OTF-001: Improper Input Sanitation: The path parameter of the requested URL is not sanitized before being passed to the QT frontend

High **micahflee** published **GHSA-ch22-x2v3-v6vq** on Jan 18

Package

OnionShare (OnionShare)

Affected versions	Patched versions
< 2.5	2.5

Description

Between September 26, 2021 and October 8, 2021, Radically Open Security conducted a penetration test of OnionShare 2.4, funded by the Open Technology Fund's Red Team lab. This is an issue from that penetration test.

- Vulnerability ID: OTF-001
- Vulnerability type: Improper Input Sanitization
- Threat level: Elevated

Description:

The `path` parameter of the requested URL is not sanitized before being passed to the QT frontend.

Technical description:

The `path` parameter is not sanitized before being passed to the constructor of the `QLabel`.

onionshare/desktop/src/onionshare/tab/mode/__init__.py
Lines 499 to 509 in d08d5f0

```
499         def handle_request_individual_file_started(self, event):
```

```
500             """
501             Handle REQUEST_INDVIDIDUAL_FILES_STARTED event.
502             Used in both Share and Website modes, so implemented here.
503             """
504             self.toggle_history.update_indicator(True)
505             self.history.requests_count += 1
506             self.history.update_requests()
507 
508             item = IndividualFileHistoryItem(self.common, event["data"], event["path"
```

onionshare/desktop/src/onionshare/tab/mode/history.py
Lines 456 to 483 in `d08d5f0`

```
456     class IndividualFileHistoryItem(HistoryItem):
457         """
458         Individual file history item, for share mode viewing of individual files
459         """
460 
461         def __init__(self, common, data, path):
462             super(IndividualFileHistoryItem, self).__init__()
463             self.status = HistoryItem.STATUS_STARTED
464             self.common = common
465 
466             self.id = id
467             self.path = path
```

https://doc.qt.io/qt-5/qlabel.html#details

> Warning: When passing a QString to the constructor or calling setText(), make sure to sanitize your input, as QLabel tries to guess whether it displays the text as plain text or as rich text, a subset of HTML 4 markup. You may want to call setTextFormat() explicitly, e.g. in case you expect the text to be in plain format but cannot control the text source (for instance when displaying data loaded from the Web).

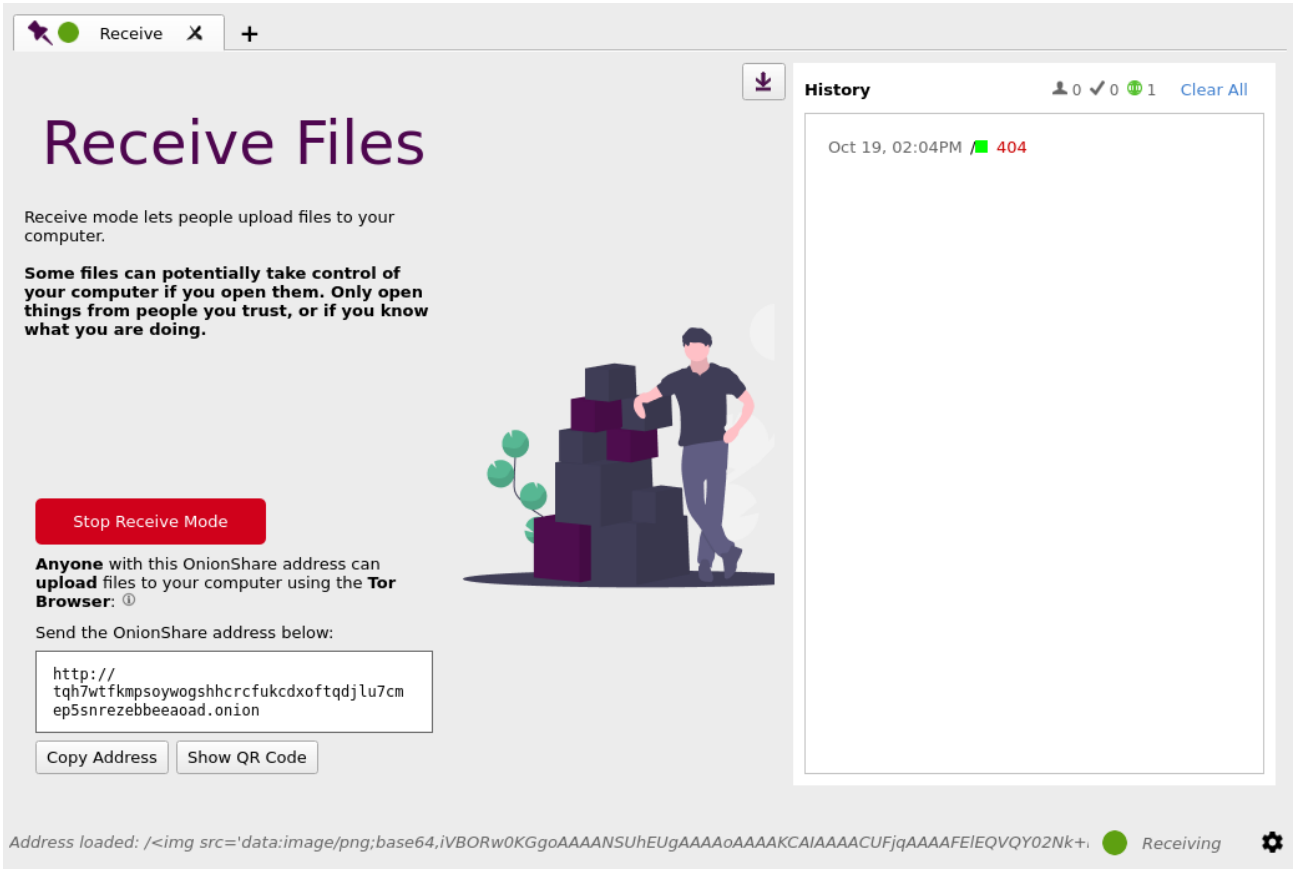
This path is used in all components for displaying the server access history. This leads to a rendered HTML4 Subset (QT RichText editor) in the Onionshare frontend.

In the following example an adversary injects a crafted image file into an Onionshare instance with receive mode and renders it in the history component of the Onionshare application.

The only requirement is another visit to the shared site with the following parameter attached to the path of the URL:

```
<img
src='
 />
```

This will be rendered as a green square in the history tab where the path value is supposed to be (the value itself is shown at the bottom of the page).



Possible scenarios where this could lead to remote code execution would be a 0-day in libpng or other internal image rendering (OTF-014 (page 12)) of the QT framework.

The QT documentation indicates that external files could be rendered, but we were unable to find a QT code path allowing for it.

Impact:

An adversary with knowledge of the Onion service address in public mode or with authentication in private mode can render arbitrary HTML (QT-HTML4 Subset) in the server desktop application. This requires the desktop application with rendered history, therefore the impact is only elevated.

Recommendation:

- Manually define the text format of the QLabel via `setTextFormat()`

Severity

High

CVE ID

CVE-2022-21690

Weaknesses

No CWEs