

Klonbarhetsrevision — sammanfattande rapport

Vad en konkurrent kan läsa ut ur er app

Joplin desktop 3.7.21 — statisk analys av klienten

Objekt	Joplin desktop 3.7.21 (Electron, Windows x64)
Beställd av	Internt test (ingen extern beställare)
Utförd av	Axel Sjöberg, Kodro
Datum	15 oktober 2026
Klassificering	Konfidentiellt

Demonstrationsrapport, byggd på ett internt test av en öppen app (Joplin, MIT-licens). Ingen kund har beställt analysen. En skarp kundrapport levereras krypterat och endast till beställaren.

Om Kodro

Kodro utför klonbarhetsrevisioner: en oberoende granskning av vad som i praktiken går att läsa ut ur en levererad applikation med dagens AI-stödda analysverktyg. Granskningen görs statiskt på exakt den fil som kunden distribuerar, med kundens skriftliga tillstånd, och mynnar ut i en konkret, åtgärdbar rapport.

Syftet är defensivt: att visa en tillverkare vad en konkurrent eller en motiverad tredjepart kan utvinna ur produkten — källkod, affärslogik, oannonserade funktioner och eventuella inbäddade hemligheter — innan någon annan gör det. Kodro angriper aldrig produktionssystem och använder aldrig funna hemligheter.

Tjänst	Klonbarhetsrevision (statisk analys av levererad klient)
Utförs av	Axel Sjöberg, grundare
Kontakt	kontakt@kodro.se · kodro.se

Meddelanden och förbehåll

Upphovsrätt och spridning

© 2026 Kodro. Rapporten är konfidentiell och får inte reproduceras eller spridas, helt eller delvis, utan skriftligt medgivande. I ett skarpt uppdrag levereras rapporten krypterat och endast till beställaren. Detta exemplar är en demonstration byggd på en öppen app (Joplin, MIT-licens) och har inte beställts av någon kund.

Täckningsförbehåll

Granskningen är tidsbegränsad och bygger enbart på statisk analys av den levererade filen vid en viss tidpunkt. Den är inte en uttömmande förteckning över allt som kan utläsas ur produkten, och frånvaron av ett fynd är inte en garanti för att motsvarande problem inte finns. Resultatet gäller den analyserade versionen; en ny release kan införa nya läckor och bör granskas på nytt.

Oberoende

Kodro har inget ägande i eller annan koppling till det granskade projektet. Analysen utfördes på en publikt tillgänglig, verifierad installationsfil.

Innehåll

Om Kodro	2
Meddelanden och förbehåll	3
Uppdragssammanfattning	5
Analysobjekt	6
1 Sammanfattning för ledningen	7
2 Översikt över fynd	8
3 Detaljerade fynd	9
4 Åtgärdsplan	15
5 Rekommenderad uppföljning	16
Bilaga A Skolor och kategorier	17
Bilaga B Metod i detalj	18

Uppdragssammanfattning

Kontaktuppgifter

Granskare	Axel Sjöberg — grundare, Kodro · kontakt@kodro.se
Rapportansvarig	Axel Sjöberg

Tidslinje

15 okt 2026	Installationsfil hämtad från officiell GitHub-release och verifierad
15 okt 2026	Statisk analys genomförd
15 okt 2026	Rapport sammanställd och levererad

I ett skarpt uppdrag löper tidslinjen normalt över flera dagar, med en avstämning efter leverans och en möjlig omgranskning efter åtgärd.

Analysobjekt

Granskningen omfattar en (1) analysenhet: den officiella skrivbordsklienten för Windows.

Objekt	Joplin desktop
Version	3.7.21
Fil	Joplin-Setup-3.7.21.exe (361 008 040 byte)
Typ	Electron-applikation (NSIS-installer, app.asar)
Plattform	Windows x64
Källa	Projektets officiella GitHub-release
Verifiering	sha512 kontrollerad mot latest.yml
Licens	MIT (öppen källkod)

Serverlogiken (Joplin Cloud) och den inbäddade Chromium-runtimen ligger utanför omfattningen.

