Federal Office
for Information Security

# RdpCacheStitcher

https://github.com/BSI-Bund/RdpCacheStitcher

https://www.bsi.bund.de

## Table of Contents

1 Overview....................2
2 Prerequisites....................2
2.1 Linux....................2
2.2 Windows....................2
3 Workflow....................3
3.1 Starting a new case....................3
3.2 The work area....................3
3.3 Reconstructing images....................5
3.3.1 Manually placing a tile....................5
3.3.2 Getting hints for empty cells....................6
3.3.3 Autoplace....................8
3.4 Working with multiple screens and exporting images....................8
4 Keyboard shortcuts....................9
5 License and sources....................9

# 1 Overview

To minimize bandwidth consumption during an RDP session, a Windows client keeps a cache of screen contents on disk that often will not get deleted when the session ends. This makes this cache files of great interest to forensic investgators, providing a chance to look over the shoulder of the user after the fact. Tools exist that are capable of parsing and extracting the contents of this cache, for example the BMC-tools by the French agency ANSSI (https://github.com/ANSSI-FR/bmc-tools).

The problem is that this RDP cache consists of a lot of very small tiles, usually several thousands of them and mostly unordered, and reconstructing meaningful images out of these tiles manually is often extremely hard or even practically impossible.

This is where *RdpCacheStitcher* comes into play. It provides the forensic analyst with a graphical user interface that helps reconstructing multiple images from a collection of RDP cache tiles and supports this tedious work with several heuristics that try to guess which tiles belong together.

# 2 Prerequisites

64 bit binaries for Linux and Microsoft Windows are provided. In principal, *RdpCacheStitcher* should be able to run on OS X too, but you have to compile the sources yourself in that case.

## 2.1 Linux

The Linux binary depends on external Qt widget libraries to run. On Ubuntu, you can install these with the command `sudo apt install libqt5widgets5`.

## 2.2 Windows

The Windows version depends on the Microsoft Visual C++ 2017 Redistributable (64 bit) package, which in case it is not already installed you can download directly from Microsoft under https://aka.ms/vs/16/release/vc_redist.x64.exe.

# 3 Workflow

This section describes the typical workflow when trying to reconstruct useful images out of RDP cache tiles. First, a new case is opened by reading in a collection of these tiles. Then, multiple images can be constructed on different screens supported by several helper tools and options. Finally, the resulting images can be exported to PNG image files.

