Allow arbitrary URLs, expect arbitrary code execution

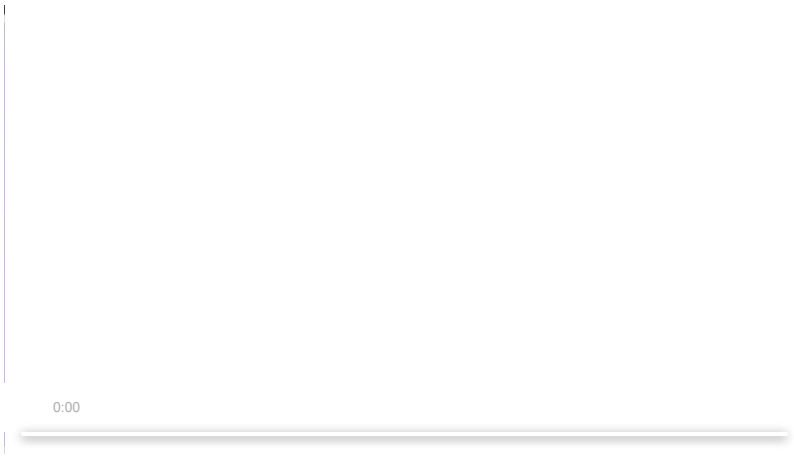
APRIL 15, 2021

BY FABIAN BRÄUNLEIN, LUKAS EULER

- We found and reported 1-click code execution vulnerabilities in popular software including **Telegram**, **Nextcloud**, **VLC**, **Libre-/OpenOffice**, **Bitcoin/Dogecoin Wallets**, **Wireshark** and **Mumble**
- Desktop applications which pass user supplied URLs to be opened by the operating system are frequently vulnerable to **code execution with user interaction**
- Code execution can be achieved either when a URL pointing to a malicious executable (`.desktop`, `.jar`, `.exe`, ...) hosted on an internet accessible file share (`nfs`, `webdav`, `smb`, ...) is opened, or an additional vulnerability in the opened application's URI handler is exploited
- Vulnerabilities following this pattern have already been found in other software, with more expected to be revealed going forward

Introduction

In this post, we show code execution vulnerabilities in numerous desktop applications, all with the same root cause: insufficient validation of user input that is later treated as a URL and opened with the help of the operating system. The required user interaction and exploitation strategy depends on the desktop environment and whether the application was hardened, for instance, with a URI-scheme allow/block list. As an example, here is what exploitation of this issue in Nextcloud (< 3.1.3) on Xubuntu 20.04 looks like:



Exploiting Nextcloud on Xubuntu

After explaining the root cause, vulnerable patterns and oddities of different OS's and desktop environments, we'll explore how this vulnerability type can be exploited in various popular desktop applications.

Table of Contents

- Introduction
- Root cause: user-supplied URLs opened by the OS
- Finding vulnerable features is straightforward

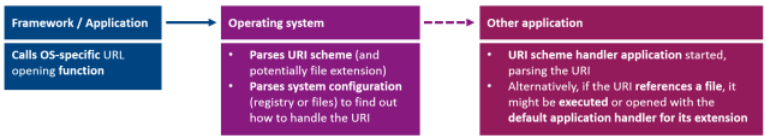
    - Mac (Catalina 10.15.6)
- Vulnerabilities
    - Nextcloud
    - Telegram
    - VLC
    - Open-/LibreOffice
    - Mumble
    - Bitcoin/Dogecoin Wallets
    - Wireshark
    - Bonus-Vulnerability: WinSCP
- Systematic mitigation requires contributions from OS, Framework, and Application maintainers
- Conclusion

Root cause: user-supplied URLs opened by the OS

A common way to open files and links from a native desktop application is by passing a URI to the operating system to handle (e.g. to open the default mail application for a `mailto:` link).

This is done via the following functions/programs:

- Windows: `ShellExecute*`
- Linux: `xdg-open` (detects desktop environment and calls `gio open`, `gvfs-open`, `gnome-open`, `mate-open`, `exo-open` or `enlightment_open`)
- Mac: `NSWorkspace#openURL()`



General URI handling flow overview

When a user-supplied URL is opened in this way without additional checks, this can lead to code execution:

- By exploiting OS behavior for specific URI schemes and file extensions
- By exploiting vulnerabilities in 3rd party application URL handlers (e.g. this vulnerability in the `steam://` protocol)

Browsers are aware of the potential security implications and disable `file://`-links as one of the most dangerous URI schemes, as well as at least showing a popup before navigating to other external URLs.

While these additional checks have been implemented over time by security-conscious browser developers, they are missing in many other applications.

Finding vulnerable features is straightforward

For any given software, check all features where user-supplied values are opened as URLs (e.g. hyperlinks). If the feature, under the hood, uses the OS to handle the opening and allows arbitrary schemes without comprehensive warning messages, there is likely a way to exploit the feature on certain platforms.

We found that QT's `QDesktopServices::openUrl()` function fulfills the first condition and checked popular QT-based open source software for instances where the function is called with insufficiently validated user input. Tools such as searchcode allow to easily expand a search across millions of indexed open source projects

Operating systems and desktop environments have different URL opening behaviors

From our point of view, the ideal URL opening behavior for an OS includes the following characteristics:

- **Does not automatically mount** previously unmounted file shares without a comprehensive user warning as simply mounting an `smb` share can cause credential leakage
- **Displays a comprehensive user warning** before opening an executable or risky (i.e. `.docm`) file from a remote file share

The remainder of this section contains a detailed write-up of deviations from this behavior we have observed in different operating systems. If you are not interested in those specifics, you can skip ahead to the vulnerabilities section where we demo the different vulnerable desktop applications.

