

# Chrome++ , ложные срабатывания антивируса и защита сборки

## Что такое Chrome++ и почему ему можно доверять

Chrome++ (репозиторий [Bush2021/chrome\\_plus](https://github.com/Bush2021/chrome_plus)) — давно существующий, широко используемый и уважаемый проект с открытым исходным кодом под лицензией GPL-3.0. Его код открыт и доступен для проверки кем угодно.

Именно он делает сборку портативной: рядом с `chrome.exe` кладётся его `version.dll` — «прокси-DLL». `chrome.exe` импортирует `VERSION.dll`, этого имени нет в списке `KnownDLLs`, поэтому браузер грузит соседнюю обёртку вместо системной, и профиль остаётся в `Data` рядом с браузером, а не в `%LOCALAPPDATA%`. Это стандартный, документированный приём портативности — тот же, что и у множества других портативных сборок на Chromium.

**Вирусов в нём нет и взяться им неоткуда.** Код открыт и проверяем, а сборка берёт не «что-нибудь из интернета», а конкретный релиз Chrome++ и логирует его `SHA256` — этот отпечаток можно сверить с опубликованным в апстриме. Что скачано, ровно то и уходит в сборку, без подмен.

## Почему антивирус иногда ругается

Неподписанная DLL с именем системной, положенная вплотную к `chrome.exe`, чтобы тот загрузил её вместо настоящей, — для эвристики антивируса это классический признак «DLL hijacking». Эвристика реагирует на **форму**, а не на содержимое: тело файла тут ни при чём.

Поэтому это **ложное срабатывание** (false positive). Тем же приёмом Microsoft Defender периодически ошибочно ловил и сам Google Chrome, и приложения на Electron — это известная и признанная проблема эвристик, а не свойство файла.

Проявляется это грубо: Defender выхватывает `version.dll` прямо в момент сборки, и упаковка архива падает с невнятным `OSError: [Errno 22] Invalid argument` — будто проблема с

диском, хотя на деле файл забрал антивирус (Windows вернул `ERROR_VIRUS_INFECTED`), а С-рантайм свёл неизвестный код к `EINVAL`).

## Что с этим делает программа

- **Галка «Обойти Defender на время сборки»** (`guard_defender`, включена по умолчанию) стоит у команд `Собрать`, `Обновить` и `Chrome++`. На время операции папка исключается из Defender и возвращается **сразу после — даже если сборка упала**. Windows один раз спросит UAC. `Собрать` исключает `output`; `Обновить` и `Chrome++` — ещё и папку обновляемой сборки (она обычно вне `output`).
- **Если Defender не запущен** — галка ничего не делает и UAC не спрашивает. Проверка идёт без прав администратора (служба `WinDefend`), так что на машинах без Defender сборка идёт как раньше.
- **Команда `Defender: убрать исключение`** (раздел `Обслуживание`) снимает исключение вручную — на редкий случай, когда прогон убили жёстко и исключение осталось. Снять исключение, которого нет, — не ошибка.

Механизм безопасен по устройству: исключение живёт только на время сборки и держится тремя предохранителями (файл-замок, слежение за родительским процессом, жёсткий таймаут), поэтому оно не может остаться открытым незаметно.

## Две обёртки на выбор

Обёртка выбирается в поле `Портативность` - и при сборке, и при обновлении:

| Вариант                        | Что кладётся в <code>App</code>                      | Реестр                                          | Антивирус                               |
|--------------------------------|------------------------------------------------------|-------------------------------------------------|-----------------------------------------|
| <code>Chrome++</code>          | <code>version.dll</code> и <code>chrome++.ini</code> | ветка стирается при выходе, если включена галка | эвристики регулярно принимают за угрозу |
| <code>Прокси библиотека</code> | <code>version.dll</code> и <code>version.ini</code>  | запись блокируется, копиться нечему             | Microsoft не ругается, см. ниже         |

Профиль в обоих случаях остаётся в папке сборки: библиотека держит его сама.

**Прокси библиотека** - проект [neurostalker/proksi-biblioteka](https://github.com/neurostalker/proksi-biblioteka) на GitFlic, исходники на Delphi лежат в том же репозитории. Она делает ту же работу, что и Chrome++, и умеет сверх того: блокировать запись в реестр, снимать бит защиты процесса (это нужно ядрам Chromium 118+ и 126+),

убирать метрики и не давать браузеру создавать папки за пределами профиля. Разрядности только x86 и x64 - ARM64 закрывает лишь Chrome++.