1 Sammanfattning för ledningen

Klonbarhet E	Exponering Medel	Klontest ~10 min
------------------------	----------------------------	----------------------------

Engagemangsöversikt

Kodro granskade den levererade Joplin-klienten statistiskt, med samma typ av verktyg som en konkurrent skulle använda, för att fastställa hur mycket av produkten som kan rekonstrueras utan tillgång till källkod eller servrar. Arbetet omfattade uppackning av installationsfilen, extraktion av appens kod och en genomgång av källfiler, source maps och inställningar.

Observationer och påverkan

Klienten avslöjar i praktiken hela klientsidans design: datamodell, hela E2EE-schemat, synkprotokollet och alla sync targets. Installeraren innehåller dessutom appens TypeScript-källfiler och webpack source maps med 2 723 originalsökvägar, så ingen dekompilering behövs — kärnfunktionen kunde specificeras på cirka tio minuter (Klonbarhet E). Joplin är dock avsiktligt open source (MIT), och klientkoden är redan fri att kopiera från GitHub.

Det allvarligaste enskilda fyndet är de medskickade source maps (Exponering Medel). Inga inbäddade produktionsnycklar hittades, AI-funktionens nyckelfält var tomta med integritetsvänliga standardvärden, och runtimen var aktuell. Det som skyddar affären — Joplin Cloud-tjänsten, kontologik och varumärke — ligger på servern och var inte åtkomligt.

Fallet visar varför de två måtten hålls isär: klienten är lätt att kopiera, men nästan inget känsligt läcker. För en proprietär produkt med samma öppenhet vore slutsatsen den omvända.

Kritisk	0
Hög	0
Medel	1
Låg	3
Info	2

Rekommendationer

1. Sluta skicka med source maps (.map) och .ts-filer i produktionspaketet (F01).
2. Ta bort oannonserade feature-flaggor ur releasebygget, eller håll dem i separata beta-kanaler (F02).
3. Dokumentera medvetet att klientsidans skydd ligger på servern, och bekräfta att inga framtida nycklar hamnar i klienten (F03, F06).

2 Översikt över fynd

<i>ID</i>	<i>Titel</i>	<i>Typ</i>	<i>Allvarlighet</i>
F01	Source maps och .ts-källfiler medskickade i produktionspaketet	Källkodsexponering	Medel
F02	Dolda feature-flaggor i releasebygget	Roadmap-läckage	Låg
F03	Komplett E2EE-schema avläsbart ur klienten	Affärslogik	Låg
F04	Synkprotokoll och alla sync targets kartläggbara	Affärslogik	Låg
F05	Agentisk AI-assistent med verktygskatalog synlig i koden	Dold funktion	Info
F06	Inga inbäddade produktionsnycklar; aktuell Electron-runtime	Hemlighet	Info

3 Detaljerade fynd

F01 Source maps och .ts-källfiler medskickade i produktionspaketet

Allvarlighet	● Medel	Typ	Källkodsexponering
Åtgärdsinsats	Liten	Fynd-ID	F01
Plats	resources/app.asar → main.bundle.js.map, main-html.bundle.js.map, 391 .ts-filer		

Beskrivning

Produktions-app.asar innehåller webpack source maps (main.bundle.js.map 3,2 MB, main-html.bundle.js.map 5,0 MB) och 391 .ts-källfiler. Source map:en listar 2 723 originalsökvägar och kartlägger hela den interna modulstrukturen (@joplin/lib/services/e2ee/..., Synchronizer.ts, alla sync targets). Den transpilerade koden har dessutom bevarade klass- och funktionsnamn.

Belägg (maskerat)

```
sources: 2723, sourcesContent: nej
../lib/services/e2ee/EncryptionService.ts
../lib/Synchronizer.ts
../lib/file-api-driver-joplinServer.ts
main.bundle.js.map = 3 318 651 byte
main-html.bundle.js.map = 5 287 396 byte
```

Påverkan för verksamheten

För en proprietär produkt vore detta nära en fullständig källkodsläcka: en konkurrent läser logiken ordagrant utan att dekompilera. För Joplin (MIT) är den direkta risken låg, men medskickade maps gör även eventuell closed-source-kod (t.ex. framtida kommersiella tillägg) trivial att läsa och ökar paketstorleken i onödan.

Rekommendation

1. Sätt devtool: false (eller hidden-source-map utan att packa .map) i produktionsbygget och exkludera .ts ur asar.
2. Lägg en kontroll i bygget som felar om .map eller .ts finns i app.asar i en release.

