Anti-kloon research: hoe bescherm je een SaaS tegen kopiëren?

Voor: Easy TS (VCA-certificerings-tool voor installatiebedrijven)
Doel: wat doen SaaS-bouwers als ze hun product aan andere bedrijven verkopen (white-label, multi-tenant, ook aan concurrenten)?
Datum: 4 augustus 2026
Vorm: per onderdeel wat het is, hoe het werkt, echte voorbeelden met bron, en of het past op onze VCA-tool.

Kern in één zin: technische sloten vertragen een kopieerder, maar de echte bescherming is dat de waarde op je eigen server zit, in je data, in je contracten en in je snelheid.

---

1. Server-side core: de waarde verlaat nooit je server

Wat het is. De belangrijkste architectuur-regel in de hele softwarewereld: de client (browser, app) is vijandig gebied. Alles wat daar draait kan iedereen lezen. Dus: alle echte logica, berekeningen en regels draaien op je eigen server. De browser is alleen een scherm.

Hoe het werkt.
- De client stuurt alleen **bedoelingen** (klik, invoer). De server beslist wat er echt gebeurt.
- In de game-industrie heet dit **server-authoritative**: "client input is a suggestion, server state is the truth" (AccelByte).
- Dezelfde regel geldt voor SaaS: de client bevat geen geheimen, geen business-regels, geen sleutels. Zelfs obfuscatie-redenen noemen dit expliciet: "do not store secrets or sensitive data in frontend code" (nextjs-webpack-obfuscator, GitHub).
- Praktisch patroon: dunne frontend + API. Je API is het product. Wie je site kopieert heeft alleen de etalage, niet de winkel.

Echte ervaring. Op Stack Overflow is het antwoord op "kan ik mijn client-side JS beschermen?" al jaren hetzelfde: **nee**, niet te doen, de-obfuscatie is triviaal. Wil je iets geheim houden, dan moet het op de server draaien (Stack Overflow). ionCube (verkopers van code-bescherming!) zegt het zelf ook: op de server kunnen we je code beschermen, in de browser niet (ionCube blog).

Past dit op onze VCA-tool? Ja, dit is pilaar nummer 1.
- VCA-specifieke logica (welke documenten verplicht zijn, checklists, geldigheid van certificaten, herinneringsregels) hoort in de backend.
- Zorg dat een gekopieerde frontend een lege shell is zonder de examen-/checklist-data en regels.
- Bonus: koppel de tool aan externe data (VCA-registers, normen-updates). Dat kan een kloon nooit meenemen.

---

2. Licensing en tenant-isolatie: de klant kan de installatie niet meenemen

Wat het is. Als je multi-tenant of white-label verkoopt, wil je twee dingen: elke klant ziet alleen zíjn data, en niemand kan "de hele installatie" kopieren en zelfstandig doordraaien.

Hoe het werkt.
- **Per-tenant data-isolatie.** Het moderne patroon is Row-Level Security in de database (bijv. PostgreSQL RLS): de database zelf weigert data van een andere tenant, ook als de applicatie een fout maakt. Productie-voorbeeld: multi-tenant SaaS platform met PostgreSQL RLS. Let op: white-label-platforms zonder strikte tenant-scheiding zijn "one missed query away" van een ramp (Pharos).
- **Feature-flags per tenant.** Functies aan/uit per klant, per abonnementsvorm, zonder nieuwe deploy (ConfigCat). Zo verkoop je smaken van hetzelfde product en kun je bij wanbetaling of ruzie per tenant dimmen.
- **Licentie-validatie server-side.** De gouden regel bij licentiesleutels: nooit alleen in de client controleren. Validatie op de server, bij elke start of met een heartbeat, met activatie-limieten per installatie en intrekken via webhooks. "No system is 100% piracy-proof, maar server-side enforcement dekt 95%+ van casual piracy" (Fungies.io). Tooling: Keygen.sh, Cryptolens, LicenseSpring (Keygen docs). Concreet spec-voorbeeld van zo'n inbouw: WorkLenz + Keygen.sh met grace period en feature-gating.
- **Belangrijkste punt bij white-label:** de klant host zelf NIETS. Jij host. De klant krijgt een subdomein of eigen domein op jouw infrastructuur. Dan is er letterlijk geen installatie om mee te nemen.

Past dit op onze VCA-tool? Ja.
- Easy TS draait centraal bij jullie. Concurrenten die de tool white-label afnemen krijgen accounts, geen code.
- Per tenant: eigen data-isolatie (RLS), eigen feature-flags, eigen branding.
- Licentie-check in de backend bij login/API-calls. Stopt de betaling, stopt de toegang.