`version.ini` программа пишет свой, а не берёт авторский. В авторском по умолчанию подменяется User-Agent на Яндекс, глушится трафик к серверам Google и блокируются широковебательные рассылки; наш файл делает только портативность: пути `..\Data` и `..\Cache`, спецпапки внутрь сборки, `REGOFF` по галке `Не оставлять следов в Windows`. Остальные ключи описаны в README библиотеки - файл лежит в `App` и правится руками.

Скачанный архив обёртки, как и раньше, логируется с `SHA256` - что скачано, то и уходит в сборку.

## Проверка на VirusTotal, 17 августа 2026

Проверяли обе библиотеки версии 1.0.7.4 из релиза на GitFlic:

| Файл                                         | Размер, байт | SHA256                                                                        |
|----------------------------------------------|--------------|-------------------------------------------------------------------------------|
| <code>version</code><br><code>x32.dll</code> | 39 936       | <code>DBB82B80BB47DFAB47E09DFDA777478EC7A76599A4B47AFBE4D44C77C3EA0E03</code> |
| <code>version</code><br><code>x64.dll</code> | 91 136       | <code>FAEA4A01468ACDD54C390CBC3A18101ADF6F8F51525607EC2CAB19B42235AAAB</code> |

Обе не подписаны (`NotSigned`), издатель в свойствах - `Свободный софт`.

Что показал отчёт по обоим файлам:

- **Microsoft - Undetected.** Это и есть движок Defender, и ради этой строки всё затевалось.
- **NANO-Antivirus, Acronis (Static ML) - Undetected.**
- **Cynet - Malicious (score 100), DeepInstinct - MALICIOUS.** Оба - чисто машинное обучение; ложные срабатывания на неподписанной Delphi-библиотеке, которая перехватывает функции WinAPI, для них обычное дело.

Чего эта проверка **не** доказывает:

- Движок Microsoft в VirusTotal сигнатурный. На живой машине Defender добавляет облачную эвристику и репутацию по редкости файла, которых в скане нет, поэтому вердикт там может оказаться другим.

- Подписи нет ни у прокси библиотеки, ни у Chrome++, так что для SmartScreen обе одинаково «неизвестный издатель».
- На живом Defender мы не проверяли и проверить не могли: на машинах владельца он удалён - служб `WinDefend` и `WdFilter` нет, и там любой файл выглядит чистым.

Отчёт на VirusTotal открывается только через reCAPTCHA, поэтому страницу открывал человек, а не программа: капчу мы не обходим.

## Получателю сборки

Исключение работает **только на машине, где идёт сборка**. У того, кому вы отдали готовую сборку, Defender может так же придаться к `version.dll` при распаковке — и тогда браузер молча перестанет быть портативным (профиль уйдёт в Windows). Помогает одно из: исключить папку в его Defender, отправить файл в Microsoft как ложное срабатывание, или подписать `version.dll` своим сертификатом (на будущее).

# Chrome++ , antivirus false positives, and the build guard

## What Chrome++ is, and why it can be trusted

Chrome++ (the [Bush2021/chrome\\_plus](#) repository) is a long-standing, widely used, and respected open-source project released under the GPL-3.0 license. Its source is public and can be audited by anyone.

It is what makes the build portable: its `version.dll` is placed next to `chrome.exe` as a proxy DLL. `chrome.exe` imports `VERSION.dll`, the name is not in `KnownDLLs`, so the browser loads the neighbouring wrapper instead of the system one, and the profile stays in `Data` beside the browser rather than in `%LOCALAPPDATA%`. This is a standard, documented portability technique — the same one used by many other portable Chromium builds.

**There is no malware in it, and there is nowhere for malware to come from.** The code is open and auditable, and the builder does not pull "something from the internet" — it takes a specific Chrome++ release and logs its `SHA256`, a fingerprint you can check against the published upstream artifact. What is downloaded is exactly what ships, with no substitution.

## Why antivirus sometimes complains

---

An unsigned DLL bearing a system name, placed right next to `chrome.exe` so it is loaded in place of the real one, is a textbook "DLL hijacking" shape to an antivirus heuristic. The heuristic reacts to the **shape**, not the contents: the file's body is irrelevant to it.

So this is a **false positive**. Microsoft Defender itself has periodically and mistakenly flagged Google Chrome and Electron apps the same way — this is a known, acknowledged limitation of such heuristics, not a property of the file.

It surfaces bluntly: Defender grabs `version.dll` mid-build and archive packaging dies with an unhelpful `OSError: [Errno 22] Invalid argument` — as if the disk were at fault, when in fact antivirus took the file (Windows returned `ERROR_VIRUS_INFECTED`), and the C runtime collapsed the unknown code to `EINVAL`).

