Back to overview

SEPTEMBER 6, 2024

THREAT INTELLIGENCE

CVE-2024-23119: CRITICAL SQL INJECTION VULNERABILITY IN CENTREON

by Security News

Overview

The SonicWall Capture Labs threat research team became aware of the threat CVE-2024-23119, assessed its impact and developed mitigation measures for this vulnerability.

CVE-2024-23119 is a high-severity SQL Injection vulnerability in Centreon, impacting Centreon Web versions prior to 22.10.17, 23.04.13, and 23.10.5. Centreon is a widely used network, system and application monitoring tool. This issue resides within the insertGraphTemplate function, which fails to properly validate user inputs before incorporating them into SQL queries. As a result, authenticated attackers can execute arbitrary SQL commands, potentially gaining control over the database and executing code within the context of the service account. The vulnerability is categorized with a CVSS base score of 8.8, reflecting a high risk due to its potential impact on confidentiality, integrity and availability. The exploit prediction scoring system (EPSS) estimates a 0.07% chance of exploitation in the next 30 days, indicating it is less likely but still notable. This vulnerability was initially reported by Zero Day Initiative (ZDI) as ZDI-CAN-22339 and has been addressed in Centreon Web versions 22.10.15, 23.04.10, and 23.10.1. For further details and mitigation, refer to the advisory provided by ZDI and the Centreon GitHub repository.

Technical Overview

CVE-2024-23119 stems from an SQL Injection issue occurring during the creation of graph templates. Centreon utilizes a web interface that communicates over HTTP/HTTPS. The vulnerability is identified in the formGraphTemplate.php and insertGraphTemplateInDB() functions, which are executed through main.get.php (see Figure 1). The core issue arises from inadequate validation and sanitization of specific request parameters that are used to construct SQL queries.

```
// Process parameters and include relevant scripts
switch ($o) {
    case "a":
    case "c":
    case "w":
        require_once $path . "formGraphTemplate.php";
        break;
    default:
        require_once $path . "listGraphTemplates.php";
        break;
}
```

Figure 1: Code Snippet from 'main.get.php'

When a graph template is created, an HTTP POST request is sent to main.get.php with parameters including p, o and submitA. The formGraphTemplate.php script processes these parameters and invokes the insertGraphTemplateInDB() function (see Figure 2), which then calls insertGraphTemplate().

```
if ($form->validate()) {
    if ($form->getSubmitValue("submitA")) {
        $graphObj->setValue(insertGraphTemplateInDB());
    }
    $form->freeze();
}
```

Figure 2: Code Snippet from 'formGraphTemplate.php'

In this function, an SQL query is constructed to insert data into the giv_graphs_template table. While some parameters are sanitized, others like lower_limit, upper_limit, size_to_max, default_tpl1, and scaled are directly incorporated into the SQL query without proper sanitization (see Figure 3). This lack of sanitization permits attackers to inject arbitrary SQL commands.

```
$rq = "INSERT INTO `giv_graphs_template` (lower_limit, upper_limit, size_to_max, default_tpl1, scaled)
VALUES (";
$rq .= isset($ret["lower_limit"]) ? "'" . $ret["lower_limit"] . "', " : "NULL, ";
$rq .= isset($ret["upper_limit"]) ? "'" . $ret["upper_limit"] . "', " : "NULL, ";
$rq .= isset($ret["size_to_max"]) ? "'" . $ret["size_to_max"] . "', " : "0, ";
$rq .= isset($ret["default_tpl1"]) ? "'" . $ret["default_tpl1"] . "', " : "NULL, ";
$rq .= isset($ret["scaled"]) ? "'" . $ret["scaled"] . "'" : "'0'";
```

Figure 3: Code snippet from 'insertgraphTemplate() in DB-Func.php'

Exploiting this SQL Injection vulnerability allows a remote, authenticated attacker to craft malicious requests to the server, leading to potential data leakage, data corruption or full control over the database. This vulnerability highlights the importance of rigorous input validation and the use of parameterized queries to prevent such critical security issues in web applications.

