1 A8 Software and Data Integrity Failures

1.1 Serializacion

¿Qué es la serialización?

La serialización es el proceso de convertir un objeto en un formato de datos que pueda restaurarse más adelante. Las personas suelen serializar objetos para guardarlos en un almacenamiento o para enviarlos como parte de comunicaciones. La deserialización es lo contrario de ese proceso que toma datos estructurados a partir de algún formato y los reconstruye en un objeto. Hoy en día, el formato de datos más popular para serializar datos es JSON. Antes de eso, era XML.

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"usuario";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

Serialización nativa

Muchos lenguajes de programación ofrecen una capacidad nativa para serializar objetos. Estos formatos nativos suelen ofrecer más funciones que JSON o XML, incluida la personalización del proceso de serialización. Desafortunadamente, las características de estos mecanismos de deserialización nativos pueden reutilizarse para generar efectos maliciosos cuando se opera con datos que no son de confianza. Se ha descubierto que los ataques contra deserializadores permiten ataques de denegación de servicio, control de acceso y ejecución remota de código.

Lenguajes de programación afectados conocidos

- PHP
- Pitón
- Rubí
- Java
- C
- C++

Datos, no código

SÓLO los datos están serializados. El código no está serializado. La deserialización crea un nuevo objeto y copia todos los datos del flujo de bytes, para obtener un objeto idéntico al objeto que fue serializado.

Código vulnerable

El siguiente es un ejemplo bien conocido de una vulnerabilidad de deserialización de Java.

```
InputStream is = request.getInputStream();

ObjectInputStream ois = new ObjectInputStream(is);

AcmeObject acme = (AcmeObject)ois.readObject();
```

Está esperando un `AcmeObject`objeto, pero se ejecutará `readObject()`antes de que se produzca la conversión. Si un atacante encuentra la clase adecuada que implementa operaciones peligrosas `readObject()`, podría serializar ese objeto y obligar a la aplicación vulnerable a realizar esas acciones.

Clase incluida en ClassPath

Los atacantes necesitan encontrar una clase en el classpath que admita la serialización y con implementaciones peligrosas en `readObject()`.

```
package org.dummy.insecure.framework;


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.Serializable;
import java.time.LocalDateTime;


public class VulnerableTaskHolder implements Serializable {


        private static final long serialVersionUID = 1;


        private String taskName;
        private String taskAction;
        private LocalDateTime requestedExecutionTime;


        public VulnerableTaskHolder(String taskName, String taskAction) {
                super();
                this.taskName = taskName;
```

```
                this.taskAction = taskAction;
                this.requestedExecutionTime = LocalDateTime.now();
        }

        private void readObject( ObjectInputStream stream ) throws Exception
{
        //deserialize data so taskName and taskAction are available
                stream.defaultReadObject();

                //blindly run some code. #code injection
                Runtime.getRuntime().exec(taskAction);
     }
}
```

Explotar

Si la clase Java que se muestra arriba existe, los atacantes pueden serializar ese objeto y obtener la ejecución remota de código.

```
VulnerableTaskHolder go = new VulnerableTaskHolder("delete all", "rm -rf some
file");


ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(bos);
oos.writeObject(go);
oos.flush();
byte[] exploit = bos.toByteArray();
```

¿Qué es una cadena de gadgets?

Es extraño (pero podría suceder) encontrar un dispositivo que ejecute acciones peligrosas cuando se deserializa. Sin embargo, es mucho más fácil encontrar un dispositivo que ejecute acciones en otro dispositivo cuando se deserializa, y que ese segundo dispositivo ejecute más acciones en un tercer dispositivo, y así sucesivamente hasta que se desencadene una acción realmente peligrosa. Ese conjunto de gadgets que pueden usarse en un proceso de deserialización para lograr acciones peligrosas se llama "Gadget Chain".

Encontrar dispositivos para construir cadenas de dispositivos es un tema activo para los investigadores de seguridad. Este tipo de investigación suele requerir dedicar una gran cantidad de tiempo a leer el código.

