Phar Deserialization leads to Remote Code Execution

High **sreichel** published **GHSA-fg79-cr9c-7369** last week

Package

php **openmage/magento-lts** (Composer)

Affected versions

<= 20.16.0

Patched versions

20.17.0

Description

Source: https://hackerone.com/reports/3482926
Original GHSA: https://github.com/OpenMage/magento-lts/security/advisories/GHSA-9qv7-qm9f-2xfh

---

PHP functions such as `getimagesize()`, `file_exists()`, and `is_readable()` can trigger deserialization when processing `phar://` stream wrapper paths. OpenMage LTS uses these functions with potentially controllable file paths during image validation and media handling. An attacker who can upload a malicious phar file (disguised as an image) and trigger one of these functions with a `phar://` path can achieve arbitrary code execution.

Severity

High - CVSS 3.1 Score: 8.1

CVSS Vector

```
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
```

Metric	Value	Justification
Attack Vector (AV)	Network	Exploitable via file upload and web requests

Metric	Value	Justification
Attack Complexity (AC)	High	Requires file upload + triggering phar:// access
Privileges Required (PR)	None	Some upload vectors don't require authentication
User Interaction (UI)	None	Exploitation is automatic once triggered
Scope (S)	Unchanged	Impacts the vulnerable component
Confidentiality (C)	High	Full system access via RCE
Integrity (I)	High	Arbitrary code execution
Availability (A)	High	Complete system compromise possible

Affected Products

- OpenMage LTS versions < 20.16.1
- All versions derived from Magento 1.x with these code paths

Affected Files

File	Line	Vulnerable Functio
`app/code/core/Mage/Core/Model/File/Validator/Image.php`	72	`getimagesize($filePa`
`app/code/core/Mage/Cms/Model/Wysiwyg/Images/Storage.php`	137	`getimagesize($item->getFilename())`
`lib/Varien/Image.php`	71	`$this->_getAdapter()->open($this->_fileNar`

Vulnerability Details

PHP's phar (PHP Archive) format stores metadata that is serialized. When PHP's stream wrapper functions access a file using the `phar://` protocol, the metadata is automatically deserialized. This occurs even with seemingly safe functions like `file_exists()` or `getimagesize()`.

A polyglot file can be crafted that is both a valid image (passing initial validation) and a valid phar archive containing malicious serialized objects. When the application later processes this file using `phar://`, the deserialization triggers a gadget chain leading to RCE.

Attack Flow

1. **Create polyglot file**: Attacker creates a file that is both valid JPEG and valid PHAR
2. **Upload file**: Attacker uploads the polyglot via product images, CMS media, or import

3. **Trigger phar:// access**: Attacker causes the application to access the file using `phar://` wrapper
4. **Code execution**: PHAR metadata deserialization triggers gadget chain

Proof of Concept

```php
<?php
// Create malicious phar file
class ExploitGadget {
    public $cmd = 'id > /tmp/pwned';
    function __destruct() {
        system($this->cmd);
    }
}

$phar = new Phar('exploit.phar');
$phar->startBuffering();
$phar->addFromString('test.txt', 'test');
$phar->setStub('<?php __HALT_COMPILER(); ?>');
$phar->setMetadata(new ExploitGadget());
$phar->stopBuffering();

// Rename to appear as image
rename('exploit.phar', 'exploit.jpg');

// When getimagesize('phar://path/to/exploit.jpg') is called,
// the ExploitGadget::__destruct() method executes
```

Remediation

v20.16.1 (Patch Release)

Block `phar://` paths before passing to vulnerable functions:

```php
// Before (vulnerable)
[$imageWidth, $imageHeight, $fileType] = getimagesize($filePath);

// After (fixed)
if (str_starts_with($filePath, 'phar://')) {
    throw new Exception('Invalid image path.');
}
[$imageWidth, $imageHeight, $fileType] = getimagesize($filePath);
```

Additionally, ICO files (which cannot be re-encoded by GD) are now scanned for phar signatures:

- `__HALT_COMPILER();` - Required phar stub
- `<?php` - PHP opening tag
- `<?=` - PHP short echo tag

v20.17.0 (Minor Release)

Additional hardening measures:

1. **ICO uploads removed**: ICO file support is completely removed from new image uploads. This eliminates the polyglot attack vector entirely since all other image formats are re-encoded by GD, which strips any embedded phar metadata.
2. **Phar wrapper disabled**: The `phar://` stream wrapper is unregistered at application bootstrap, preventing any phar deserialization attacks regardless of code path.
3. **Cache deserialization hardening**: All `unserialize()` calls on cached data now use `allowed_classes => false` as defense-in-depth.

Note: Existing uploaded ICO files will continue to work. Only new ICO uploads will be rejected. Users are encouraged to use PNG favicons for new uploads.

Workarounds

If immediate upgrade is not possible:

1. **Disable phar stream wrapper** (if not needed):

   ```
   ; php.ini
   disable_functions = phar://
   ```

   Or in code:

   ```
   stream_wrapper_unregister('phar');
   ```

2. **Strict upload validation**: Implement additional validation beyond file extension
3. **File storage isolation**: Store uploads outside web root with randomized names
4. **Web Application Firewall**: Block requests containing `phar://` in parameters

References

- CWE-502: Deserialization of Untrusted Data
- PHP Phar Deserialization
- OWASP: Unrestricted File Upload

Credit

This vulnerability was discovered and responsibly disclosed by blackhat2013 through HackerOne.

Timeline

- **2025-12-31**: Vulnerability reported via HackerOne
- **2026-01-21**: Fix developed and tested
- **2026-XX-XX**: Security release v20.16.1 published (phar:// blocking + ICO signature scanning)
- **2026-XX-XX**: Release v20.17.0 published (ICO support removed entirely)
- **2026-XX-XX**: Public disclosure

Severity

High 8.1 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

CVE ID

CVE-2026-25524

Weaknesses

▶ CWE-502