Triggering the Vulnerability

- **Send Malicious POST Requests:** An attacker can trigger the vulnerability by sending a specially crafted HTTP POST request to the Centreon web interface. This request must include malicious SQL payloads in parameters that are not properly sanitized.
- **Exploit Unsanitized Parameters:** The vulnerability arises from insufficient input validation in the lower_limit, upper_limit, size_to_max, default_tpl1, and scaled An attacker can trigger this vulnerability by injecting SQL commands into these parameters when creating or modifying graph templates.
- **Access via Graph Template Interface:** The attack must be executed through the graph template creation or modification interface, specifically by setting the request parameter p to "20404" and other relevant parameters to trigger the vulnerable code path.
- **Authenticated Access Required:** The attacker must have authenticated access to the Centreon web interface. This means the attacker needs to log in and have the appropriate permissions to create or modify graph templates to exploit this vulnerability effectively.

Exploitation

Exploiting CVE-2024-23119 involves a series of methodical steps to leverage the SQL injection vulnerability in the Centreon web management interface., The attacker must first authenticate to the Centreon API by sending a POST request to the /api/latest/authentication/providers/configurations/local endpoint. This request includes a JSON payload with valid credentials to gain access to the server's API.

```
<div id="validForm">
  <p>
    <input class="btc bt_success" name="submitA" value="Save" type="
    submit" />
       <input class="btc bt_danger" name="reset" value="
    Delete" type="reset" />
  </p>
</div>
<input name="graph_id" type="hidden" value="" />
<input name="o" type="hidden" value="a" />
<input name="centreon_token" type="hidden" value="
5f428dc7b5a2fc39450d505e991cf7e8" />

</form>
```

Figure 4: CSRF Token

Next, the attacker retrieves a CSRF token from the /main.get.php?p=20404&o=a endpoint (see Figure 4), which is necessary for making authenticated requests. The CSRF token is extracted from the HTML response using a regular expression to ensure that subsequent interactions with the server are authorized. With the token in hand, the attacker crafts a malicious payload designed to exploit the SQL injection vulnerability. This payload is injected into specific fields, such as lower_limit, upper_limit, size_to_max, default_tpl1, or scaled, in a graph template creation request. For example, a crafted payload like 1', NULL, 0, NULL, NULL, '0', NULL, NULL); CREATE TABLE poc (id int); # could be used to create a new table named poc in the database.

Figure 5: Exploitation using SQL injection

Finally, the attacker sends the malicious request to the /main.get.php?p=20404 endpoint with the payload and necessary parameters, including the CSRF token. Upon successful execution of the payload, the attacker verifies the impact by checking the database for changes, such as the presence of the newly created table. This initial access can be leveraged for further exploitation, potentially leading to more severe consequences like data breaches or unauthorized access.

SonicWall Protections

To ensure SonicWall customers are prepared for any exploitation that may occur due to this vulnerability, the following signatures have been released:

- IPS: 20295 Centreon main.get.php SQL Injection 9

Remediation Recommendations

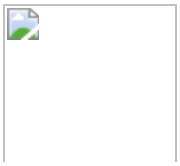
The risks posed by this vulnerability can be mitigated or eliminated by:

- Upgrade to Centreon Web versions 22.10.15, 23.04.10, or 23.10.1
- Monitor and review system logs for suspicious activity.
- Utilize up-to-date IPS signatures to filter network traffic.
- Restrict user privileges and sanitize user inputs.

Relevant Links

- ZDI-24-113 | Zero Day Initiative
- National Vulnerability Database (NVD)
- Common Vulnerability Scoring System Calculation
- CWE-89: Improper Neutralization of Special Elements used in SQL Command
- Centreon Products

SHARE THIS ARTICLE



AN ARTICLE BY

SONICWALL

Security News

The SonicWall Capture Labs Threat Research Team gathers, analyzes and vets cross-vector threat information from the SonicWall Capture Threat network, consisting of global devices and resources, including more than 1 million security sensors in nearly 200 countries and territories. The research team identifies, analyzes, and mitigates critical vulnerabilities and malware daily through in-depth research, which drives protection for all SonicWall customers. In addition to safeguarding networks globally, the research team supports the larger threat intelligence community by releasing weekly deep technical analyses of the most critical threats to small businesses, providing critical knowledge that defenders need to protect their networks.

FOLLOW US