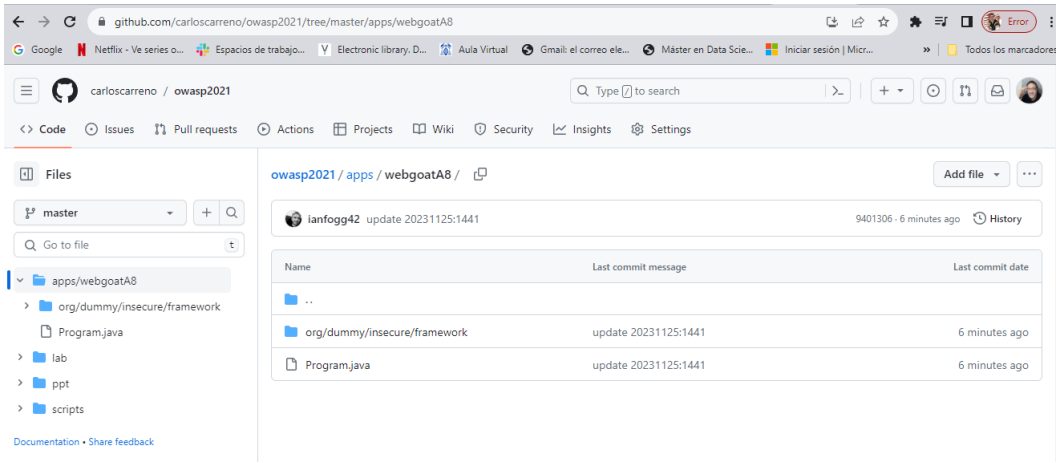
Intentemos

El siguiente cuadro de entrada recibe un objeto serializado (una cadena) y lo deserializa.

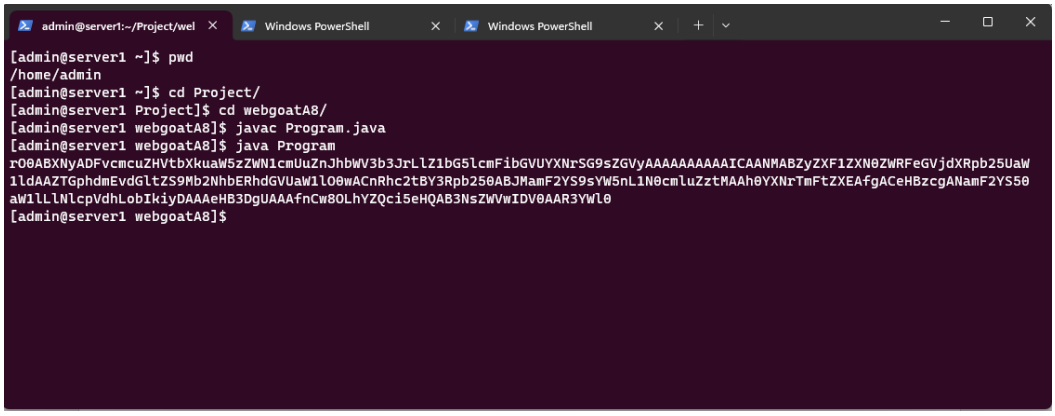
```
rO0ABXQAVklmIHlvdSBkZXNlcmlhbGl6ZSBtZSBkb3duLCBJIHNoYWxsIGJlY29tZSBtb3JlIHBvd
2VyZnVsIHRoYW4geW91IGNhbiBwb3NzaWJseSBpbWFnaW5l
```

Intente cambiar este objeto serializado para retrasar la respuesta de la página exactamente 5 segundos.

Crea la carpeta **Project/webgoatA8** y copia los archivos y directorios de la carpeta apps desde:



Ejecuta el programa y obtén el token.



El token obtenido envíalo en el formulario.

Let's try

The following input box receives a serialized object (a string) and it deserializes it.

```
rO0ABXQAVklmIHlvdSBkZXNlcmlhbGl6ZSBtZSBkb3duLCBJIHNoYWxsIGJlY29tZSBtb3JlIHBvd2VyZnVsIHRoYW4geW91IGNhbiBwb3NzaWJseSBpbWFnaW5l
```

Try to change this serialized object in order to delay the page response for exactly 5 seconds.

3NsZWVwIDV0AAR3YWI0 Submit

The task is not executable between now and the next ten minutes, so the action will be ignored. Maybe you copied an old solution? Let's try again.