Exploiting Deserialization Vulnerabilities in PHP Applications

OWASP
The Open Web Application Security Project

Andrew Kramer

OWASP
The Open Web Application Security Project

- Andrew Kramer
  - andrew@jmpesp.org
- M.S. and B.S. from DSU
- Currently Comp-Sci Faculty at DSU
- Worked at a National Lab
- Worked in Pentesting
- Hacker hobbyist :)

OWASP
The Open Web Application Security Project

- PHP hacking in the good ol' days!
  - SQL injection
  - File inclusion
  - Command injection

OWASP
The Open Web Application Security Project

- This has gotten harder :(
  - SQL prepared statements
  - PHP patched the null-byte (`%00`) injection trick
  - `allow_url_include` off by default
  - Web application firewalls
  - Coding practices are generally improving

OWASP
The Open Web Application Security Project

- More exotic PHP vulnerabilities
  - Type juggling issues
  - `preg_replace()` with “`e`” modifier
  - **Deserialization of user supplied data**

OWASP
The Open Web Application Security Project

- FYI: This is NOT unique to PHP
  - PHP: `serialize()` / `unserialize()`
  - Python: `pickle.dump()` / `pickle.load()`
  - Ruby: `Marshal.dump()` / `Marshal.load()`
  - Java: `Serializable`
  - Etc...

- (de)serialization: The process of converting a complex object or data structure to/from a plain text stream

```php
class Person {

    private $name;
    private $age;

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

}

$me = new Person("Andrew", 27);
echo serialize($me);
```

```
O:6:"Person":2:{s:12:"Personname";s:6:"Andrew";s:11:"Personage";i:27;}
```

OWASP
The Open Web Application Security Project

- Commonly used for...
  - Storing objects in a database or textfile
  - Storing complex things in session data
  - Passing objects between systems
  - Etc

- The problem with unserialize:
  - `unserialize()` can instantiate objects
  - Objects can “do things”, i.e. execute code.

- What if we control the input?

- Problem 1: We can't define functions in the serialized object, only data. :(

```
class Person {

    private $name;
    private $age;

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

}

$me = new Person("Andrew", 27);
echo serialize($me);
```

```
O:6:"Person":2:{s:12:"Personname";s:6:"Andrew";s:11:"Personage";i:27;}
```

OWASP
The Open Web Application Security Project

- Solution: Lots of classes (probably) already exist. They have functions!
  - Maybe there's one that writes to a file?
  - Or executes a SQL statement?
  - Or executes code?

```
-rw-r--r-- 1 root root 1542 Apr 10 20:43 dbconnection.class.php
-rw-r--r-- 1 root root 2334 Apr 10 21:03 index.php
-rw-r--r-- 1 root root  906 Apr 10 20:39 messageapp.class.php
-rw-r--r-- 1 root root  338 Nov 21  2015 user.class.php
```

OWASP
The Open Web Application Security Project

- Problem 2: If we find one, how do we get the function(s) to execute?
  - Unserializing sets the "`$cmd`"
  - But DOESN'T actually call `OhNo()`

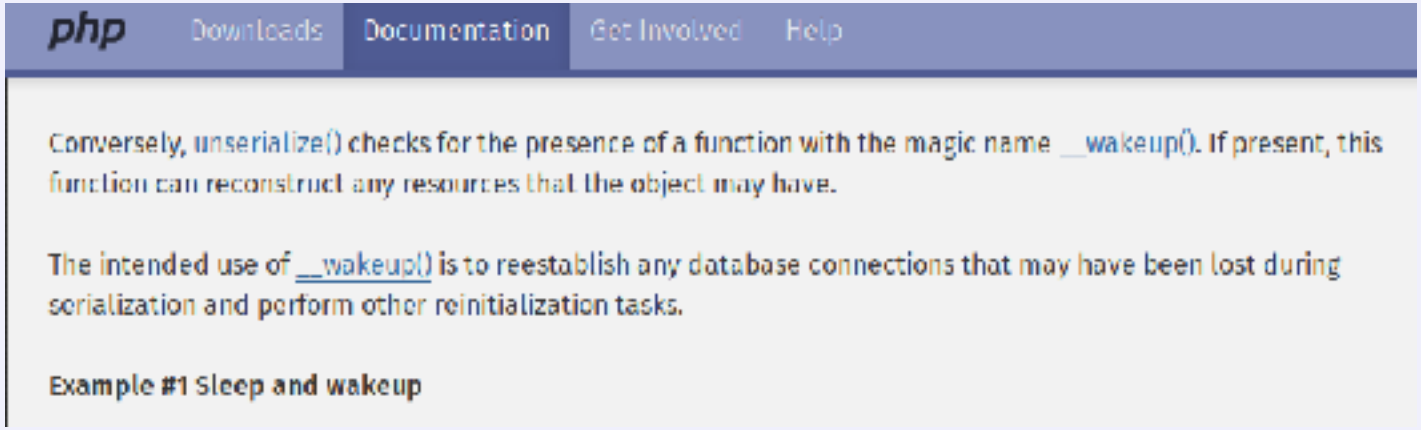
```
class Yikes {

    private $cmd;

    public function OhNo() {
        system($this->cmd);
    }

}
```

- Solution: PHP implements several “magic methods” that are automatically called.
  - `__wakeup()`, `__destruct()`, etc...
  - Do any of *those* lead to interesting code?
  -



https://secure.php.net/manual/en/language.oop5.magic.php

OWASP
The Open Web Application Security Project

- Realistically, probably like...

```
class Application {

    private $log_file;
    private $name;

    // ...

    public function __destruct() {
        file_put_contents(
            $this->log_file,
            "Shutdown: " . $this->name
        );
    }

}
```

- Case study:
  - Invision Power Board
  - Version 3.3.4
  - User data passed to `unserialize()` via the cookie value "`member_id`". Exploitable using the "`db_driver_mysql`" class.
  - https://www.exploit-db.com/exploits/22398/

OWASP
The Open Web Application Security Project

- The vulnerability…
  - /admin/sources/base/core.php
  - IPSCookie::get()

```
static public function get($name)
{

    // ...

    $_value = $_COOKIE[ ipsRegistry::$settings['cookie_id'].$name ];

    if ( substr( $_value, 0, 2 ) == 'a:' )
    {
        return unserialize( stripslashes( urldecode( $_value ) ) );
```

- The abusable class:
  - *(Dramatization)*

```
class db_driver_mysql {

    private $use_debug_log;
    private $debug_log;

    public function __destruct() {

        if($this->use_debug_log) {
            $log = fopen($this->debug_log, "a");
            fwrite($log, $_SERVER["QUERY_STRING"]);
            fclose($log);
        }

    }
```

OWASP
The Open Web Application Security Project

- Demo time!
  - DakotaCon 2017 CTF - “ClassyChat”
  - Live site: http://classychat.hostbin.org/
  - Code: https://jmpesp.org/public/classychat.zip

OWASP
The Open Web Application Security Project

- Questions?
- Comments?

- Contact: andrew@jmpesp.org