

Klonbarhetsrevision — sammanfattande rapport

Vad en konkurrent kan läsa ut ur er app

Bitwarden Desktop 2026.10.0 — statisk analys av klienten

Objekt	Bitwarden Desktop 2026.10.0 (Electron, Windows x64)
Beställd av	Internt övningsuppdrag (ingen extern beställare)
Utförd av	Axel Sjöberg, Kodro
Datum	16 oktober 2026
Klassificering	Konfidentiellt

Övningsrapport, byggd på en öppen app (Bitwarden Desktop, GPL-3.0). 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 utan skriftligt medgivande. Detta exemplar är ett internt övningsuppdrag på en öppen app (Bitwarden Desktop-klienten, GPL-3.0) och har inte beställts av någon kund. Delar av Bitwardens SDK gick 2024 över till en egen, icke-öppen licens; dessa komponenter behandlas inte som öppen källkod och analyserades inte.

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 publikt tillgänglig källkod och en verifierad officiell release. Bitwarden är en säkerhetsprodukt: skulle en verklig, utnyttjbar sårbarhet hittas rapporteras den först ansvarsfullt till Bitwardens bug bounty-program, och inget publiceras förrän den är åtgärdad.

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	13
5 Rekommenderad uppföljning	14
Bilaga A Skalar och kategorier	15
Bilaga B Metod i detalj	16

Uppdragssammanfattning

Kontaktuppgifter

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

Tidslinje

16 okt 2026	Källkod (github.com/bitwarden/clients @ 3cd5ae6) och officiell release hämtade och verifierade
16 okt 2026	Statisk analys genomförd; klontest 13:32–13:46
16 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, jämförd mot den publicerade källkoden.

Objekt	Bitwarden Desktop
Version	2026.10.0
Fil	Bitwarden-2026.10.0-x64.appx
Verifiering	SHA-256 490d2d86...ae6c4
Källkod	github.com/bitwarden/clients @ 3cd5ae6
Typ	Electron-applikation (app.asar)
Plattform	Windows x64
Licens	GPL-3.0 (desktop-klienten)

Bitwardens liveservrar, API och konton samt de proprietära sdk-internal-komponenterna (egen licens sedan 2024) ligger utanför omfattningen och behandlas inte som öppen källkod.

1 Sammanfattning för ledningen

Klonbarhet E	Exponering Låg	Klontest ~14 min
------------------------	--------------------------	----------------------------

Engagemangsöversikt

Kodro granskade den levererade Bitwarden-klienten statistiskt, jämförd mot den publicerade källkoden, för att fastställa hur snabbt valvets kärnfunktion (datamodell, nyckelhärledning, valvkryptering och synkprotokoll) kan specificeras för kloning, och vad klienten läcker utöver det som redan är publikt.

Observationer och påverkan

Bitwarden Desktop går att specificera för kloning extremt snabbt — kärnfunktionen beskrevs som en byggbar spec på cirka fjorton minuter (Klonbarhet E). Det beror helt på att produkten är öppen källkod och att Bitwarden själva publicerar sitt säkerhetsprotokoll: KDF-parametrarna (PBKDF2-SHA256 600 000 iter., eller Argon2id 6·32 MiB·4) och valvets kryptodesign ligger öppet i källkoden och i det publicerade whitepaperet, och kompatibla tredjepartsklienter finns redan.

Exponeringen är samtidigt låg. Klienten innehåller inga inbäddade hemligheter, konto-tokens eller tredjepartsnycklar — zero-knowledge-designen bekräftades. De två åtgärdbara fynden är informationsläckage med begränsad påverkan: medskickade källkartor för vissa chunkar (F02) och en roadmap-läcka via feature-flaggor och interna ärendenummer (F03).

Detta är ett skolexempel på hög klonbarhet med låg exponering som inte är ett problem: öppenheten är ett medvetet val, och affärsvärdet ligger i den hostade tjänsten, revisionerna och ekosystemet — inte i att klientkoden hålls hemlig.

Kritisk	0
Hög	0
Medel	0
Låg	2
Info	2

Rekommendationer

1. Uteslut källkartor (.map) ur release-artefakten — ett generellt build-hygien-mönster (F02).
2. Säkerställ server-/behörighetsspärr för riskflaggan EnableBasicAuthResponse och rensa utgångna feature-flaggor (F03).
3. Behåll och kommunicera att säkerheten bygger på öppen, granskad design och zero-knowledge — ingen teknisk brådska finns (F01, F04).