Windows 10 19042

- Executable `.jar` files do not trigger a warning when they are located on a mounted file share (standard JRE installation required)
- UNC paths for all compatible file share protocols cause automatic mounting without a warning:
  - `smb: \\<hostname>\<filename>`
  - `webdav: \\<hostname>\DavWWWRoot\<filename>`
  - `webdavs: \\<hostname>@SSL\DavWWWRoot\<filename>`
- Many applications convert file URLs to UNC paths: `file://<hostname>/DavWWWRoot/<filename>` becomes `\\hostname\DavWWWRoot\<filename>`, allowing one to bypass client-side checks
- When the UNC path points to a file in the root folder of the share, mounting and opening the file is done with a single URL open/click (otherwise, taking two clicks to first mount and then open)

Xubuntu 20.04 (Xfce)

- Executing a `.desktop` file (and therefore running the specified command) does not trigger a warning when it's located on a mounted file share and the file has the executable bit set
- `nfs` URLs cause automatic mounting without a warning/notification and allow for mounting and execution via a single URL open action
- `dav` and `davs` URLs pointing to the root folder of an unmounted share cause automatic mounting. If the server is modified to return a `collection` element in the response to the first PROPFIND request to `/file`, automatic mounting is also done for URLs pointing to specific files on the share
- `dav`, `davs`, `ftp`, `ftps` URLs cause automatic mounting without a warning. When the mounting is initiated by a URL pointing to an executable file, a warning message about the unknown origin of that file is shown after mounting. However, even if the execution is canceled by the user, when the same URL is opened again, with the share now already being mounted, the file is executed and no further warning is displayed
- `smb` URLs initiate a mounting process which shows a connect dialog (not a security warning) that can be confirmed with one click on the pre-selected confirm button
- `sftp` URLs initiate a mounting process which shows the host key confirmation dialog on first connect

Other Linux Operating Systems

The exact opening behavior is dependent on the desktop environment and configuration. After quick review, xfce seems to have the most exploitable features:

- **Auto-mounting:** In Kubuntu/KDE (`kde-open5`), remote files are auto-opened but the remote share is not permanently mounted. When a remote file is opened, it is downloaded and then

- **File types:** .desktop files are often not parsed/executed, but rather opened in a text editor. Since this was unexpected to many, and meant that there was no on-board way to launch .desktop files from the command line, a bug was filed in 2009 https://bugs.launchpad.net/ubuntu/+source/glib2.0/+bug/378783 and many similar discussions can be found on the Internet. In December 2020, the bug was closed with the introduction of a new separate `gio launch` command. Users that have implemented workarounds, for example, by associating `.desktop` files with `dex` are likely still vulnerable.

Snap

Snap apps are subject to an additional URI scheme allowlist. Initially, this list only contained `http`, `https`, `mailto` and `snap`, which broke a lot of applications including Google Chrome. Recently more URI schemes were added.

The snap team has the explicit goal to harden `xdg-open` calls using the following criteria:

- The scheme is understood and documented in the code
- The scheme itself does not cause `xdg-open` to open files (e.g. `file://`)
- It is verified that the recipient of the url (ie, the callee of `xdg-open`) won't process file paths or other arguments in a way that can be leveraged to break out of the sandbox (requires understanding how the url can drive the recipient application)

From a security perspective, this much appreciated. However, the initial usability decrease was high which probably drove users to overall less constrained .deb packages. Even though more URI schemes were added, many use cases are still broken. As a rather mundane example, we found that `ftp://`-links are opened (safely in the browser) from the “normal” Telegram desktop app, but nothing happens in the snap version (“user-open error: supplied URL scheme “ftp” is not allowed")

Mac (Catalina 10.15.6)

- `smb` URLs open a connect dialog. Confirming the the connection will mount the share and open a Finder (file explorer) view
- `smb` URLs to specific files on shares are interpreted as URLs to a share’s root folder. They trigger another connect dialog even if the real root folder has been mounted already. Confirming that connection adds an additional entry with the name of the referenced file to `/Volumes`
- Files on already mounted shares can be referenced via `file` URLs pointing to `/Volumes/<share_name>/<file_name>`
- `file` URLs to specific files open the file with the associated application only for some file types like `.txt`, and only if the file has been opened manually before (for example by double-clicking in the Finder)
- In all other tested cases, `file` URLs pointing to specific files open up the Finder for the containing folder rather than opening the file

Please note: Besides these different combinations of file share URI schemes and extensions, many more dangers can be introduced with custom URL scheme handlers, independent of the OS. In this blog post, we disclose one such RCE in a 3rd party application that allows for arbitrary code execution without additional user interaction. In an upcoming blog post, we’ll explore a similar vulnerability in a Windows 10 default URI handler.