## 3.1 Starting a new case

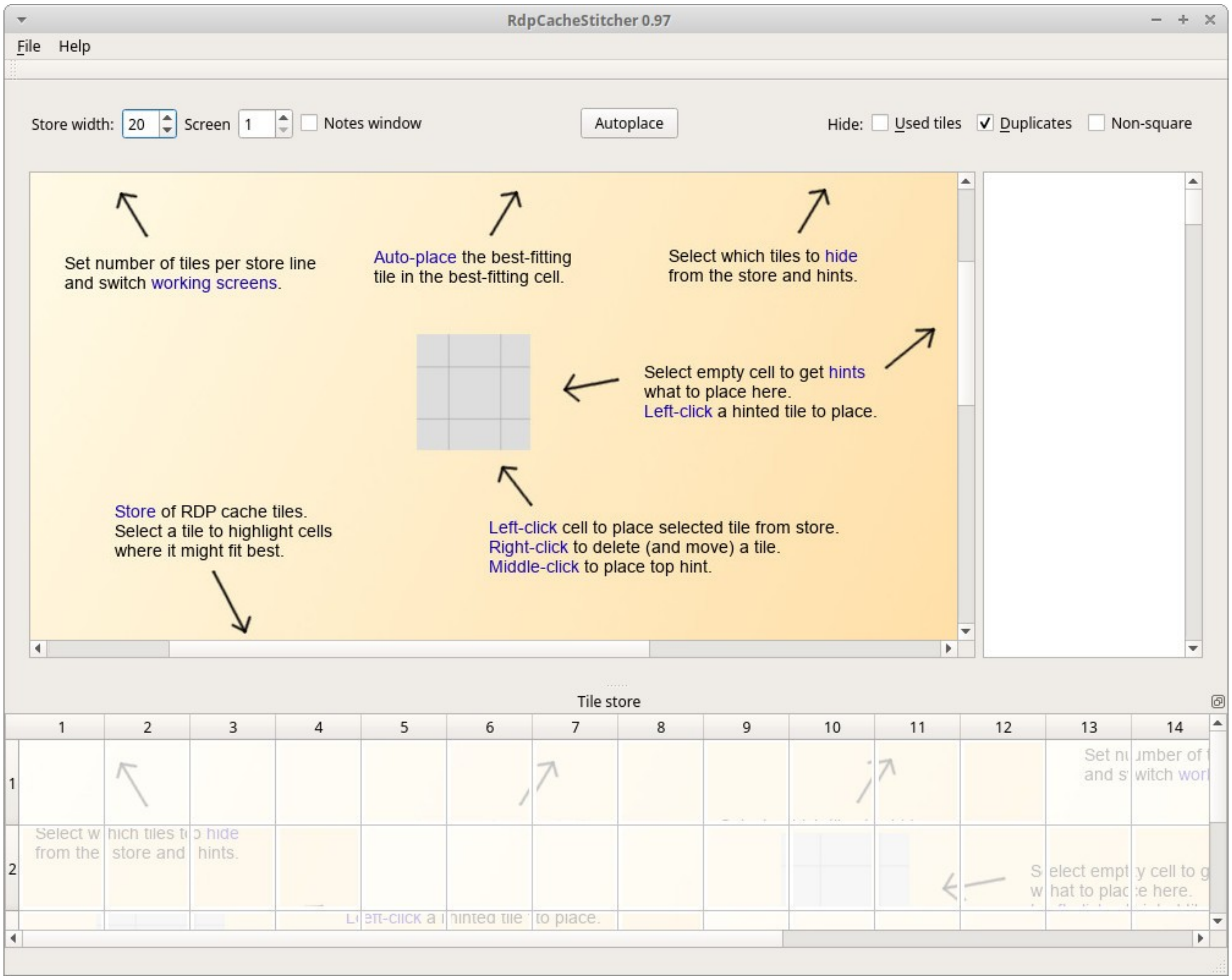
The input for *RdpCacheStitcher* is a collection of images in *.bmp* bitmap format extracted from an RDP cache by third-party tools like the *BMC-tools* mentioned in section 1. This is called a *case*. From the menu, select `File → New case…` and select the directory where the *.bmp* tile bitmaps are located. *RdpCacheStitcher* will now read in these tiles, doing some calculations in the background for later use. When all tiles are read in, a pop-up window will inform you how many tiles have been imported successfully, how many duplicate tiles (tiles which are identical visually) and non-square, rectangular tiles have been detected and how many tiles failed to load.

Duplicate and non-square tiles can be hidden from view to reduce clutter, which will be explained in more detail below.

Note that after the tiles have been imported, the directory containing the tiles is no longer needed. When saving a case to disk via `File → Save case (as…)`, all information belonging to the case, including the tile graphics, are stored in a single `.rcs` case file.

## 3.2 The work area

The main window is divided into several areas.



Store width: 20 Screen 1 ☐ Notes window

Hide: ☐ Used tiles ☑ Duplicates ☐ Non-square

Set number of tiles per store line and switch working screens.

Auto-place the best-fitting tile in the best-fitting cell.

Select which tiles to hide from the store and hints.

Select empty cell to get hints what to place here. Left-click a hinted tile to place.

Store of RDP cache tiles. Select a tile to highlight cells where it might fit best.

Left-click cell to place selected tile from store. Right-click to delete (and move) a tile. Middle-click to place top hint.

In the middle, using up most of the area, the current **screen** is shown where tiles can be placed to reconstruct a meaningful image (initially empty, signified by a grid of grey **cells**).

Below the screen area is the **tile store** which contains all tiles from the current case. The number of tiles per line can be set by changing the “Store width” control in the upper left corner. Tiles already used are displayed transparent, and certain tiles (already used, duplicate and rectangular tiles) can be hidden by checking the respective boxes in the top right corner.

Note that you can drag the border between the screen area and the tile store to adjust the size of the store, and even drag the tile store to change it into a floating window which you can then move and resize individually, for example for to place it on a second display. Use the small icon in the top right corner to dock it to the main window again.