F02 Dolda feature-flaggor i releasebygget

Allvarlighet ● **Låg**

Typ Roadmap-läckage

Åtgärdsinsats Liten

Fynd-ID F02

Plats `main.bundle.js` - inställningsdefinitioner (`featureFlag.*`, `buildFlag.*`, `ai.*`)

Beskrivning

Releasebygget innehåller feature- och build-flaggor som inte är synliga för användaren (`public:false`) och som pekar ut funktioner under utveckling. De avslöjar roadmap och kan slås på av den som läser koden.

Belägg (*maskerat*)

<code>featureFlag.noteLock</code>	<code>default=false</code>	DOLD (<code>public:false</code>)
<code>featureFlag.conflictResolution</code>	<code>default=false</code>	DOLD
<code>buildFlag.voiceTypingEnabled</code>	<code>default=true</code>	DOLD
<code>featureFlag.enableSemanticSearch</code>	<code>default=true</code>	public (advanced)
<code>ai.enabled</code>	<code>default=false</code>	public
<code>ai.allowRemote</code>	<code>default=false</code>	public

Påverkan för verksamheten

Avslöjar kommande funktioner (not-låsning, egen konfliktlösning, röstdiktering) innan lansering. Begränsad direkt risk, men ger konkurrenter förspårning och kan slås på av nyfikna användare i ett halvfärdigt skick.

Rekommendation

1. Håll oannonserade flaggor i separata beta- eller nightly-kanaler, inte i stabil release.
2. Generera inställningskatalogen per releasekanal så att dolda flaggor strippas ur det stabila bygget.

F03 Komplet E2EE-schema avläsbart ur klienten

Allvarlighet	● Låg	Typ	Affärslogik
Åtgärdsinsats	Ingen (by design)	Fynd-ID	F03
Plats	main.bundle.js → @joplin/lib/services/e2ee/* (EncryptionService, crypto.ts, ppk/*)		

Beskrivning

Hela end-to-end-krypteringen kan läsas ur klienten: metodenum, cipher, nyckelhärledning och nyckeldelning. Ingen gissning behövs, parametrarna står i klartext i den transpilerade koden.

Belägg (*maskerat*)

```
EncryptionMethod: SJCL,SJCL2,SJCL3,SJCL4,SJCL1a,Custom,SJCL1b,KeyV1,FileV1,StringV1
StringV1/FileV1: cipherAlgorithm: AES_256_GCM, authTagLength:16,
  digestAlgorithm: sha512, keyLength:32, iterationCount: 220000
Äldre: SJCL (AES-CCM) för bakåtkompatibilitet
Nyckeldelning: ppk/ RSA via WebCrypto subtle
```

Påverkan för verksamheten

En tredjepart kan bygga en krypteringskompatibel klient utan att röra servern. För Joplin är detta avsett (interoperabilitet, öppen standard) och exponerar inga användarhemligheter, servern ser bara chifftext. För en proprietär produkt vore samma exponering en genväg till en kompatibel klon.

Rekommendation

1. Ingen åtgärd krävs för Joplin: designen är medveten och korrekt (hemligheter härleds klientsidan, servern ser bara chifftext).
2. För en kommersiell produkt: dokumentera att E2EE-parametrar är publika och att säkerheten vilar på nyckelhantering, inte på att schemat är hemligt.

F04 Synkprotokoll och alla sync targets kartläggbara

Allvarlighet ● **Låg**

Typ Affärslogik

Åtgärdsinsats Ingen (by design)

Fynd-ID F04

Plats `main.bundle.js` → `Synchronizer.ts`, `BaseSyncTarget`, `file-api-driver-*`

Beskrivning

Hela synkmekaniken och samtliga sync targets kan kartläggas ur klienten: abstraktionen, låshantering, schemaversioner och migrationer, delta-synk och konfliktobjekt, samt endpoints i klartext.