2 Översikt över fynd

<i>ID</i>	<i>Titel</i>	<i>Typ</i>	<i>Allvarlighet</i>
F01	Inga inbäddade hemligheter i klienten (zero-knowledge bekräftad)	Hemlighet	Info
F02	Källkartor med originalkällkod följer med i releasen	Build-hygien	Låg
F03	Feature-flaggor och interna projektkoder exponerar roadmap	Dold funktion	Låg
F04	Valvets kryptodesign är fullt specificerbar ur klienten	Affärslogik	Info

3 Detaljerade fynd

F01 Inga inbäddade hemligheter i klienten (zero-knowledge bekräftad)

Allvarlighet	● Info	Typ	Hemlighet
Åtgärdsinsats	Liten	Fynd-ID	F01
Plats	app/main.js, main.js, preload.js (shippad bundle); libs/ (källkod)		

Beskrivning

Genomsökning av både den shippade klienten och källkoden efter API-nycklar, klient-hemligheter, privata nycklar och telemetri-DSN:er gav inga träffar. De enda hårdkodade värdena är Bitwardens publika standard-endpoints och hjälplänkar. apiKey/secret-träffarna i källkoden är inloggningsstrategier (API-nyckel-login där användaren själv har client_id/secret), inte inbäddade värden.

Belägg (maskerat)

```
# Maskerade exempel på de "längsta" strängarna (inga nycklar):  
'ABCDE...789+/'      # base64-alfabetet  
'juris...eName'      # ord/regex-fragment  
# Telemetri-/DSN-sökning: 0 träffar
```

Påverkan för verksamheten

Positivt fynd. En angripare som packar upp klienten får inga serverhemligheter, konto-tokens eller tredjepartsnycklar att missbruka. Detta är förväntat för en zero-knowledge-produkt och är en styrka: klientens säkerhet vilar inte på att binären hålls hemlig.

Rekommendation

1. Ingen åtgärd krävs.
2. Behåll rutinen att aldrig bygga in serverhemligheter i klientpaket; verifiera i CI att inga nycklar läcker vid framtida releaser.

F02 Källkartor med originalkällkod följer med i releasen

Allvarlighet ● **Låg**

Typ Build-hygien

Åtgärdsinsats Liten

Fynd-ID F02

Plats 815.js.map, app/vendor.js.map, app/main.*.css.map i app.asar

Beskrivning

Den shippade klienten innehåller källkartor (.map) med fältet `sourcesContent`, dvs. den ursprungliga källkoden. Det gäller en lazy-laddad funktionschunk (815.js, 0,3 MB), vendor-biblioteken (vendor.js.map, 9,9 MB) och CSS. Appens egna kärnbundlar (app/main.js 19 MB och Electron-main.js) shippas däremot minifierade utan källkartor — god hygien för huvudkoden.

Belägg (maskerat)

```
815.js           → sista raden: //# sourceMappingURL=815.js.map
815.js.map       → innehåller "sourcesContent" (originalkod)
app/vendor.js.map → innehåller "sourcesContent" (9,9 MB)
app/main.js      → ingen sourceMappingURL (minifierad, 535 rader)
main.js          → ingen sourceMappingURL (minifierad)
```

Påverkan för verksamheten

För Bitwarden är påverkan liten: koden är redan publicerad under GPLv3, så källkartorna avslöjar inget utöver det publika repot. Observationen tas ändå med eftersom samma byggkedja, tillämpad på en proprietär produkt, skulle lämna ut originalkällkod direkt ur den levererade filen — då vore detta ett Medel/Högt fynd.

Rekommendation

1. Uteslut .map-filer ur release-artefakten (sourceMap: false för produktionsbygget, eller rensa dem i paketeringssteget).
2. Ladda i stället upp källkartor separat till en symbolserver för felrapportering, inte in i den distribuerade appen.

F03 Feature-flaggor och interna projektkoder exponerar roadmap

Allvarlighet ● **Låg**

Typ Dold funktion

Åtgärdsinsats Medel

Fynd-ID F03

Plats `libs/common/src/enums/feature-flag.enum.ts` (finns även i shippad bundle)

Beskrivning

Klienten innehåller en komplett lista med 70 feature-flaggor och 43 unika interna projektkoder (pm-#####). Flera avslöjar opublicerad eller pågående funktionalitet (Windows-autotype, native credential sync, SSH-ECDSA m.m.). Två sticker ut: `DebugDisableSelfHostPremiumCheck` (debug-flagga för premium-kontroll) och `EnableBasicAuthResponse`, kommenterad som att den gates en säkerhetsrisk.

Belägg (maskerat)

```
/* Billing */
DebugDisableSelfHostPremiumCheck = "debug-disable-self-host-premium-check",
// Note: This flag gates security risks and should not be turned on without
// changes to the underlying experience
EnableBasicAuthResponse = "enable-basic-auth-response",
/* Desktop Native */
WindowsDesktopAutotype = "windows-desktop-autotype",
```

Påverkan för verksamheten

En konkurrent eller analytiker kan läsa av produktplanen och interna ärendenummer och se vilka funktioner som är på väg men inte annonserade. Påverkan är begränsad eftersom flaggorna i slutändan styrs av servern och listan dessutom är publik via open source, men den avslöjar internt arbete tidigare än önskat och antyder att en riskabel funktion finns i kodbasen.