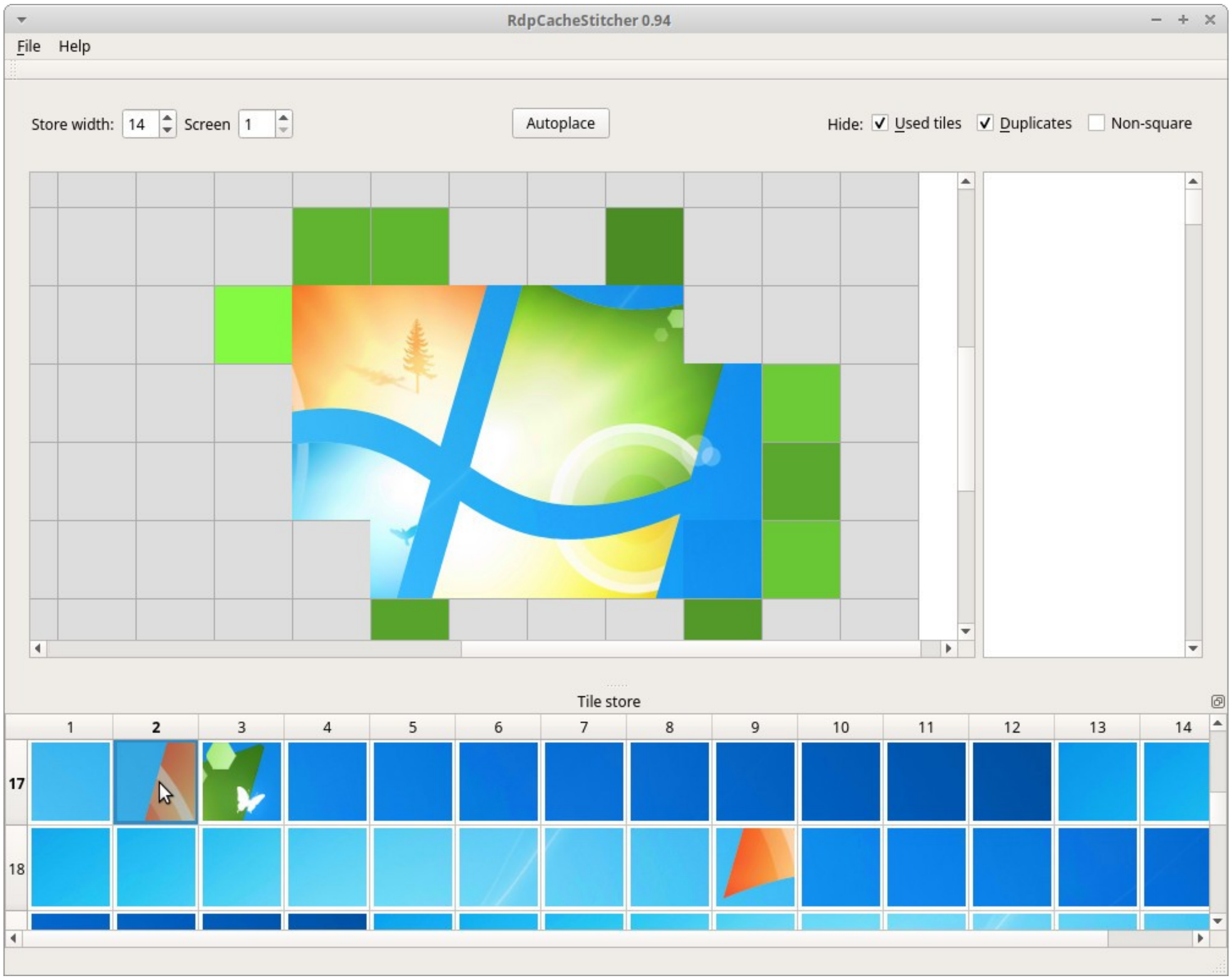
To the right of the screen is an initially empty area that can display a list of tiles best fitting to a specific cell (**hints**), which will be populated when clicking on an empty cell (see below).

## 3.3 Reconstructing images

There are three ways how tiles can be added to a screen: Manually by selecting them in the tile store and putting them into a cell, by selecting an empty cell and get hints what tiles might fit there, and by automatically placing the best hint.