Summary of preconditions, observed behavior and exploitation strategies

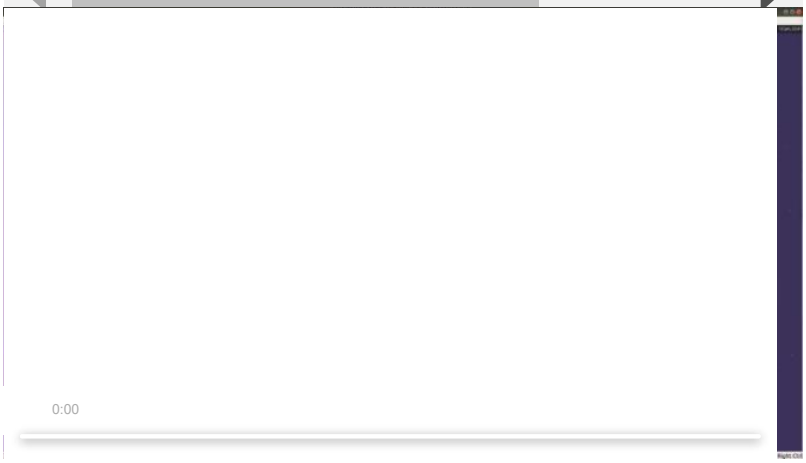
Vulnerabilities

Nextcloud

The Nextcloud Desktop client uses `QDesktopServices::openUrl` in various places, however, the most interesting case is when the user connects to a Nextcloud server. In this case, the server’s login page is loaded in a WebView. QT's default behavior is that a click on a link in a WebView does not directly call the OS' handler, so it's safe against our attack. However, the Nextcloud code was specifically intercepting those requests and passing them to `QDesktopServices::openUrl` by overwriting `acceptNavigationRequest()`

```
            QDesktopServices::openUrl(url);
            return false;
    }
```

Without any filtering on the URI scheme, this gives many possibilities and allows for smooth exploitation without additional confirmation as shown in the video in the Introduction section. The following two videos show an alternative exploit strategy for Xubuntu (using `sftp://`) and exploitation on Windows in combination with a vulnerable URI handler:



Alternative exploitation on Xubuntu via the `sftp://` scheme

Exploitation on Windows, here using a vulnerable URI handling application. Alternatively, a file:// URI pointing to a remote executable/jar file could also be used

The issue has been fixed by the Nextcloud team by replacing `QDesktopServices::openUrl` with their utility function `Utility::openBrowser`, which implements an additional AllowList-check (`http`/`https`/`oauthtest`) before passing it to `QDesktopServices::openUrl`.

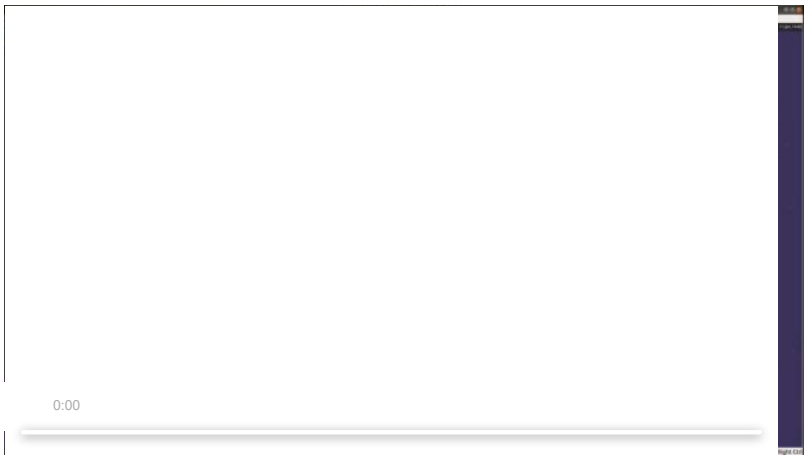
CVE: CVE-2021-22879
Patch: Validate sensitive URLs to only allow http(s) schemes
HackerOne report: Nextcloud Desktop Client RCE via malicious URI schemes
Security Advisory: https://nextcloud.com/security/advisory/?id=NC-SA-2021-008

Telegram

The Telegram Desktop Application for Windows/Linux/Mac OS seemed like an interesting target because it's based on Qt and passes links directly to `QDesktopServices::openUrl`.

While Telegram optionally supports End-to-End-encrypted chats, the Desktop Application only supports non-E2E-encrypted chats. In this case, Telegram makes use of their ability to filter the sent URLs

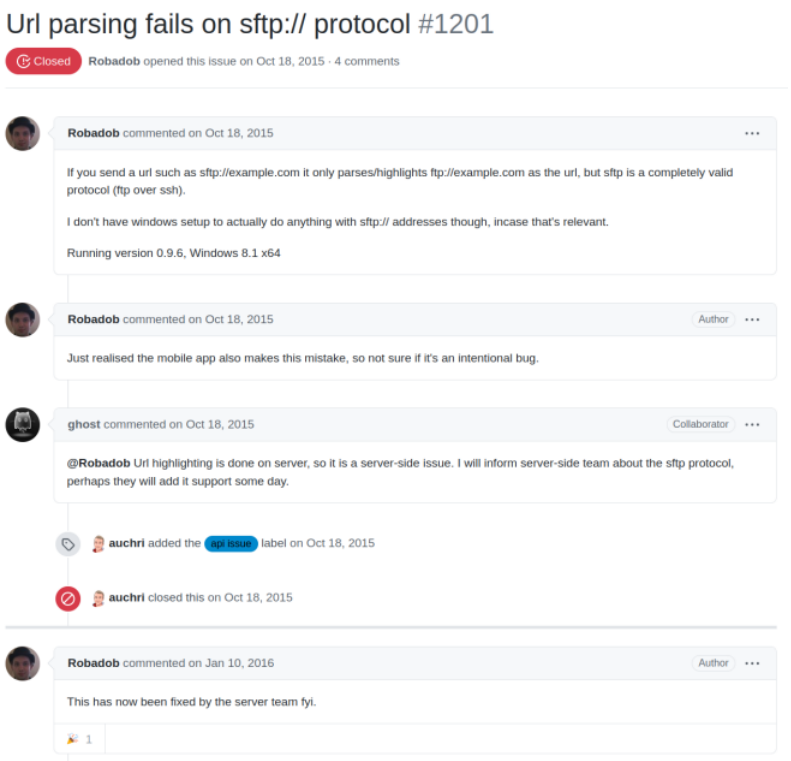
With `messageEntityTextUrl`, any text can point to any URL (offset and length define which part of the message to underline while the `url` parameter defines the destination). In this case, the backend performs strict checks on the URL and in many cases returns a "400 - Unsupported url protocol". We found that for `messageEntityUrl` items (which have no URL field, so the underlined text is also the link destination), a slightly more relaxed filter list is used that allows the `sftp://` URI scheme. On Xubuntu, this can be exploited by linking to an executable .desktop file via `sftp://` (including the username and with an empty password set on the server for minimal interaction):



