```
 ________   __________  __________  __________  ________    _____   _________  _____      _____   _________  __________  __________  _____   _________  _____   _   
|        | |          ||          ||          ||        |  |     | |         ||     |    |     | |         ||          ||          ||     | |         ||     | | | |
|  _      ||     _____||  ________||     _____||    _   |  |     | |    _    ||     |    |     | |_____    ||    _     ||__      __||     | |    _    ||     |_| | |
| | |     ||    |_____|| |________||    |_____||   | |  |_ |     | |   |_|   ||     |    |     |  _____|   ||   |_|    |   |    |   |     | |   | |   ||         |
| |_|     ||     _____||________  ||     _____||   |_|     ||     | |         ||     |____ |     | |   ______||          |   |    |   |     | |   |_|   ||    _    |
|         ||    |_____| ________| ||    |_____||    __     ||     | |    _    ||          ||     | |  |______||    _     |   |    |   |     | |         ||   | |   |
|________| |__________||__________||__________||___|  |_||_____| |___| |___||__________||_____| |__________||___| |___|   |____|   |_____| |_________||_| |___|
 _________  __________  __________  __________  ___________  _____   _  _________
|         ||          ||          ||          ||           ||     | | ||         |
|    _    ||__      __||__      __||    _     ||           ||     |_| ||   ______|
|   |_|   |   |    |      |    |   |   |_|    ||           ||       _|| |________
|         |   |    |      |    |   |          ||         _||      |_  ||________  |
|    _    |   |    |      |    |   |    _     ||        |_ ||    _    | ________| |
|___| |___|   |____|      |____|   |___| |___||___________||____| |_||__________|
```

Overview

- Attacks that take advantage of the serialization and deserialization of objects
- When an object is desterilized certain actions are performed
- PHP - magic functions
- Django - internal object functions

PHP

- The vulnerability occurs when user-supplied input is not properly sanitized before being passed to the unserialize() PHP function.
- Since PHP allows object serialization, attackers could pass ad-hoc serialized strings to a vulnerable unserialize() call, resulting in an arbitrary PHP object(s) injection into the application scope.
- In order to successfully exploit a PHP Object Injection vulnerability two conditions must be met:
  - The application must have a class which implements a PHP magic method (such as __wakeup or __destruct) that can be used to carry out malicious attacks, or to start a "POP chain".
  - All of the classes used during the attack must be declared when the vulnerable unserialize() is being called, otherwise object autoloading must be supported for such classes.

PHP Example

```
class Example1
{
   public $cache_file;

   function __construct()
   {
      // some PHP code...
   }

   function __destruct()
   {
      $file = "/var/www/cache/tmp/{$this->cache_file}";
      if (file_exists($file)) @unlink($file);
   }
}

// some PHP code...

$user_data = unserialize($_GET['data']);

// some PHP code...
```

PHP Example

What if you send:

```
http://testsite.com/vuln.php?data=O:8:"Example1":1:{s:
10:"cache_file";s:15:"../../index.php";}
```

Django

- Objects can be serialized and deserialized
- These objects can have functions that are called on deserialization
- These functions can execute arbitrary commands
- To mitigate this the framework encrypts the serialized object but if an attacker can get the password then all is lost

Django Example

- https://fail0verflow.com/blog/2014/plaidctf2014-web200-reeekeeeeee.html

Mitigations

- Never trust user input
- Sanitize objects before deserialization
- Encrypt objects that you are serializing and before sending to the users
- Don’t loose you secret encryption key :)

Resources

- https://www.owasp.org/index.php/PHP_Object_Injection
- http://heine.familiedeelstra.com/security/unserialize