### 3.3.1 Manually placing a tile

If you select a specific tile in the tile store, the program tries to guess where it might fit best. Empty cells adjacent to other tiles are coloured in green depending on how well the tile would fit there: The lighter the color, the more the program thinks it should go there.



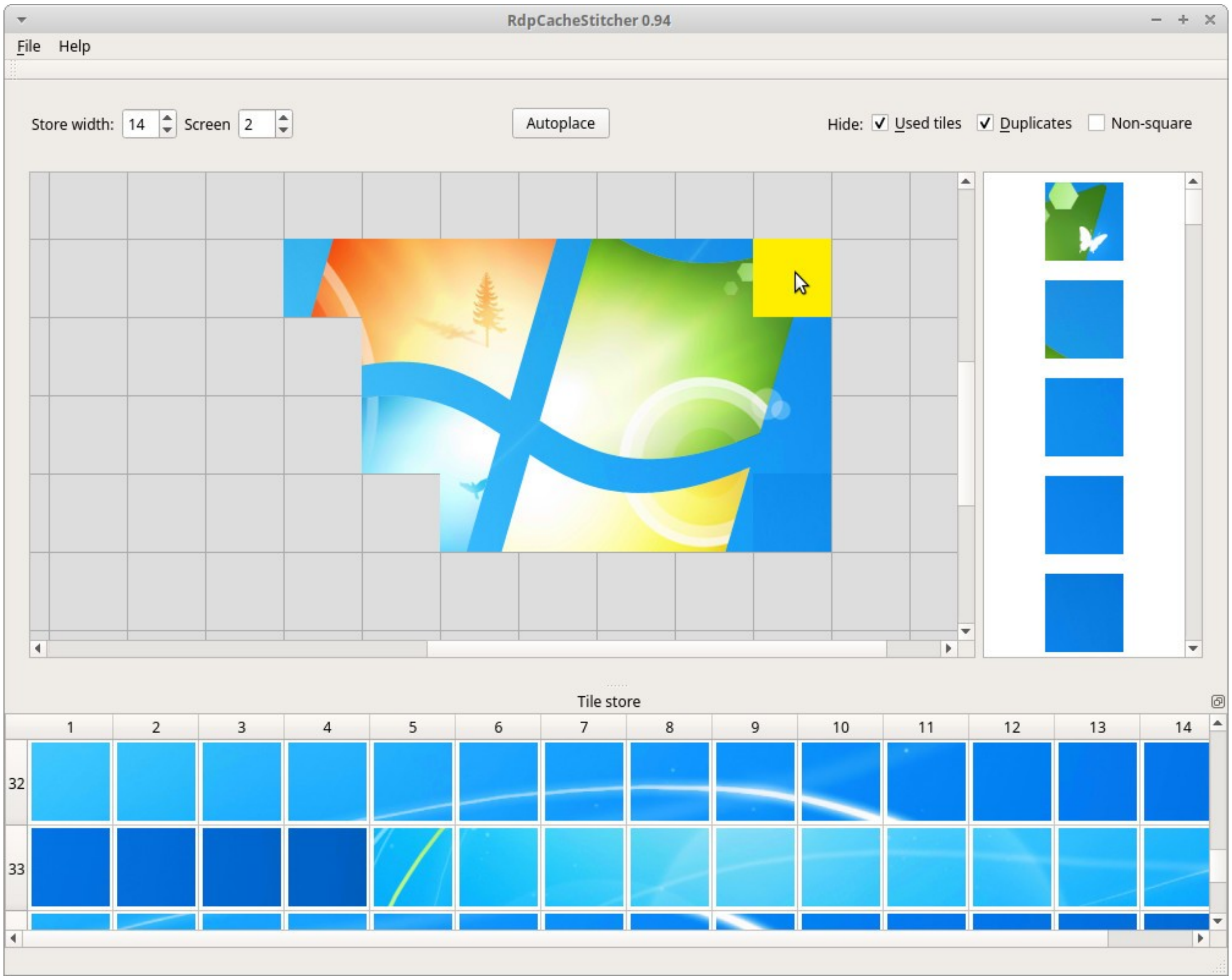
For example, in the screenshot above the second tile in the store is selected and *RdpCacheStitcher* proposes several empty cells, correctly guessing that the cell in the top left would be the best fit.