---

3. Canary traps en honeytokens in data: bewijs dat het van jou komt

Wat het is. Je verstopt in je data of content unieke, onzichtbare sporen. Als die sporen bij een concurrent opduiken, heb je bewijs van kopiëren. Dit is oud en bewezen: kaartenmakers deden het al met nep-straatjes.

Echte voorbeelden.
- **Genius vs Google (2019).** Lyrics-site Genius verving apostrofes door een patroon van rechte en gekrulde apostrofes. Dat spelde "REDHANDED" in morsecode. Het patroon dook op in Googles zoekresultaten: bewijs van scrapen (PCMag, Variety).
- **Trap streets.** Kaartenmakers stoppen nep-straten in kaarten. De AA in Engeland betaalde in 2001 een schikking van **20 miljoen pond** aan Ordnance Survey nadat kopiëren zo was bewezen (Atlas Obscura).
- **Fictieve entries in woordenboeken.** Het New Oxford American Dictionary verzon het woord "esquivalience" puur om kopieerders te pakken (lexpress.mu). Advocaten adviseren dit expliciet voor naslagwerken en datasets: "fictitious entries … to trace copying by competitors" (Heer Law).
- **Honeytokens in databases.** Nep-records (nep-klant, nep-credential) die alarm slaan zodra iemand ze aanraakt of exporteert (CrowdStrike, CounterCraft).

Eerlijke waarschuwing. Bewijs hebben is niet hetzelfde als winnen. Genius **verloor** uiteindelijk: zij bezaten zelf de rechten op de songteksten niet, dus er was geen inbreuk op hún recht (Variety, The Playground). Les: de trap levert bewijs, maar je moet ook eigenaar zijn van het gekopieerde.

Past dit op onze VCA-tool? Ja, goedkoop en sterk.
- Verstop in jullie VCA-checklists, teksten en documenttemplates **unieke formuleringen en micro-variaties per klant/tenant**.

- Eén of twee fictieve, onopvallende voorbeeld-records of voorbeeldbedrijven in demo- en voorbeelddata.
- Per-tenant subtiele verschillen in gegenereerde content. Komt de tekst van klant X bij een concurrent terug, dan weet je precies via wie het lekte.
- Dit is legale bewijsvoering, geen backdoor. Helemaal binnen de grenzen.

---

4. Watermarking in output: rapporten en PDF's traceren

Wat het is. Elk document dat de tool uitspuugt (VCA-rapport, certificaat-overzicht, PDF) krijgt een stempel. Zichtbaar of onzichtbaar.

Hoe het werkt.
- **Zichtbaar:** logo + "gegenereerd door Easy TS" in de footer. Bij white-label bewust kiezen: mag de reseller het merk weghalen, of niet? Dat is een contractuele en commerciële keuze, geen technische.
- **Onzichtbaar (forensisch):** per ontvanger een uniek watermerk in layout, tekstlaag of metadata. Bij een lek upload je het verdachte document en je ziet wiens kopie het was. Dit heet **transactional watermarking** (InkShield, PhantomPDF).
- **Eerlijke beperkingen:** een watermerk overleeft een doorgestuurde PDF prima, maar geen screenshot, overtypen of agressieve hercompressie (sp-tracer, GitHub). Forensische watermarking werkt **na** het lek (attributie), niet ervoor (preventie). Dat is DRM, en dat heeft zijn eigen grenzen (Fora Soft).

Past dit op onze VCA-tool? Ja, deels.
- Zichtbaar: "Powered by Easy TS" op rapporten. Gratis marketing + eigendoms-signaal.
- Onzichtbaar: per-tenant unieke ID in PDF-metadata en een subtiel tekst-spoor in gegenereerde documenten. Goedkoop om in te bouwen bij de PDF-generator, waardevol bij een dispuut.
- Geen dure DRM nodig. Dit is een certificeringstool, geen Hollywood-film.

---

5. Obfuscatie: waar het helpt en waar het lariekoek is

Wat het is. Client-side code (JavaScript) onleesbaar maken: namen hashen, strings encoden.

Wat eerlijk is.
- Het is **geen beveiliging**. De browser moet de code kunnen lezen, dus iedere gebruiker ook. De-obfuscatie is technisch triviaal (Stack Overflow).
- Het is hooguit **vertraging** tegen luie kopieerders. Jscrambler (verkoper ervan) positioneert het zelf als een laag, niet als oplossing (Jscrambler).
- Minificatie (die je toch al doet voor snelheid) geeft 80% van het effect gratis.
- Gevaar: wie op obfuscatie vertrouwt stopt per ongeluk echte logica in de client. Dat is precies de fout uit onderdeel 1.