## What the program does about it

---

- The **"Work around Defender while building"** checkbox (`guard_defender`, on by default) sits on `Build`, `Update`, and `Chrome++`. For the length of the operation the folder is excluded from Defender and restored **the moment it finishes — even if the build fails**. Windows asks for UAC once. `Build` excludes `output`; `Update` and `Chrome++` also exclude the build being updated (usually outside `output`).
- **If Defender is not running**, the checkbox does nothing and asks for nothing. The check is unelevated (the `WinDefend` service), so on machines without Defender the build runs exactly as before.
- The `Defender: remove exclusion` command (the `Service` section) removes the exclusion by hand — for the rare case where a run was killed hard and the exclusion lingered. Removing an exclusion that is not there is not an error.

The mechanism is safe by design: the exclusion lives only for the build and is held by three fail-safes (a lock file, parent-process liveness, a hard timeout), so it can never be left open silently.

## Two wrappers to choose from

---

The wrapper is picked in the `Portability` field, both when building and when updating:

| Engine                     | What goes into <code>App</code>                        | Registry                                           | Antivirus                                 |
|----------------------------|--------------------------------------------------------|----------------------------------------------------|-------------------------------------------|
| <code>Chrome++</code>      | <code>version.dll</code> and <code>chrome++.ini</code> | the branch is wiped on exit when the box is ticked | heuristics regularly take it for a threat |
| <code>Proxy library</code> | <code>version.dll</code> and <code>version.ini</code>  | writes are blocked, so nothing accumulates         | Microsoft does not flag it, see below     |

The profile stays inside the build folder either way: the library keeps it there itself.

**The proxy library** is the [neurostalker/proksi-biblioteka](https://github.com/neurostalker/proksi-biblioteka) project on GitFlic, with its Delphi sources in the same repository. It does the same job as Chrome++ and a little more: it blocks writes to the registry, clears the process mitigation bit that Chromium 118+ and 126+ need, drops metrics, and keeps the browser from creating folders outside the profile. It ships x86 and x64 only - ARM64 is covered by Chrome++ alone.

The program writes its own `version.ini` rather than shipping the author's. The author's defaults also rewrite the user agent to Yandex, mute traffic to Google's servers and block broadcasts; ours keeps to portability: `..\Data` and `..\Cache`, special folders inside the build, and `REGOFF` driven by the `Leave no traces in Windows` checkbox. The remaining keys are documented in the library's README - the file sits in `App` and can be edited by hand.

The downloaded wrapper archive is logged with its `SHA256`, as before: what is downloaded is what ships.

## The VirusTotal check, 17 August 2026

Both libraries, version 1.0.7.4, taken from the GitFlic release:

| File                                         | Size, bytes | SHA256                                                                        |
|----------------------------------------------|-------------|-------------------------------------------------------------------------------|
| <code>version</code><br><code>x32.dll</code> | 39,936      | <code>DBB82B80BB47DFAB47E09DFDA777478EC7A76599A4B47AFBE4D44C77C3EA0E03</code> |
| <code>version</code><br><code>x64.dll</code> | 91,136      | <code>FAEA4A01468ACDD54C390CBC3A18101ADF6F8F51525607EC2CAB19B42235AAAB</code> |

Neither is signed (`NotSigned`); the publisher field reads `Свободный софт`.

What the report said about both files:

- **Microsoft - Undetected.** That is the Defender engine, and that line is the whole point of the exercise.
- **NANO-Antivirus, Acronis (Static ML) - Undetected.**
- **Cynet - Malicious (score 100), DeepInstinct - MALICIOUS.** Both are pure machine-learning engines, and false positives on an unsigned Delphi library that hooks WinAPI functions are routine for them.

What the check does **not** prove:

- Microsoft's VirusTotal engine is signature-based. On a live machine Defender adds cloud heuristics and reputation-by-rarity, neither of which is in that scan, so the verdict there can differ.
- Neither the proxy library nor Chrome++ is signed, so to SmartScreen both are equally an unknown publisher.
- We did not test against a running Defender and could not: it is removed from the owner's machines - no `WinDefend`, no `WdFilter` - and every file looks clean there.

The VirusTotal report opens only past a reCAPTCHA, so a human opened the page, not the program: captchas are not something we work around.

## For whoever receives the build

---

The exclusion only helps **on the machine doing the build**. On the machine you hand the finished build to, Defender may object to `version.dll` on unpacking the same way — and the browser then quietly stops being portable (its profile moves into Windows). The fixes are one of: exclude the folder in their Defender, submit the file to Microsoft as a false positive, or sign `version.dll` with your own certificate (a future option).