run it? newbie asking

mrdare360 - October 31, 2012, 12:14 am

Nice job, really clean and informative tutorial! Thanks for this :)

matt - June 22, 2013, 7:04 pm

This would be amazing for a java driveby! Couldn't you make it so it downloads it as a jpg and then changes it back to an exe and excecutes it? I'm working on it now! :D

Victoria J - June 22, 2013, 8:01 pm

(Jun 22, 2013 07:04 PM)Midnight Blue™ Wrote: [\[CITE\]](#) This would be amazing for a java driveby! Couldn't you make it so it downloads it as a jpg and then changes it back to an exe and excecutes it? I'm working on it now! :D

Big gravedig =| meh...

You could do what you said, but may as well just download it and run the exe, not really any

difference unless the picture is used as trickery.

JavaProphet - June 23, 2013, 9:37 am

A more compressed version would store 3 or 4(RGB/RGBA) bytes of data in each pixel, to maximize data storage.

EDIT: Out of curiosity, I decided to make a quick file->PNG and vice versa converter, source:

```
Code import java.awt.image.BufferedImage;
```

```
import java.io.File;
```

```
import java.io.IOException;
```

```
import java.nio.ByteBuffer;
```

```
import java.nio.file.Files;
```

```
import java.nio.file.StandardOpenOption;
```

```
import javax.imageio.ImageIO;
```

```
public class IMGEXE {
    public static void main(String[] args) {
        boolean toimg = true;
        File from = null;
        if (args.length >= 1) {
            from = new File(args[0]);
            if (!from.exists() || !from.isFile()) {
                System.out.println((from != null ? from.toString() : "") + " does not exist, or is not a file.");
                return;
            }
        }
        File to = null;
        if (args.length >= 2) {
            to = new File(args[1]);
        }
        toimg = !(from.getName().endsWith(".png"));
        try {
            to.mkdirs();
            to.delete();
            to.createNewFile();
        } catch (IOException e) {
            System.out.println("Could not write: " + to.toString());
            return;
        }
        if (toimg) {
            byte[] frm = new byte[] {};
            try {
                byte[] tmp = Files.readAllBytes(from.toPath());
                frm = new byte[tmp.length + 4];
                ByteBuffer bb = ByteBuffer.allocate(4);
                bb.putInt(tmp.length);
                byte[] tmp2 = new byte[4];
                bb.position(0);
                bb.get(tmp2);
                System.arraycopy(tmp2, 0, frm, 0, 4);
                System.arraycopy(tmp, 0, frm, 4, tmp.length);
            } catch (IOException e) {
                System.out.println("Could not read: " + from.toString());
            }
        }
    }
}
```

```

        return;
    }
    int sqr = (int) Math.sqrt(frm.length / 4) + 1; // 4 bytes per pixel @ sqrt so can be squared for
number of pixels inside image.
    BufferedImage img = new BufferedImage(sqr, sqr, BufferedImage.TYPE_4BYTE_ABGR);
    for (int x = 0; x < img.getWidth(); x++) {
        for (int y = 0; y < img.getHeight(); y++) {
            int index = (y * img.getWidth() + x) * 4;
            ByteBuffer bb = ByteBuffer.allocate(4);
            for (int p = 0; p < 4; p++) {
                if (index + p >= frm.length) {
                    bb.put(p, (byte) 0x0);
                } else {
                    bb.put(frm[index + p]);
                }
            }
            img.setRGB(x, y, bb.getInt(0));
        }
    }
    try {
        ImageIO.write(img, to.getName().substring(to.getName().lastIndexOf(".") + 1), to);
    } catch (IOException e) {
        System.out.println("Could not write: " + to.toString());
        return;
    }
} else {
    BufferedImage img = null;
    try {
        img = ImageIO.read(from);
    } catch (IOException e) {
        System.out.println("Could not read: " + from.toString());
        return;
    }
    if (img != null) {
        ByteBuffer bb2 = ByteBuffer.allocate(4);
        bb2.putInt(0, img.getRGB(0, 0));
        int size = bb2.getInt(0);
        byte[] fil = new byte[size];
        for (int x = 0; x < img.getWidth(); x++) {
            for (int y = 0; y < img.getHeight(); y++) {
                int index = (y * img.getWidth() + x) * 4;
                if (index == 0) continue;
                ByteBuffer bb = ByteBuffer.allocate(4);
                bb.putInt(img.getRGB(x, y));
                for (int p = 0; p < 4; p++) {
                    if (index + p - 4 < fil.length) {
                        fil[index + p - 4] = bb.get(p);
                    }
                }
            }
        }
    }
    try {
        Files.write(to.toPath(), fil, StandardOpenOption.WRITE);
    }
}

```

```
        }catch (IOException e) {  
System.out.println("Could not write: " + from.toString());  
return;  
        }  
    }  
}  
System.out.println("Done!");  
}  
}
```

I especially like it since it uses the PNG compression algorithms on an EXE, reducing an 11.6mb file to a mere ~5.6 mb. This uses all 4 channels of RGBA, so no space is wasted. It also make some interesting images, similar to OP: This is of Age of Empires 3.