Exploitation on Xubuntu

In a default Windows installation, there are no applications installed for handling `sftp://` links. However, our testing machine had WinSCP installed which by default registers itself as `sftp://` URI handler. Having been downloaded 150 million times, WinSCP is popular and almost without competition as `sftp`/`scp` client for Windows, so we had a quick look to see what's possible. Check out Bonus-Vulnerability: WinSCP below to see the code execution on Windows (with WinSCP installed).

Interestingly, we could trace back the different treatment of `sftp://` to a Github issue from 2015, where a user observed and reported a seemingly surprising behavior, and the URI scheme was added without an actual use case.



sftp link handling introduced in a Telegram GitHub issue from 2015

VLC

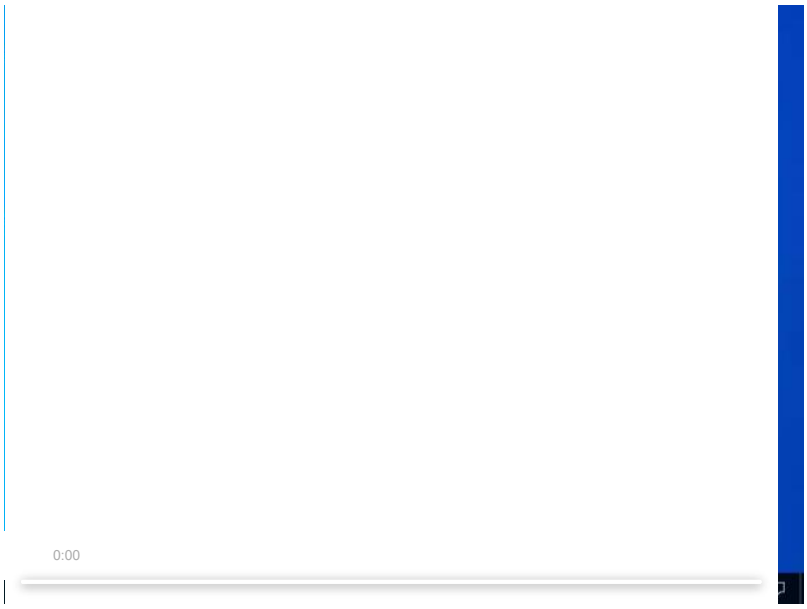
In the case of VLC, it might not be so obvious that the exploited functionality is opening a URL under the hood. The vulnerable feature is the “Show Containing Folder…” action in the context menu of a playlist item.

When clicking the item, the path of the containing directory is fetched and opened by `QDesktopServices::openUrl`. By adding an additional `/` or `/doesnotexist.mp4` to a playlist entry’s URL, “Show Containing Folder…” can be diverted to open files with the associated default application.

However, before being passed to `QDesktopServices::openUrl`, the URI is fed through the following functions:

- `vlc_uri2path`: This VLC function contains code that looks like it was intended to filter out UNC paths, but the relevant code seems to be unreachable and the function executes succesfully for UNC paths
- `FromLocalFile`: Returns a QUrl representation of a "local file" string, but maybe suprisingly also support remote files ("This function also accepts paths with a doubled leading slash (or backslash) to indicate a remote file, as in `//servername/path/to/file.txt`")
- `isLocalFile`: Returns `true` for any URI starting with `file:` ("Note that this function considers URLs with hostnames to be local file paths")

As all of those functions allow remote `file://`-paths, this can be exploited on Windows using a malicious playlist file containing a file URL or UNC path pointing to a .jar file on a WebDav share:



Exploiting VLC on Windows

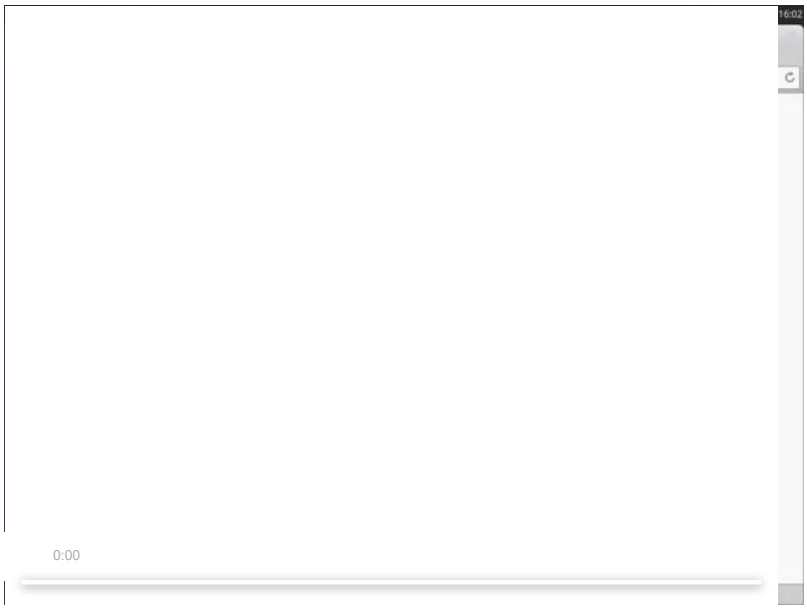
Exploitation on Linux is heavily restricted due to the URI scheme limitation. As no remote files can be referenced to force an auto-mount, only already-mounted remote shares (e.g. where the playlist resides) can be referenced using the special `/run/user` directory (e.g. `file:///run/user/1000/gvfs/sftp:host=<host>,user=<user>`)

The issue was mitigated by adding a check to ensure that the opened URI is a directory, preventing the RCE.
`vlc_uri2path` was not changed. So although the function may appear to have the goal of disallowing remote files, UNC files can still be specified (leading to an NTLM hash leak or potentially other unexpected behavior when `vlc_uri2path` is used).

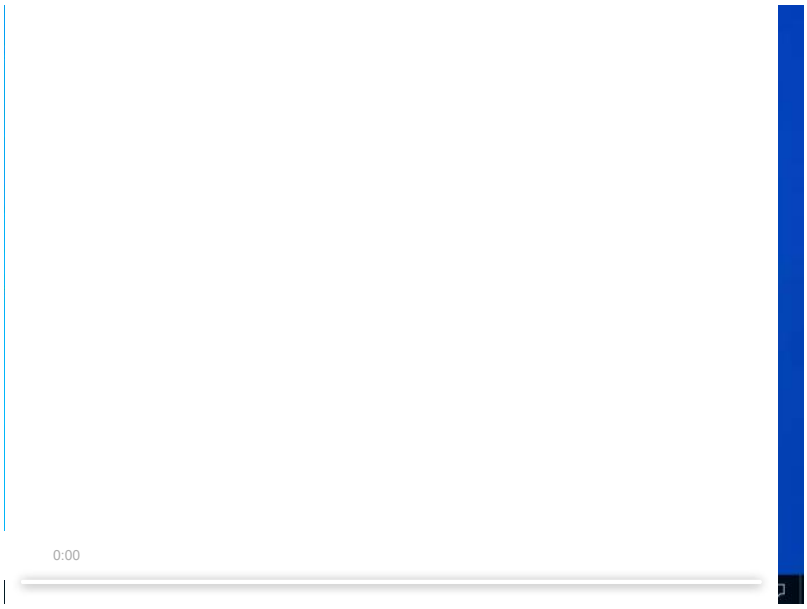
We reported the vulnerability to VLC on January 18th, together with 3 patch candidates. One candidate was merged on Feb 08th and the patched version 3.0.13 will presumably be released next week.

Pending Security Advisory: https://www.videolan.org/security/sb-vlc3013.html

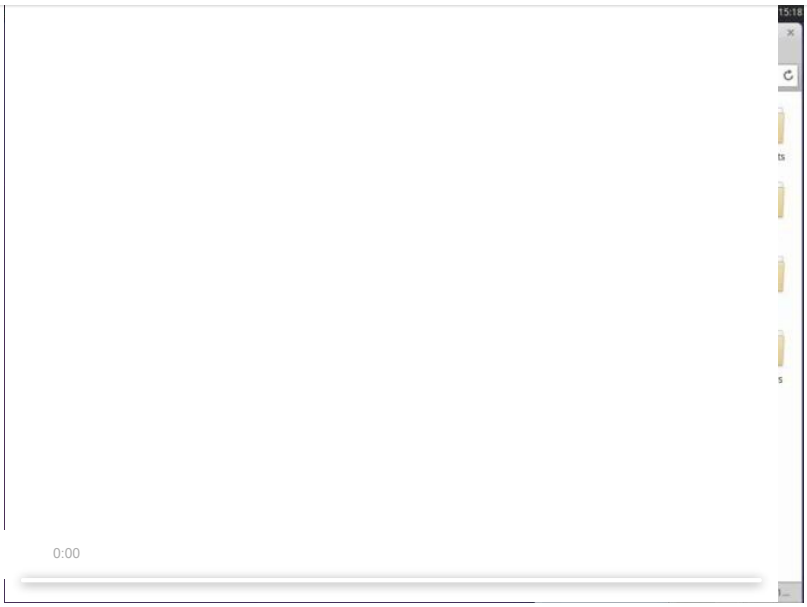
Hyperlinks can be CTRL-clicked, sending them on to a call to `ShellExecute` on Windows, or `xdg-open` on Linux. One-click exploits are shown for OpenOffice on Windows and Xubuntu, as well as LibreOffice on Xubuntu:



Exploiting OpenOffice on Xubuntu

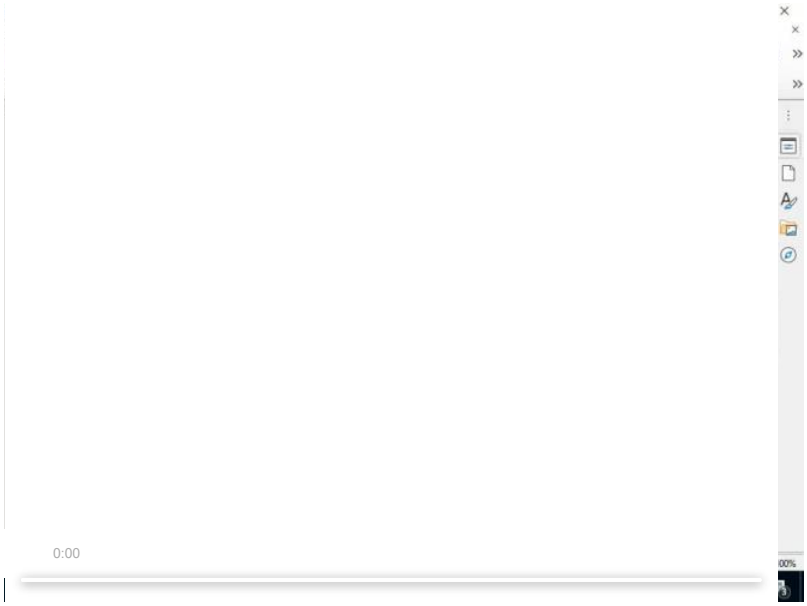


Exploiting OpenOffice on Windows



Exploiting LibreOffice on Xubuntu

In the Windows version of LibreOffice, a file extension blacklist aimed at protecting against this type of attack was implemented long before our research. However, we quickly found a way to bypass this blacklist, allowing for 2-click exploitation on Windows, showcasing the unreliability of such an approach.



Bypassing the LibreOffice file extension blacklist on Windows

The office suites allow for files with pretty complex content and functionality and are sometimes used in contexts where niche features like `ftp` hyperlinks in documents are actually expected to work. Therefore, a fully restrictive fix would not be possible without a potentially critical impact on the user experience. As a fix, we suggest displaying a comprehensive warning message to the user before opening any non http(s) hyperlinks. This would match the behavior displayed by the Microsoft Office suite.

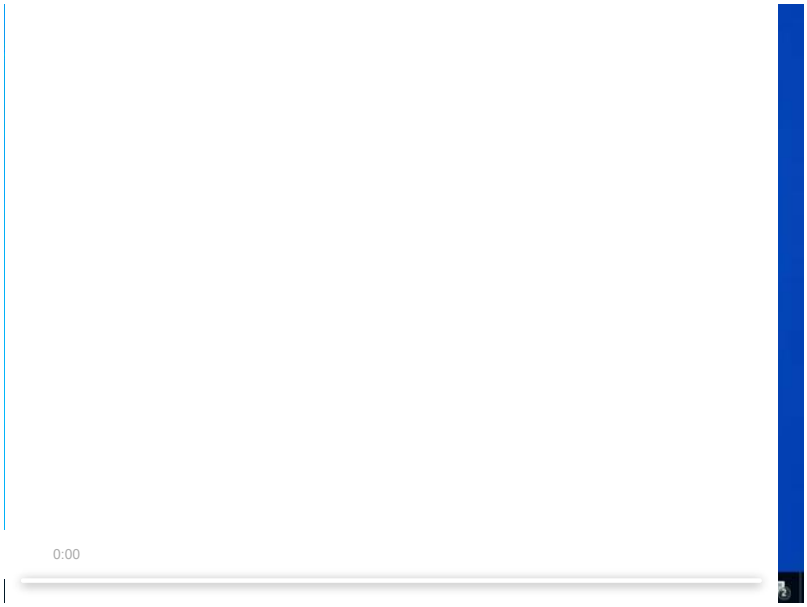
OpenOffice response

Pending CVE: CVE-2021-30245
Pending Patch: The OpenOffice team is currently working on a fix which addresses the issue on all platforms to be included in the upcoming 4.1.10 release.

LibreOffice opted to only patch the file extension blacklist bypass for Windows. CVE-2021-25631 was assigned for the blacklist bypass.
Regarding the Xubuntu/xfce exploit, they argued that protecting against the showcased 1-click code execution is not their responsibility as the app developer.
As a result, LibreOffice on Xubuntu is still vulnerable to the exploit shown above, and at the time of writing the team has no intention of publishing a patch to fix the issue.

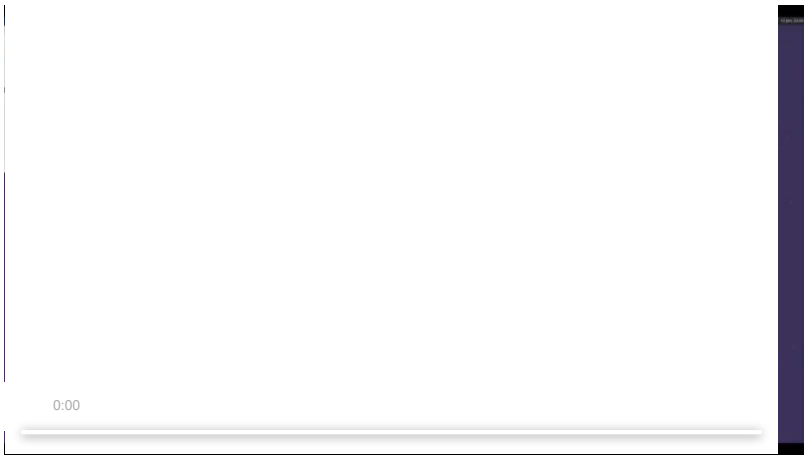
Mumble

The Mumble voice chatting software features a centrally managed public server list which makes it convenient for users to find and connect to servers that have opted in to be listed. In addition to the server name, server operators can provide a URL meant to link to an associated website. When a user chooses the `Open Webpage` action in the context menu for a publicly listed server, the URL is passed to `QDesktopServices::openUrl()`



Exploiting Mumble on Windows

In the video, the `Open Webpage` action is selected twice to first mount the file share and then execute the hosted .jar file. A one-click exploit for windows should have been possible by using a webdav URL which points to a malicious file in the root folder of the share. We do not have video evidence of that as we discovered this optimization later when the patch was already underway and we did not want to mess around with the public server list anymore.



