EXHIBIT 6

An analysis of Dorkbot's infection vectors (part 2)

msft-mmpc | 21 Nov 2012 3:35 AM | 0

In part 1 of this series, we talked about Dorkbot and its spreading mechanisms that required user interaction. In this post, we'll talk about how Dorkbot spreads automatically, via drive-by downloads and Autorun files.

Spreading vectors not requiring user interaction: Drive-by downloads and Autorun files

Dorkbot can also spread automatically, without user interaction. We recently encountered a malicious Java applet that exploits the vulnerability described in CVE-2012-4681 to distribute the Dorkbot worm. We detect the applet as Exploit:Java/CVE-2012-4681.HD. Let's take a closer look at how this exploit works.

Java applets that are not digitally signed are considered untrusted. They are executed with limited permissions by the Java Runtime Environment. Before it can download and execute arbitrary files, Exploit:Java/CVE-2012-4681.HD has to disable the security manager, which defines the security policy of the applet. The security manager can be disabled with a call to *System.setSecurityManager (null)*, but applets are restricted from calling this method directly.

The exploit relies on vulnerabilities in the implementation of the following two methods:

- Method *com.sun.beans.finder.ClassFinder.findClass(String,ClassLoader)*
- Method *com.sun.beans.finder.MethodFinder.findAccessibleMethod (Class,String,Class[])*

We decompiled the method ClassFinder.findClass to determine why it was vulnerable. As shown in Figure 8, *ClassFinder.findClass* calls the method *Class.forName* in its internal implementation. The method *Class.forName* in turn only looks at the immediate caller to perform security checks. As you can see, the vulnerability lies in the way *Class.forName* is used, and not in the method *Class.forName* itself.

The fix was to perform an additional package access check at the beginning of method *ClassFinder.findClass*, a check that fails if an applet attempts to access a restricted Java class (Figure 8).

```
package com.sun.beans.finder;

public final class ClassFinder
{
    public static Class findClass(String name) throws ClassNotFoundException
    {
        ReflectUtil.checkPackageAccess(name);
        ...
        return Class.forName(name);
    }
}

package java.lang;

public final class Class implements Serializable, GenericDeclaration, Type, AnnotatedElement
{
    ...
    public static Class forName(String name) throws ClassNotFoundException
    {
        return forName0(name, true, ClassLoader.getCallerClassLoader());
    }
}
```

***Figure 8**: The vulnerability in com.sun.beans.finder.ClassFinder.findClass (String,ClassLoader)*

Another issue, this time in the implementation of the method *sun.awt.SunToolkit.getField(Class,String)*, allows one to access private members of Java classes. The method *SunToolkit.getField* would not be accessible by default to user code, but the exploit calls it with the help of a *java.beans.Expression* object. *java.beans.Expression.execute()* is also vulnerable because it relies on the two vulnerable methods described above.

Exploit:Java/CVE-2012-4681.HD calls *SunToolkit.getField* to modify a private member of a *java.beans.Statement* object and set the access control context to

"all permissions". The class Statement can be used to invoke methods from arbitrary classes with modified access control context value. The exploit relies on a Statement object with modified access control context to invoke the privileged method *System.setSecurityManager*. After this, it has the permission to download additional malware (Figure 9).

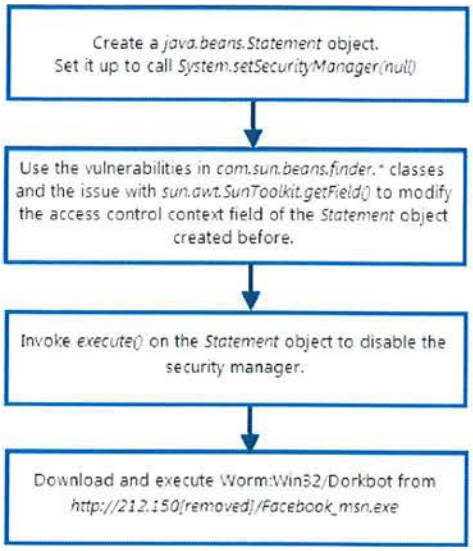

***Figure 9**: Execution flow of Exploit:Java/CVE-2012-4681.HD*

As is typical for Java exploits nowadays, the code of Exploit:Java/CVE-2012-4681.HD is heavily obfuscated to try to bypass AV detection. Figure 10 shows how the exploit retrieves the private field "acc" of the *java.beans.Statement* class, a field that defines the access control context.

***Figure 10**: Obfuscated code in Exploit:Java/CVE-2012-4681.HD*

Exploits for CVE-2012-4681 are guaranteed to work if the Java Runtime Environment is vulnerable (unlike exploits for memory corruptions, for instance). They are also platform independent (so they can also infect *nix and Mac users) and target a huge base of Java installations.

Unsurprisingly, as shown in Figure 11, our telemetry indicates that exploits for CVE-2012-4681 have been widely used to distribute malware since the vulnerability was first made public in late August 2012. A security update to resolve it was released around the same time.

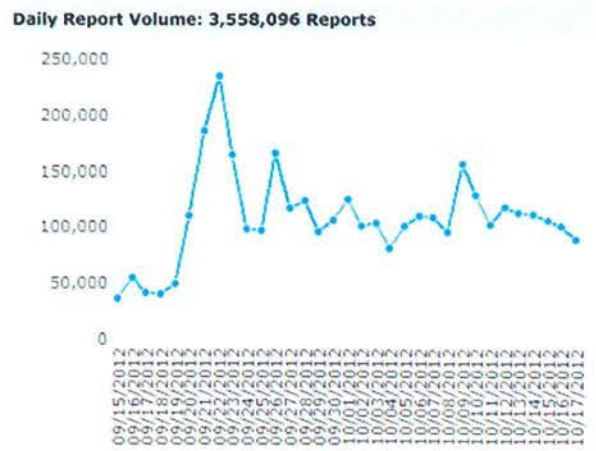

***Figure 11**: Infections attempts with CVE-2012-4681 Java exploits reported from September 15th to October 17th, 2012*

To avoid getting infected through drive-by downloads, make sure your software is up to date – for Java specifically, we talked about that in a previous post.

Worm:Win32/Dorkbot can also infect removable drives, by creating an autorun.inf file that points to a copy of the worm. If you have Autorun enabled in your computer, Dorkbot automatically runs whenever the removable drive is accessed. Fortunately, this distribution method is not very effective anymore as explained in a previous blog post. Please keep your Windows up-to-date to deal with this infection vector.

Conclusion

As we previously mentioned, malware these days use a variety of ways to infect computers and Dorkbot is no exception. And its access to a C&C server allows for a certain level of dynamic behavior. Because of this, we advise users to be more vigilant against all the different channels that Dorkbot uses.

And finally, always make sure your definitions are up-to-date for your antivirus solution. If you don't have one and you're running Windows XP, Vista, or 7, you can download and install Microsoft Security Essentials for free. If you're using Windows 8, make sure your antivirus program is enabled and running properly.

The following are the SHA1s of the samples that we've analyzed for this blog post:

- Exploit:Java/CVE-2012-4681.HD - f624121d44b87369ba9ffa975db64fbb7bc395b3
- Worm:Win32/Dorkbot spreading component - 11a2ddb73af46060802537dec0f8799e2a0dc13f
- Worm:Win32/Dorkbot.A - 4176f4193b1ef64569bf0ab220113cce6074df4e
- Worm:Win32/Dorkbot.I - 37c09e044ebe57eb66aa6c72cb039140b3b985f1

Horea Coroiu, MMPC Munich

0

Comments