Belägg *(maskerat)*

```
BaseSyncTarget + file-api-driver-joplinServer / onedrive / dropbox / s3 / webdav /  
filesystem  
LockHandler, MigrationHandler (migrations 1-3), ItemUploader, lastProcessedChangeId  
(delta)  
Endpoints: api.joplincloud.com, joplinusercontent.com,  
           objects.joplinusercontent.com, s3.amazonaws.com
```

Påverkan för verksamheten

En konkurrent kan bygga en klient som synkar mot samma backends, eller en egen kompatibel server. För Joplin är protokollet avsett att vara öppet. Affärsvärdet skyddas av den hostade Joplin Cloud-tjänsten (konto, lagring, fakturering), inte av att protokollet är hemligt.

Rekommendation

1. Ingen åtgärd för Joplin: öppet protokoll är en medveten designprincip.
2. För en kommersiell produkt: lägg synk- och kontologik på servern och exponera bara ett minimalt, autentiserat API i klienten.

F05 Agentisk AI-assistent med verktygskatalog synlig i koden

Allvarlighet ● **Info**

Typ Dold funktion

Åtgärdsinsats Ingen

Fynd-ID F05

Plats `app-src/gui/ChatPanel/*`, `services/aiStatusBridge.ts`, `main.bundle.js` (ai.*)

Beskrivning

Den levererade koden innehåller en fullt inkopplad agentisk AI-chattassistent över anteckningar: Redux-actions, en ChatConversation-modell, en chattpanel och en komplett verktygskatalog. Assistenten är avstängd som standard (ai.enabled=false) men hela arkitekturen är avläsbar.

Belägg (maskerat)

```
ai.chat.providerType = "openai-compatible"    (OpenAI eller lokal Ollama)
ai.chat.baseUrl/apiKey/model = ""             (tomma som standard)
ai.allowRemote = false, ai.embedding.enabled = true
Verktyg: edit_current, search_notes, read_note, create_note,
         update_note, delete_note, manage_tags, semantic_search_notes
Lokal RAG: onnxruntime-node + sqlite-vec (embeddingar)
```

Påverkan för verksamheten

Avslöjar en betydande, delvis oannonserad funktion (agentisk AI med skriv- och raderverktyg mot användarens anteckningar) inklusive dess design och standardinställningar. Positivt ur integritetssynpunkt: inga nycklar inbäddade, fjärr-AI avstängt som standard, lokal embedding-väg finns. Risk främst som roadmap- och designläckage.

Rekommendation

1. Säkerställ att skriv- och raderverktyg (delete_note, update_note) kräver tydligt användarmedgivande.
2. Behåll apiKey-fältet tomt i bygget och lägg till en kontroll enligt F06.

F06 Inga inbäddade produktionsnycklar; aktuell Electron-runtime

Allvarlighet	● Info	Typ	Hemlighet
Åtgärdsinsats	Liten	Fynd-ID	F06
Plats	app-src/ (inställningar, AI-moduler), Joplin.exe (runtime)		

Beskrivning

Till skillnad från många klienter innehåller Joplin inga inbäddade produktionsnycklar eller tokens. AI-nyckelfältet är tomt som standard, fjärr-AI är avstängt, och OAuth-client-ID:n för OneDrive och Dropbox är per design publika i ett OSS-projekt. Runtime är dessutom aktuell.

Belägg (maskerat)

```
ai.chat.apiKey = ""          (tomt), ai.allowRemote = false
appId = "SET_ME"            (platshållare, ingen riktig nyckel)
Inga sk_/pk_/Bearer-/DSN-mönster med giltigt innehåll funna i egen kod
Runtime: Electron 42.3.0 / Chrome 148.0.7778.180 (aktuell)
```

Påverkan för verksamheten

Positivt: ingen omedelbar ekonomisk risk via läckt nyckel, och den aktuella Chromium-runtimen får säkerhetsuppdateringar. Detta är en styrka att bevara.

Rekommendation

1. Bevara nuvarande praxis: inga hemligheter i klienten, fjärr-AI opt-in.
2. Lägg till en kontroll i bygget som felar om ett icke-tomt apiKey eller token hamnar i app.asar.
3. Behåll den snabba Electron-uppgraderingskadensen.

4 Åtgärdsplan

Omedelbart

- Exkludera .map och .ts ur produktions-asar (webpack devtool: false, packa bara dist) (F01).