A **left click** in a cell on the screen places the currently selected tile from the store. If you place a tile near on of the borders, the available screen space grows in that direction so that you don’t have to worry about running out of space in case an image gets unexpectedly large.

**Right-clicking** a tile in the screen area deletes it. It also automatically selects the tile in the store, which means you can immediately place it again elsewhere, i.e. you can move an already placed tile that way.

### 3.3.2 Getting hints for empty cells

If you are not sure what tile to place into an empty cell adjacent to other tiles on the screen, *RdpCacheStitcher* can give you some suggestions. If you **left-click** an empty cell, the area to the right gets populated by tiles that might fit there, sorted by how well they seem to fit.



Store width: 14 Screen 2

Hide: ☑ Used tiles ☑ Duplicates □ Non-square

In the screenshot above, the cell in the top right corner of the image is selected, marked in yellow. To the right, hints for what to place there are displayed, the top tile actually being the correct one.

If you hover the mouse over one of the hints, the respective tile gets shown in the selected cell as a preview for you to see how it would look there. **Left-click** a hint to place the tile in the selected cell.

Alternatively, if you see that the top hint already is the correct tile for a selected cell, you can **middle-click** the cell to automatically place the top tile from the hints so you don't need to move the mouse over to the right and back. In fact, you can also middle-click an empty cell without displaying the hints first, if you feel like the program will be able find the correct tile in this case. This works well for complex graphics and less well for tiles with text or very few colors on their borders.

Note that the *hide used*, *hide non-square* and *hide duplicate* tiles checkboxes in the upper right not only hide the respective tiles from the tile store view, but also from being used as hints. Sometimes it can be of benefit to hide already used tiles when relying on hints a lot while reconstructing an image.

### 3.3.3 Autoplace

The *autoplace* button at the top tries to find the best fitting tile for any of the empty cells. For every empty cell adjacent to a tile on the screen, the best fitting tile is computed, and from all these best fitting tiles, the overall best is placed automatically.

Since you don't have control over which cell gets filled next by *autoplace*, this feature will probably less useful in the general case. It can however be surprisingly helpful for quickly reconstructing larger areas of complex, photo-like images like desktop backgrounds or the borders of a window overlapping such an image.

Note that again the *hide used*, *hide non-square* and *hide duplicate* tiles checkboxes exclude the respective tiles from being considered by *autoplace*. Especially the *hide used* checkbox should usually be ticked when using *autoplace*.

## 3.4 Working with multiple screens and exporting images

Usually, analysts will want to reconstruct multiple and different images from one RDP cache, for example images from different points in time. *RdpCacheStitcher* supports this with the ability to switch the current working screen using the *Screen* control in the upper left corner. Each screen can have a different size, and all screens will be stored in a *.rcs* file when saving a case.

In addition, the program allows an analyst to take notes. Use the *Notes window* checkbox to show or hide the *Screen notes* window containing a text editor where you can write down notes which are individual per screen.

The reconstructed images together with their respective notes can be exported as *.png* image files with the `File → Export screen images…` menu item. Choose a directory and a name prefix, and the images will be exported with a consecutive number added to that prefix.

Only screens that actually contain any tiles are exported, and the resulting images are cropped to minimum size. Empty cells will be exported as transparent areas.

Alongside the *.png* image files, a *.txt* file is written which contains all the notes corresponding to the exported screen images, if any.

# 4 Keyboard shortcuts

The following keyboard shortcuts are supported by *RdpCacheStitcher*:

| | |
|---|---|
| `Alt + n` | Hide/show screen notes window |
| `Alt + u` | Hide/show used tiles |
| `Alt + d` | Hide/show duplicate tiles |
| `Ctrl + n` | New case... |
| `Ctrl + s` | Save case |
| `Ctrl + o` | Open case… |
| `Ctrl + e` | Export screen images… |
| `Ctrl + q` | Exit |

# 5 License and sources

RdpCacheStitcher is copyright 2020 Bundesamt für Sicherheit in der Informationstechnik (BSI).

RdpCacheStitcher is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

RdpCacheStitcher is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with RdpCacheStitcher. If not, see <https://www.gnu.org/licenses/>.

RdpCacheStitcher uses the open source (L)GPL v3 version of Qt, which you can download at http://download.qt.io/archive/qt/.