Past dit op onze VCA-tool? Minimaal.
- Gewone productie-minificatie: ja, doen we toch.
- Dure obfuscatie-tooling: nee, niet nodig als de logica server-side zit.
- Energie steken in onderdeel 1 en 2 in plaats van hier.

---

6. Juridisch: wat werkt echt afdwingbaar (NL/EU)

Wat het is. De laag die vaak het sterkst is: eigendom vastleggen en contractueel dichttimmeren.

Wat in de praktijk werkt.
- **Auteursrecht heb je automatisch** op code en teksten. Je hoeft niets te registreren in NL. Wel: leg vast wie het maakte en wanneer (git-historie, i-Depot bij de KvK/Belastingdienst als bewijs van datum).
- **Licentieovereenkomst is het werkpaard.** De Nederlandse praktijk-checklist: rechten duidelijk omschrijven, auteursrecht behouden, **reverse engineering en herdistributie uitsluiten**, regels voor beëindiging (SenS Juristen).
- **White-label contract:** expliciet "Provider behoudt alle IP, reseller zal dat niet betwisten" plus verbod op reverse engineering, sublicentie en meenemen van klantdata (solvLegal). Bij embedded/managed modellen: scope, aansprakelijkheid en update-verplichtingen vastleggen (Logan & Partners).
- **NDA's:** nuttig in de verkoopfase, maar een NDA tegen "een concurrent die gewoon kijkt wat je publieke product doet" werkt niet. Ideeën en functionaliteit zijn niet beschermd; alleen de concrete uitwerking (code, tekst, design) wel.

- **Source-code escrow:** klanten vragen dit soms (continuïteit als jij failliet gaat). Kan prima, maar met strikte vrijgave-voorwaarden (faillissement, staken van support) en nooit vrij te gebruiken voor competitie (Traverse Legal).
- **Bedrijfsgeheim (Wet bescherming bedrijfsgeheimen, NL implementatie van de EU-richtlijn):** werkt alleen als je het geheim ook echt als geheim behandelt (beperkte toegang, contractuele geheimhouding). Een gekopieerde frontend is per definitie geen geheim meer.
- **Echte ervaring van founders:** bij een pixel-voor-pixel gekopieerde UI adviseren mede-founders: documenteer met timestamps (Wayback Machine), stuur een cease and desist, en accepteer dat "opnieuw nagebouwd vanaf nul" juridisch bijna niet aan te pakken is (terms.law thread). De HN-thread over een volledig gekloonde B2B-SaaS-site eindigt in hetzelfde: DMCA/hosting-melding, watermerken, en verder vooral harder doorbouwen (Ask HN via Gigazine, HN zelf).

Past dit op onze VCA-tool? Ja, dit is verplichte kost.
- Één goede SaaS-/white-label-overeenkomst laten opstellen door een ICT-jurist. Kern: IP blijft bij Easy TS, geen reverse engineering, geen dataroof bij vertrek, duidelijke beëindiging.
- Merk "Easy TS" registreren (BOIP, Benelux-merk). Goedkoop, en beschermt de naam die klanten kennen.
- Bewaar git-historie en ontwerpdocumenten als datum-bewijs.

---

7. De belangrijkste les: welke moat werkt ECHT?

Het patroon uit alle bronnen: technische sloten zijn een drempel, geen muur. Wie echt wil kopiëren bouwt je product na. De ervaringen van founders en de literatuur wijzen consequent dezelfde kant op:

- **Kopiëren is normaal en onvermijdelijk.** De Amazon-seller-community zegt het bot: "Dat is de vrije markt. Zonder octrooi en bij exact dezelfde fabricage kun je niets doen" (Amazon seller forums). Clonen is zelfs een erkende strategie; kloners zoeken juist producten die zijn gestopt met innoveren (SaaSify). Omgekeerd: **blijf bewegen en de kloon loopt altijd achter**.
- **De kloon mist het proces, niet alleen het product.** Een founder die een concurrent 6 maanden kopieerde concludeerde: kopieer hun onderzoeksproces, niet hun output, want

je ziet van buiten nooit waarom keuzes zo zijn (Indie Hackers). Die "onzichtbare kennis" zit bij Easy TS in jullie domeinkennis van VCA en installatiebedrijven.