Kort sikt

- Flytta oannonserade feature-flaggor (noteLock, conflictResolution, voiceTypingEnabled) till separata förhandskanaler eller strippa ur stabil release (F02).
- Lägg till en kontroll i bygget som felar om en icke-tom apiKey eller token hamnar i klientpaketet (F06).

Strukturellt

- Behåll medvetet designen där all affärskritisk logik och alla hemligheter ligger på servern, och dokumentera det som ett arkitekturkrav (F03, F04).

5 Rekommenderad uppföljning

Kör en ny analys efter nästa större release, eller kvartalsvis via abonnemang, så att nya läckor — till exempel en framtida inbäddad AI-nyckel — upptäcks innan de når produktion. En omgranskning efter genomförd åtgärd bekräftar att source maps och dolda flaggor inte längre följer med releasen.

Bilaga A — Skalar och kategorier

Klonbarhet och exponering mäter olika saker och vägs var för sig. Klonbarheten anger hur snabbt kärnfunktionen kan återskapas; exponeringen anger det allvarligaste enskilda fyndet.

Klonbarhetsskala

Nivå	Klontest	Innebörd
A	Går inte	Kärnlogiken ligger på servern och kan inte rekonstrueras ur klienten.
B	Mer än en arbetsvecka	Kräver betydande dekompilering och egen återuppbyggnad.
C	En till fem arbetsdagar	Logiken är läsbar men utspridd; kräver sammanställning.
D	En till åtta timmar	Stora delar kan läsas direkt; begränsat återställningsarbete.
E	Under en timme	Koden kan läsas nästan ordagrant; specifikationen faller nästan ut direkt.

Exponeringsnivåer

Nivå	Innebörd
● Kritisk	Inbäddad produktionshemlighet med omedelbar ekonomisk eller säkerhetsmässig konsekvens (giltig nyckel, token, lösenord).
● Hög	Känslig affärslogik eller hemlighet som ger en konkurrent betydande och direkt fördel.
● Medel	Exponering som bör åtgärdas men utan omedelbar hemlighetsförlust (t.ex. medskickad källkod eller source maps).
● Låg	Avläsbar information av begränsat värde, ofta avsiktligt öppen eller svår att utnyttja.
● Info	Observation utan direkt risk; noteras för fullständighet eller som positivt fynd.

Fyndkategorier

Kategori	Innebörd
Källkodsexponering	Original- eller nära-originalkällkod är läsbar ur den levererade filen.
Roadmap-läckage	Oannonserade funktioner, flaggor eller planer syns i bygget.
Affärslogik	Intern logik, protokoll eller scheman kan kartläggas ur klienten.
Dold funktion	Funktionalitet som finns i koden men inte är exponerad för användaren.
Build-hygien	Byggartefakter (t.ex. source maps) följer oavsiktligt med i releasen.
Interna API:er	Klientens interna gränssnitt (IPC, bryggor) kan kartläggas ur paketet.
Hemlighet	Nycklar, tokens eller motsvarande — eller bekräftad frånvaro av sådana (positivt fynd).

Bilaga B — Metod i detalj

Analysen utfördes lokalt och defensivt. Inga produktionssystem kontaktades, installeraren kördes aldrig, och inga funna nycklar användes.

1. **Anskaffning och verifiering.** Den officiella installeraren hämtades från projektets GitHub-release och verifierades med sha512 mot latest.yml (storlek 361 008 040 byte).
2. **Uppackning utan körning.** De två inbäddade 7z-arkiven skars ut ur NSIS-stubben och packades upp. Appens egen kod (72 MB av 361 MB) extraherades ur resources/app.asar.
3. **Strängsökning.** Maskerad sökning efter nyckel- och hemlighetsmönster (sk_/pk_, Bearer, DSN m.fl.) i egen kod och beroenden.
4. **Kodläsning.** Genomgång av huvudprocessen, inställningsdefinitioner, källfiler och webpack source maps för att kartlägga modulstruktur, protokoll och funktioner.
5. **Runtime-kontroll.** Identifiering av Electron- och Chromium-version för att bedöma aktualitet.

Avgränsningar

Endast statisk analys av den levererade filen. Ingen nätverkstrafik mot testmiljö eller produktion observerades. Serverlogiken (Joplin Cloud) och den inbäddade Chromium-runtimen analyserades inte på instruktionsnivå.