Exploiting Mumble on Xubuntu

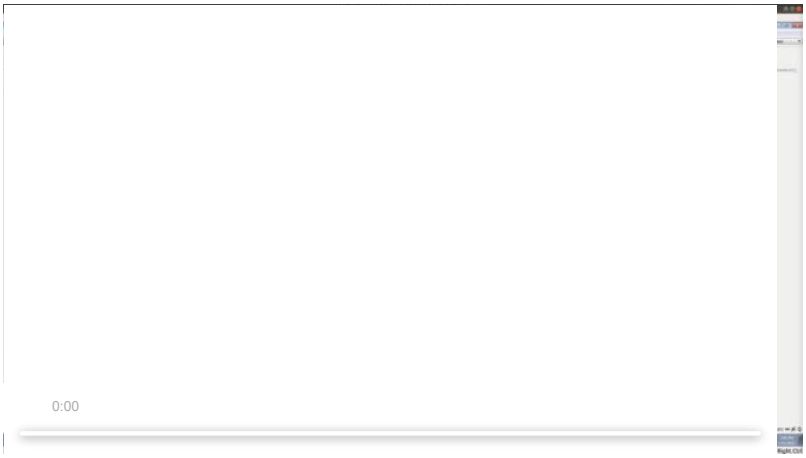
In the above video the user must confirm an OS dialog to connect to the public smb share. One-click RCE should again also be possible here using an nfs share, but we did not create a PoC due to

server list with entries enticing users to perform the `Open Webpage` action and gain widespread code execution (i.e. “Free Mumble server – Visit our website to get your own!”).

CVE: CVE-2021-27229
Patch: Restricting allowed schemes to http and https

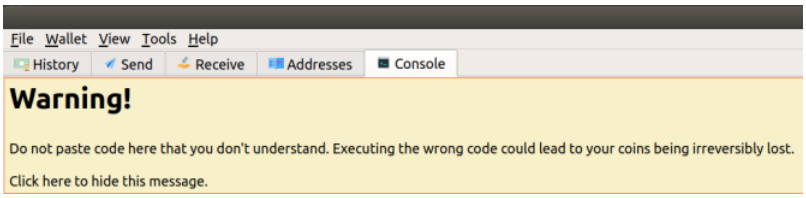
Bitcoin/Dogecoin Wallets

The Bitcoin-Qt client (and any Altcoin using its codebase) allows users to specify the blockchain explorer website they want to use in the GUI settings window by defining a list of URIs (split by '|'). Those entries are then shown in the context menu for a transaction where no validation is performed on the URI scheme, and the URI is opened with its default application:



Exploiting Bitcoin wallet on Windows

While some social engineering is required to get the victim to add their malicious URI, in the world of cryptocurrency scams, where users transfer their coins in the hope to receive them back doubled, this is still quite a low bar. Scammers have previously also instructed their victims to run malicious commands in the client's RPC console, after which warning messages were added to the console:



Warning message in RPC Console

Adding Blockchain Explorer URLs to the average user should seem less dangerous/more of a normal interaction than pasting code snippets into the RPC console.

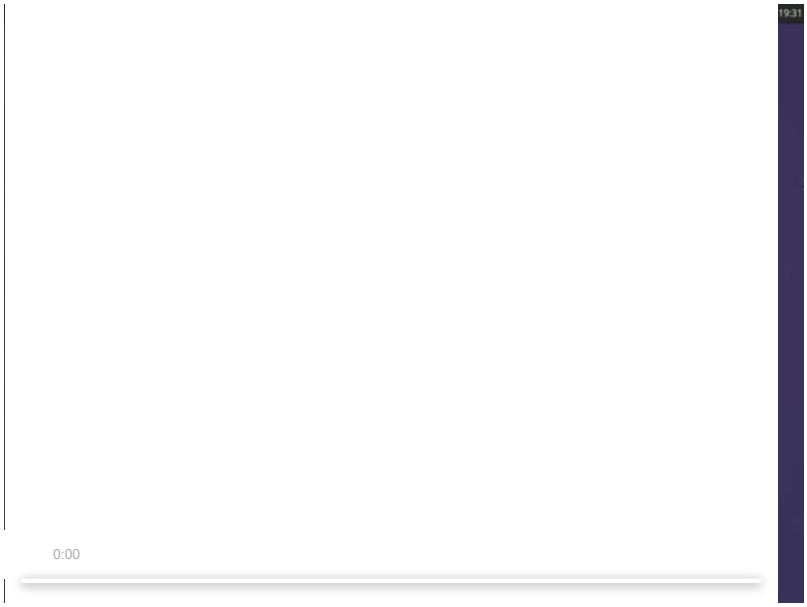
The issue was disclosed to Bitcoin Core, Bitcoin Gold, Bitcoin Cash, Bitcoin ABC and Dogecoin on January 18th.