Rekommendation

1. Säkerställ att riskflaggor som `EnableBasicAuthResponse` har server-/behörighetsspärr, inte bara en kommentar.
2. Följ kodbasens egen regel (flaggor ska vara kortlivade och tas bort när de aktiverats); rensa utgångna flaggor och överväg att inte exponera interna ärendenummer i klientsträngar.

F04 Valvets kryptodesign är fullt specificerbar ur klienten

Allvarlighet	● Info	Typ	Affärslogik
Åtgärdsinsats	Liten	Fynd-ID	F04
Plats	libs/legacy-crypto/src/models/kdf-config.ts, libs/common/src/key-management/*		

Beskrivning

Hela klonkärnan enligt scope — datamodell, nyckelhärledning och valvkryptering — är dokumenterad i klartext i klientens källkod (och motsvarar det publicerade säkerhetswhitepaperet). Nyckelhärledningen är standardprimitiver med explicita parametrar. De faktiska krypto-operationerna anropas numera via den proprietära sdk-internal (utanför scope), men parametrarna räcker för att återimplementera dem. Ett positivt fynd: explicit downgrade-skydd vid prelogin.

Belägg (*maskerat*)

```
export class PBKDF2KdfConfig {
  static ITERATIONS = new RangeWithDefault(600_000, 2_000_000, 600_000);
  static PRELOGIN_ITERATIONS_MIN = 5000; // ...
}
export class Argon2KdfConfig {
  static MEMORY = new RangeWithDefault(16, 1024, 32);          // MiB
  static PARALLELISM = new RangeWithDefault(1, 16, 4);
  static ITERATIONS = new RangeWithDefault(2, 10, 6);
}
// prelogin: "...possible pre-login downgrade attack detected."
```

Påverkan för verksamheten

En kompatibel klient kan specificeras på minuter eftersom designen är öppen och publicerad — detta är grunden för Klonbarhet E. Det är dock by design: Bitwarden är open source och publicerar sitt protokoll, och affärsvärdet ligger i den hostade tjänsten. Att den tyngsta kryptan flyttats till en icke-öppen SDK höjer tröskeln något för en fullständig återimplementation.

Rekommendation

1. Ingen åtgärd — öppenheten är ett medvetet val och en förtroendestyrka.
2. Fortsätt kommunicera att säkerheten bygger på granskad, öppen design snarare än hemlighållen klientkod; behåll downgrade-skyddet och server-sidig parametervalidering.

4 Åtgärdsplan

Omedelbart

- Uteslut .map-filer ur release-artefakten (F02).

Kort sikt

- Verifiera server-/behörighetsspärr för EnableBasicAuthResponse och andra riskflaggor (F03).
- Rensa utgångna feature-flaggor och överväg att inte exponera interna ärendenummer i klientsträngar (F03).

Strukturellt

- Behåll CI-kontroll som fångar inbyggda hemligheter och oavsiktliga källkartor före release (F01, F02).
- Fortsätt att bygga förtroende på öppen, granskad design och zero-knowledge snarare än hemlig klientkod (F04).

5 Rekommenderad uppföljning

Kör en ny analys efter nästa större release, särskilt om SDK-gränsen eller byggkedjan ändras, så att nya läckor (källkartor, hemligheter, riskflaggor) upptäcks innan de når produktion.

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. Endast statisk analys. Inga produktionssystem, API:er eller konton kontaktades, och inga funna nycklar användes (inga fanns).

1. **Anskaffning och verifiering.** Källkod hämtades från [@ 3cd5ae6](https://github.com/bitwarden/clients) (grund- och sparse-klon av `libs/` och `apps/desktop/src`). Den shippade releasen `Bitwarden-2026.10.0-x64.appx` verifierades med SHA-256.
2. **Uppackning.** `app.asar` extraherades ur den shippade appen och packades upp lokalt.
3. **Strängsökning.** Maskerad sökning efter nyckel-, token- och DSN-mönster i både källa och shippad bundle.
4. **Kodläsning.** Genomgång av KDF- och krypto-konfiguration, feature-flagg-enum och endpoints; jämförelse mellan källkod och shippad bundle.
5. **Avgränsning av SDK.** De faktiska krypto-operationerna i `sdk-internal` (egen licens) behandlades inte som öppen källkod och analyserades inte; endast publika parametrar och protokoll granskades.

Avgränsningar

Endast statisk analys. Synkprotokollet observerades inte live (nätverkskontakt utanför tillstånd) utan bedömdes ur den publika källkoden. De proprietära `sdk-internal`-komponenterna analyserades inte. Extraktion och analys gjordes med standardverktyg (`git`, `Node`, `grep`), vilket var tillräckligt för ett Electron-/JS-mål.