- **Werkende moats:** data (jullie opgebouwde VCA-content, normen, klantgebruik), integraties (koppelingen die een kloon opnieuw moet bouwen), merk en vertrouwen (in certificering telt betrouwbaarheid zwaar), netwerkeffecten en snelheid (Andrew Chen, The Cold Start Problem, SaaS Link Building over content moats).
- **Genius-waarschuwing nogmaals:** je kunt kopiëren bewijzen en tóch verliezen als je eigen positie juridisch zwak is. Eigendom eerst, dan pas vallen.

Past dit op onze VCA-tool? Dit is de strategie.
- De moat van Easy TS is: de actuele VCA-kennisbank + certificeringsworkflows + integraties + het vertrouwen dat installatiebedrijven erin hebben.
- Technische maatregelen (1 t/m 4) zorgen dat niemand er gratis en straffeloos mee vandoor kan. Dat is hun enige taak.

Aanbevolen pakket voor Easy TS

WEL doen, in deze volgorde

1. **Architectuur dichttrekken (gratis, nu).** Alle VCA-logica, regels en data server-side. Frontend is een dunne shell. Dit is 80% van de bescherming.
2. **Hosting bij jezelf houden.** White-label = account op jullie platform, nooit een eigen installatie bij de klant. Er valt dan niets mee te nemen.
3. **Contract laten opstellen (ICT-jurist, eenmalig).** IP-behoud, verbod op reverse engineering en herdistributie, regels bij beëindiging. Plus Benelux-merkregistratie "Easy TS".
4. **Per-tenant watermerken en canary-data (klein dev-klusje).** Unieke tekst-variaties en ID's per tenant in checklists en PDF's. Bewijs bij lekken, kost weinig.
5. **Zichtbaar merk op output.** "Powered by Easy TS" op rapporten waar het contract dat toelaat.
6. **Blijf versnellen.** Nieuwe features, integraties, actuele normen. De beste anti-kloon is een kloon die altijd 6 maanden achterloopt.

NIET doen

- **Geen dure JS-obfuscatie-tooling.** Minify is genoeg; de rest is schijnveiligheid.
- **Geen DRM of "kill switches" in geleverde software.** Juridisch en reputatie-gewoonvragend, en onnodig als jij host.
- **Geen self-hosted licentie-sleutelcircus** zolang je zelf host. Pas relevant als je ooit on-premise gaat leveren (dan: server-side validatie met heartbeat, zie onderdeel 2).
- **Niet bouwen op NDA's als hoofdverdediging.** Ze helpen in de deal-fase, niet tegen kopieerders van je publieke product.

Kernzin voor het team

Sloten vertragen, bewijzen helpen, contracten beschermen, maar de moat is data + integraties + snelheid + merk. Besteed daar 90% van de energie.

Bronnen (25)

1. AccelByte: server-authoritative logic
2. Stack Overflow: client-side JS is niet te beschermen
3. ionCube: waarom client-side niet te beschermen is
4. nextjs-webpack-obfuscator (GitHub)
5. Multi-tenant SaaS met PostgreSQL RLS (GitHub)
6. ConfigCat: feature flags per tenant
7. Fungies.io: license keys developer guide
8. Keygen.sh docs
9. WorkLenz: Keygen-licentie spec (GitHub issue)
0. Pharos: white-label development risico's
1. PCMag: Genius vs Google watermerk
2. Variety: Genius verliest de zaak
3. The Playground: Genius-zaak bij Supreme Court
4. Atlas Obscura: trap streets, AA vs Ordnance Survey
5. Esquivalience: verzonnen woordenboek-entry
6. Heer Law: fictitious entries als detectie

7. CrowdStrike: honeytokens
8. CounterCraft: canary tokens uitgelegd
9. InkShield: leak tracing
10. PhantomPDF: transactionele PDF-watermerken
11. Fora Soft: forensische watermarking vs DRM
12. sp-tracer: eerlijke beperkingen van watermerken (GitHub)
13. Jscrambler: obfuscation guide
14. SenS Juristen: software-licentieovereenkomst checklist (NL)
15. solvLegal: white-label SaaS overeenkomst
16. Logan & Partners: software-distributie juridisch
17. Traverse Legal: source-code escrow
18. terms.law: concurrent kopieerde mijn UI (founders-thread)
19. Gigazine over Ask HN: gekloonde SaaS-site en de HN-thread zelf
30. SaaSify: ik kloonde een SaaS, dit leerde ik
31. Indie Hackers: concurrent kopiëren werkte niet
32. Andrew Chen: The Cold Start Problem (netwerkeffecten als moat)