Dogecoin: Fixed in v1.14.3 (released Feb 28th).
Bitcoin ABC: Fixed in version 0.22.15 (released March 9th).
Bitcoin Cash: Fixed in version 23.0.0 (released April 15th).
Bitcoin Gold: Initially responded, that "per the scoring matrix, it does not qualify as a security vulnerability" and closed the report. After sharing this blog post draft, a patch was developed and merged into master (but no new version released yet).
Bitcoin: No fix planned.

Wireshark

The QT based Wireshark packet analyzer application makes some fields which contain URLs double-clickable. These URLs were simply passed to `QDesktopServices::openUrl`, allowing for exploitation via malicious capture files or the live capture of maliciously crafted traffic.

Exploiting Wireshark on Windows



Exploiting Wireshark on Xubuntu

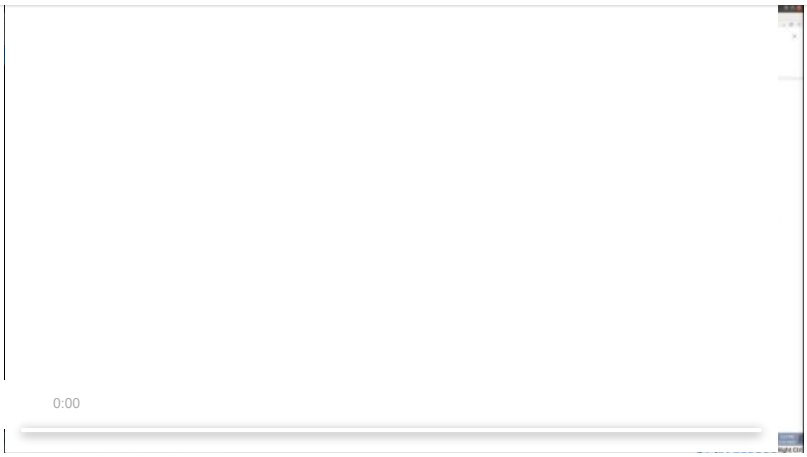
CVE: CVE-2021-22191
Patch: Changing double-click behavior to copy URLs to the clipboard rather than opening them

Bonus-Vulnerability: WinSCP

WinSCP is a remote file manager that installs itself to handle many protocols including: `sftp://`, `ftp://`, `ftps://`, `ftpes://`, `scp://`, `ssh://`, `dav://`, `davs://`, `s3://`, `winscp-sftp://`, `winscp-ftp://`, …

While it's expected that e.g. a username can be provided in the URI, the documentation also mentions WinSCP-specific parameters to save session data, e.g. `winscp-sftp://fingerprint=ssh-rsa-xxxxxxxxxxx...=@example.com/;save`.

Those parameters are so-called “Raw Settings”. Exploring the corresponding docs page, there seemed to be a few potentially dangerous parameters. Specifically, when setting `ProxyMethod` to `5` (Local) and a command as `ProxyTelnetCommand`, the provided command is executed immediately when the link is opened.



Combining the Telegram and WinSCP vulnerability for a Windows exploit

The following video shows exploitation from a website:

Exploiting the WinSCP vulnerability from Google Chrome

It's interesting to note that Chrome is neither showing the full URL nor the application that will be opened.

CVE: CVE-2021-3331
Patch: Prevent loading session settings that can lead to remote code execution from handled URLs

Systematic mitigation requires contributions from OS, Framework, and Application maintainers

This issue spans multiple layers in the targeted system's application stack, therefore making it easy for the maintainers of any one to shift the blame and avoid taking on the burden of implementing mitigation measures on their end. However, due to the diversity of client systems and their configuration states, it is crucial that every party involved takes on some amount of responsibility and adds their contribution in the form of mitigation measures:

For OS/Desktop environments:

- Remote shares should not be auto-mounted
- Appropriate warning messages should be shown (e.g. when calling `xdg-open` on a remote `.desktop` file on Xubuntu)

For Frameworks:

- Applications that let users open external URLs should validate the URLs with a URI scheme allowlist
- Applications that register themselves as a URI scheme or file extension handler need to take extra care not to introduce a vulnerability that can then be exploited from numerous other unhardened applications

Conclusion

In this post, we have explored URL handling Operating System behavior and application vulnerabilities. While most issues were quickly fixed by the developers, the following applications are still vulnerable as of 2021-04-15:

- **Bitcoin (and Bitcoin Gold) Desktop Clients:**
  It is quite surprising and noteworthy to see forks taking the issue more seriously and implementing measures to protect their users which Bitcoin does not
- **LibreOffice:**
  They did not consider it their responsibility to protect against the Xubuntu variant.
  Our recommendation to replace the file extension blacklist for Windows with a more robust measure was dismissed, even though we showcased its general unreliability by pointing out missing file extensions, as well as, the (now fixed) bypass we promptly discovered.
  Both versions will also stay susceptible to exploitation in case of other vulnerabilities in 3rd party URL handlers (see the WinSCP vulnerability shown here as an example)
- **OpenOffice:**
  A fix is scheduled to be released in the upcoming 4.1.10 version.
  We would like to use this opportunity to remind users that all files from untrusted sources (including non macro enabled documents) should be handled with utmost caution
- **VLC:**
  The patched version 3.0.13 was initially scheduled for before April 9th but its release has been postponed. It's now expected for next week

The issues were easy to find and we had a high success rate when checking applications for this vulnerability. Therefore, we expect more vulnerabilities of this type to be discovered when looking at other applications or UI frameworks.

Follow us on Twitter (@positive_sec) to keep up to date with our posts.